Efficient and Robust Algorithms for Sparse 3-D Reconstruction from Image Sequences

Der Technischen Fakultät der
Universität Erlangen-Nürnberg

zur Erlangung des Grades

D O K T O R - I N G E N I E U R

vorgelegt von

Timo Zinßer

Erlangen — 2011

Als Dissertation genehmigt von
der Technischen Fakultät der
Universität Erlangen-Nürnberg

Tag der Einreichung:	21.01.2011
Tag der Promotion:	26.07.2011
Dekan:	Prof. Dr.-Ing. Reinhard German
Berichterstatter:	Prof. em. Dr.-Ing. Dr. h.c. Heinrich Niemann
	Prof. Dr.-Ing. Gerhard Sagerer

Abstract

This thesis is concerned with algorithms for the reconstruction of the scene structure and the camera motion of an input image sequence. In particular, we focus on algorithms for sparse 3-D reconstruction, which represent the scene structure with a limited number of feature points. In order to facilitate the application of the developed algorithms in the augmented reality gear of a cognitive vision system, our main focus lies on their efficiency and robustness. Furthermore, the versatility of the developed algorithms is demonstrated by the wide range of additional applications, like object tracking, object revisualization, and the reconstruction of objects from fiberscopic images. In this thesis, the developed algorithms are categorized into three work areas: feature point tracking, structure and motion estimation, and point set registration.

Our feature point tracking system is based on the Kanade-Lucas-Tomasi tracker. We propose several enhancements to increase its efficiency and robustness, like an efficient hierarchical feature selection strategy, a new result propagation strategy for hierarchical translation estimation, and the efficient integration of an affine linear model for intensity equalization. In our experiments, the developed feature point tracking system proves to be robust to strong intensity changes and severe motion blur. At the same time, it is able to track 150 feature points at a rate of 60 frames per second. We also present a special-purpose high-speed tracking system for the self-localization of an augmented reality gear. This system is able to track ten feature points at a rate of 200 frames per second on a relatively slow mobile computer.

In addition to the estimated 2-D feature positions, our proposed structure and motion estimation system requires the intrinsic camera parameters as input. The system achieves a high versatility by combining several efficient algorithms for performing the initial reconstruction of the scene with an accurate bundle adjustment approach for optimizing the obtained result. Its robustness to outliers is ensured by the comprehensive application of M-estimators and the least median of squares technique. In this work area, we also propose the extension of the POSIT algorithm with a virtual reference point and a new key frame selection algorithm. In our experiments, the estimation system successfully copes with outlier percentages of up to 40%. It also reaches computation rates of 50 frames per second.

Our work on point set registration is based on the locally convergent ICP algorithm. We compare three different techniques for increasing the robustness of this algorithm to outliers. Our extensive experimental evaluation shows that the least fractional squares extension provides the best overall performance. Furthermore, we propose an extension for the integrated estimation of the scale factor between two point sets. As long as the initial motion between the two point sets lies within the basin of convergence of the ICP algorithm, this extension allows the accurate registration of two differently scaled point sets.

Acknowledgements

I started my work on this thesis as a researcher at the Chair for Pattern Recognition of the Faculty of Engineering of the University Erlangen-Nuremberg. Together with fellow computer vision researchers from four European universities, I worked within the VAMPIRE project, which was funded by the European Union.

First of all, I would like to thank my supervisor Professor em. Dr. Dr. h.c. Heinrich Niemann for giving me the opportunity to work in a very interesting and active area of research. His constructive criticism and his well-founded comments have significantly improved the quality of this thesis. I am especially grateful for his continuous support after his retirement.

One of the main advantages of the VAMPIRE project was the very high level of cooperation between the project partners, which yielded a very stimulating work environment. I would like to thank the project coordinator, Professor Dr. Gerhard Sagerer, for contributing to this environment with his personal commitment, and for acting as the second reviewer of this thesis. With respect to the other project partners, I will never forget the exciting cooperation with Professor Dr. Axel Pinz and Dr. Christoph Stock of the Graz University of Technology.

At the Chair for Pattern Recognition, my colleague and office mate Christoph Gräßl was the second full-time researcher within the VAMPIRE project. Working with him has always been a great pleasure for me, because our skills and interests complemented each other perfectly. I would also like to thank Dr. Ingo Scholz and Dr. Florian Vogt for creating and sharing several video sequences with known camera motion, which greatly facilitated the experimental evaluation of my algorithms. Many other colleagues encouraged and supported me in different ways. In particular, I want to express my gratitude to Professor Dr. Joachim Denzler, Professor Dr. Joachim Hornegger, Frank Mattern, Kristina Müller, Professor Dr. Elmar Noeth, Professor Dr. Dietrich Paulus, and Friedrich Popp. Last but not least, I would like to thank Professor Dr. Jochen Schmidt for proofreading this thesis.

My work on this thesis required more time than I had ever anticipated. I am eternally grateful to my wife Steffi for her never-ending patience, her much-needed encouragement, and her generally invaluable support.

Timo Zinßer

Contents

1 Introduction **1**
- 1.1 Sparse 3-D Reconstruction in VAMPIRE . . . 1
- 1.2 Related Work . . . 5
- 1.3 Contribution of this Work . . . 7
- 1.4 Outline . . . 9

2 Basic Principles of Image Formation **11**
- 2.1 Overview . . . 11
  - 2.1.1 Image Acquisition Fundamentals . . . 11
  - 2.1.2 The Digital Image . . . 12
- 2.2 Geometric Image Formation . . . 14
  - 2.2.1 Coordinate Systems . . . 14
  - 2.2.2 Projection Models . . . 16
  - 2.2.3 Camera Parameters . . . 18
  - 2.2.4 Real Lenses . . . 19
- 2.3 Radiometric Image Formation . . . 22
  - 2.3.1 Light and Color . . . 22
  - 2.3.2 Sources, Surfaces, and Sensing . . . 23
  - 2.3.3 Gamma Correction . . . 24
  - 2.3.4 Real Cameras . . . 26
- 2.4 Multiple View Relations . . . 27
  - 2.4.1 Epipolar Geometry . . . 27
  - 2.4.2 The Fundamental Matrix . . . 29
  - 2.4.3 The Essential Matrix . . . 29
- 2.5 Summary . . . 31

3 Feature Point Tracking **33**
- 3.1 Overview . . . 34
  - 3.1.1 Problem Statement . . . 34
  - 3.1.2 Related Work . . . 38
- 3.2 The Kanade-Lucas-Tomasi Tracker . . . 40
  - 3.2.1 Basic Principles . . . 41
  - 3.2.2 Feature Detection . . . 44
  - 3.2.3 Outlier Rejection . . . 47

3.3 The Feature Point Tracking System 50
3.3.1 Overview 50
3.3.2 Efficient Feature Selection 51
3.3.3 Efficient Motion Estimation 53
3.3.4 Integrated Intensity Equalization 56
3.3.5 Hierarchical Translation Estimation 60
3.3.6 Revised Outlier Rejection 64
3.3.7 System Structure 65
3.4 High-Speed Feature Point Tracking 69
3.4.1 Motivation 69
3.4.2 Block Matching 71
3.4.3 The Adapted Tracking System 73
3.5 Summary 76

4 Structure and Motion Estimation **77**
4.1 Overview 78
4.1.1 Problem Statement 78
4.1.2 Related Work 82
4.2 State-of-the-Art Algorithms 85
4.2.1 Triangulation 86
4.2.2 Absolute Camera Pose Estimation 88
4.2.3 Relative Camera Pose Estimation 94
4.2.4 Bundle Adjustment 99
4.3 The Structure and Motion Estimation System 107
4.3.1 Overview 107
4.3.2 Key Frame Selection 108
4.3.3 Merging of Segments 114
4.3.4 Robust Estimation 116
4.3.5 System Structure 125
4.4 Summary 128

5 Point Set Registration **129**
5.1 Overview 129
5.1.1 Problem Statement 129
5.1.2 Related Work 133
5.2 The Iterative Closest Point Algorithm 134
5.2.1 Basic Principles 134
5.2.2 Existing Extensions 137
5.3 Robust Correspondence Estimation 138
5.4 Integrated Scale Estimation 142
5.5 Technical Specifications 144
5.6 Summary 145

6 Experimental Evaluation **147**
6.1 Feature Point Tracking . . . 147
6.1.1 Experimental Setup . . . 148
6.1.2 Evaluation of Individual Algorithms . . . 151
6.1.3 Evaluation of the Tracking System . . . 159
6.1.4 Summary . . . 162
6.2 Structure and Motion Estimation . . . 163
6.2.1 Experimental Setup . . . 163
6.2.2 Evaluation of the POSIT Algorithm . . . 168
6.2.3 Evaluation of the Estimation System . . . 170
6.2.4 Summary . . . 183
6.3 Point Set Registration . . . 184
6.3.1 Experimental Setup . . . 184
6.3.2 Evaluation of the Standard ICP Algorithm . . . 187
6.3.3 Evaluation of Robust Correspondence Estimation . . . 189
6.3.4 Evaluation of Integrated Scale Estimation . . . 194
6.3.5 Summary . . . 199
6.4 Joint Operation . . . 200
6.4.1 Experimental Setup . . . 200
6.4.2 Test Sequence Glass . . . 203
6.4.3 Test Sequence Globe . . . 206
6.4.4 Test Sequence Table . . . 209
6.4.5 Summary . . . 213

7 Applications **215**
7.1 Applications in the VAMPIRE project . . . 215
7.1.1 Hybrid Self-Localization . . . 215
7.1.2 Pictorial Scene Representation . . . 217
7.1.3 Data-Driven Object Tracking . . . 218
7.1.4 Model-Based Object Tracking . . . 219
7.2 Other Applications . . . 219
7.2.1 Online Structure and Motion Estimation . . . 219
7.2.2 Visualization with Image-Based Models . . . 220
7.2.3 Reconstruction from Fiberscopic Images . . . 221

8 Summary and Outlook **223**
8.1 Summary . . . 223
8.2 Outlook . . . 227

A German Translation **229**
A.1 Titel . . . 229
A.2 Kurzbeschreibung . . . 229

Bibliography **231**

Index **251**

List of Figures

1.1 The VAMPIRE augmented reality gear . . . 2
1.2 The VAMPIRE cognitive vision system . . . 3
1.3 The basic task of feature point tracking . . . 4
1.4 The basic task of structure and motion estimation . . . 4
1.5 The basic task of point set registration . . . 5

2.1 The basic image acquisition setup . . . 12
2.2 Discretization of a continuous image . . . 13
2.3 The four coordinate systems of image formation . . . 15
2.4 The perspective projection model . . . 16
2.5 Two affine projection models . . . 17
2.6 Effects of aperture size and focus setting . . . 20
2.7 Effects of radial distortion . . . 21
2.8 Correlation between vignetting and aperture size . . . 21
2.9 Reflection properties of two different surfaces . . . 24
2.10 Illustration of gamma correction . . . 25
2.11 Bayer color filter array . . . 26
2.12 Epipolar geometry . . . 28

3.1 Common challenges for feature point tracking . . . 34
3.2 Correspondence of 2-D features and 3-D features . . . 35
3.3 The basic principle of feature point tracking . . . 41
3.4 Interest images for different feature window sizes . . . 45
3.5 Feature selection for different feature window sizes . . . 46
3.6 Examples of affine motion estimation . . . 48
3.7 Hierarchical search structure for efficient feature selection . . . 52
3.8 Exemplary update of the hierarchical search structure . . . 52
3.9 The basic approach for efficient motion estimation . . . 55
3.10 Effects of automatic exposure correction . . . 57
3.11 Examples of integrated intensity equalization . . . 60
3.12 Gaussian image pyramids . . . 61
3.13 Extension of images with an additional border . . . 62
3.14 The robustified result propagation strategy . . . 63
3.15 The structure of our feature point tracking system . . . 66
3.16 CMOS camera and sample images . . . 69
3.17 Subwindow capturing problem . . . 74

3.18 Block matching with biquadratic interpolation 75
3.19 Illustration of the occurrence of similar feature windows 75

4.1 Three scenarios for structure and motion estimation 80
4.2 Triangulation of a feature point 86
4.3 The three-point algorithm 89
4.4 The basic approach of the POSIT algorithm 93
4.5 The five-point algorithm 95
4.6 Three segmentation strategies 107
4.7 The basic approach of our SaM estimation system 109
4.8 Triangulation accuracy for different view ray angles 110
4.9 The basic approach of our key frame selection algorithm 111
4.10 Visualization of two quality measures 112
4.11 Post-processing of key frames 113
4.12 The basic approach for aligning two segments 115
4.13 The success probability of random sampling 118
4.14 The basic approach of the LMedS technique 119
4.15 Illustration of the implemented M-estimators 123
4.16 The structure of our SaM estimation system 125

5.1 Different configurations for point set registration 130
5.2 Configurations that prevent successful point set registration 131
5.3 The structure of our variant of the ICP algorithm 144

6.1 Four test scenes for evaluation of motion estimation 149
6.2 Three test images with varying exposure settings 149
6.3 Illustration of selected feature points 150
6.4 Accuracy for images with identical exposure 151
6.5 Accuracy for images with differing exposure 152
6.6 Illustration of shape of basin of convergence 152
6.7 Basin of convergence for images with identical exposure 153
6.8 Basin of convergence for images with differing exposure 154
6.9 Basin of convergence of affine motion estimation 155
6.10 Basin of convergence of block matching 155
6.11 Basin of convergence for different filter sizes 156
6.12 Basin of convergence with respect to feature detection 157
6.13 Basin of convergence with respect to hierarchy levels 157
6.14 Basin of convergence of “gd tra” for different window sizes 158
6.15 Basin of convergence of “gd aff iie” for different window sizes 159
6.16 Illustration of the test image sequence 160
6.17 Illustration of tracked feature points 160
6.18 Illustration of three artificial test scenes 164
6.19 Illustration of test scene “slalom” 165

6.20 Illustration of test scene “spiral” 166
6.21 Feature point positions in first image of test scene “circle” 167
6.22 Feature point positions in first image of test scene “slalom” 167
6.23 Feature point positions in first image of test scene “wobble” 168
6.24 Three reconstructions of test scene “simple” 174
6.25 Exemplary reconstructions of test scene “slalom” 176
6.26 Exemplary reconstructions of test scene “spiral” 180
6.27 Exemplary reconstructions of test scene “wobble” 182
6.28 Illustration of input point set “bunny” 184
6.29 Illustration of input point set “bench” 185
6.30 Accuracy of ICP algorithm with respect to rotation 187
6.31 Accuracy of ICP algorithm with respect to translation 188
6.32 Basin of convergence with respect to rotation 188
6.33 Basin of convergence with respect to translation 189
6.34 Exemplary registration of input point set “bunny” 190
6.35 Accuracy of robust extensions with respect to rotation 190
6.36 Accuracy of robust extensions with respect to translation 191
6.37 Basin of convergence with respect to rotation 192
6.38 Basin of convergence with respect to overlap 192
6.39 Basin of convergence with respect to rotation 193
6.40 Basin of convergence with respect to translation 193
6.41 Basin of convergence with respect to overlap 194
6.42 Accuracy of integrated scale estimation 195
6.43 Basin of convergence of integrated scale estimation 195
6.44 Successful registration with scale estimation 196
6.45 Failed registration with scale estimation 196
6.46 Basin of convergence with respect to rotation 197
6.47 Basin of convergence with respect to translation 198
6.48 Illustration of test sequence “glass” 203
6.49 Subimages of test sequence “glass” 203
6.50 Feature point tracking in test sequence “glass” 204
6.51 Reconstruction of test sequence “glass” 205
6.52 Comparison of camera poses of test sequence “glass” 205
6.53 Illustration of test sequence “globe” 206
6.54 Subimages of test sequence “globe” 206
6.55 Feature point tracking in test sequence “globe” 207
6.56 Reconstruction of test sequence “globe” 208
6.57 Comparison of camera poses of test sequence “globe” 208
6.58 Illustration of test sequence “table” 209
6.59 Subimages of test sequence “table” 209
6.60 Feature point tracking in test sequence “table” 210
6.61 Reconstruction of test sequence “table” 211
6.62 Comparison of camera poses of test sequence “table” 212

7.1 Overview of the VAMPIRE augmented reality gear 216
7.2 Frame rates of high-speed feature point tracking 217
7.3 Feature point tracking for generating image mosaics 217
7.4 Feature point tracking for data-driven object tracking 218
7.5 Preprocessed fiberscopic images 221

List of Tables

3.1 The parameters of our feature point tracking system 67

4.1 The main equation system of the five-point algorithm 97
4.2 Overview of applications of the LMedS technique 120
4.3 The parameters of our structure and motion estimation system 126

5.1 The parameters of our variant of the ICP algorithm 145

6.1 Overview of all motion estimation algorithms 148
6.2 Experimental results of evaluation of tracking system 161
6.3 Experimental results of evaluation of POSIT algorithms 169
6.4 Configuration code for structure and motion estimation 170
6.5 Results for scene “simple” w. r. t. number of feature points 172
6.6 Results for scene “simple” w. r. t. number of views 173
6.7 Results for scene “simple” w. r. t. scene planarity 174
6.8 Results for scene “slalom” with respect to segmentation 176
6.9 Results for scene “slalom” with respect to bundle adjustment 177
6.10 Results for scene “slalom” with respect to M-estimators 178
6.11 Results for scene “spiral” with respect to bundle adjustment 179
6.12 Results for scene “spiral” with respect to M-estimators 179
6.13 Results for scene “wobble” with respect to segmentation 181
6.14 Results for scene “wobble” with difficult scene configurations 182
6.15 Configuration code for point set registration 186
6.16 Computational efficiency of our variant of the ICP algorithm 198
6.17 Parameter configurations of our tracking system 201
6.18 Parameter configurations of our SaM estimation system 202
6.19 Results for test sequence “glass” 204
6.20 Results for test sequence “globe” 207
6.21 Results for test sequence “table” 211
6.22 Analysis of ground truth data 212

Chapter 1

Introduction

The main goal of this thesis is to develop, enhance, and evaluate algorithms for sparse 3-D reconstruction within the scope of the VAMPIRE project [Sag11]. The first section of this chapter shortly describes the VAMPIRE project in general, as well as the purpose of sparse 3-D reconstruction in this project. Further sections of this chapter present related work, the contribution of this thesis to the state-of-the-art, and the outline of the remaining chapters.

1.1 Sparse 3-D Reconstruction in VAMPIRE

In recent years, computer vision systems have been successfully deployed in a variety of application areas, which range from automatic quality control in manufacturing to image enhancement in medical science. However, the application area of these computer vision systems is usually very constrained, which makes it possible to separate and decouple their processes for learning and recognition. In contrast to this, the superior performance of the human visual system is achieved by tightly coupling learning and recognition, and by exploiting contextual information and background knowledge. Therefore, humans are often capable of categorizing previously unknown objects and actions.

The aim of the VAMPIRE (Visual Active Memory Processes and Interactive Retrieval) project has been to integrate this ability into a computer vision system, and thus build a cognitive vision system, which acquires, maintains, and delivers selective knowledge. In order to achieve this goal, the visual active memory stores the visual history of the dynamic world over an extended period of time. The cognitive capabilities of the system are realized as processes that operate directly on the memory, which provides the necessary contextual information and background knowledge. Finally, information acquired by the cognitive vision system can be interactively retrieved with external user queries.

For a meaningful evaluation of the VAMPIRE cognitive vision system, the developed components have been combined into a fully integrated demonstrator. This demonstrator has been designed to work in an office scenario. In a simple use case, the user first places his coffee cup on the table. This action is observed and recognized by the system. When the user later looks for his cup, which has been covered

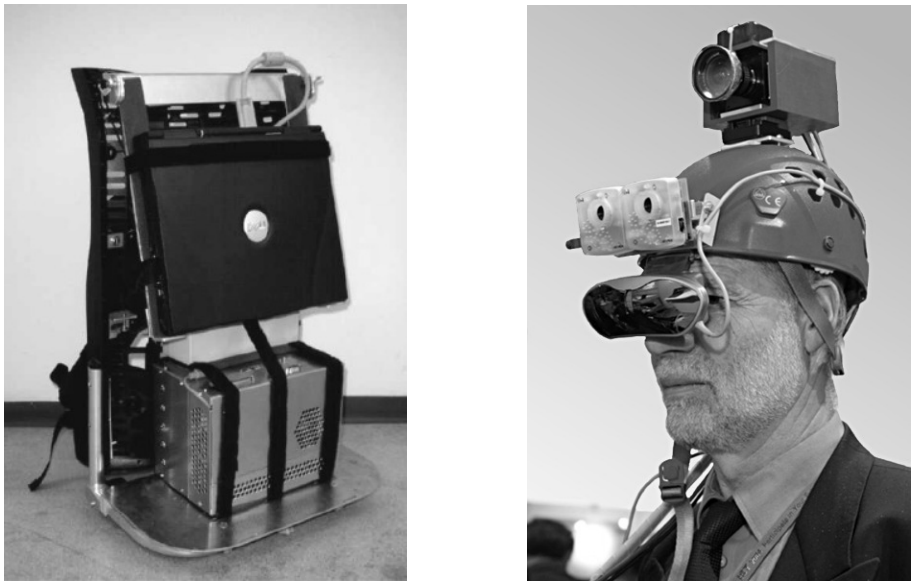


Figure 1.1: The VAMPIRE augmented reality gear consists of a backpack and a helmet. The left image shows the backpack, which houses a notebook, a custom-built single board computer, and a battery pack. The right image depicts the helmet, which carries three digital cameras and the video goggles.

with a newspaper in the meantime, he can query the system for the last known position of the cup.

The most prominent part of the demonstrator is the augmented reality gear, which is shown in Figure 1.1. It is used for acquiring data with the attached sensors. In our example use case, the coffee cup and the action of putting the cup on the table are both observed by the helmet-mounted cameras. In addition to that, the augmented reality gear also serves as the primary user interface. Consequently, it has to accept queries from the user and output the query results. For example, the position of the hidden cup can be visualized in the stereo goggles by arrows pointing in its direction or by superimposing an image of the cup at its current position.

Together with the augmented reality gear, additional software and hardware components form the VAMPIRE cognitive vision system, which is illustrated in Figure 1.2. The visual active memory is structured into four different levels. On the lowest level, image sequences are stored for further processing, whereas higher levels of the memory store more abstract data. The data processing components of the system are also organized into different levels, which correspond to the levels of the visual active memory. Apart from the shown components, there are additional memory processes for contributing important abilities like deleting data that is no longer needed.

With the help of the system structure shown in Figure 1.2, we can examine the role of sparse 3-D reconstruction in the VAMPIRE cognitive vision system. One very important component of this system deals with localization, which includes

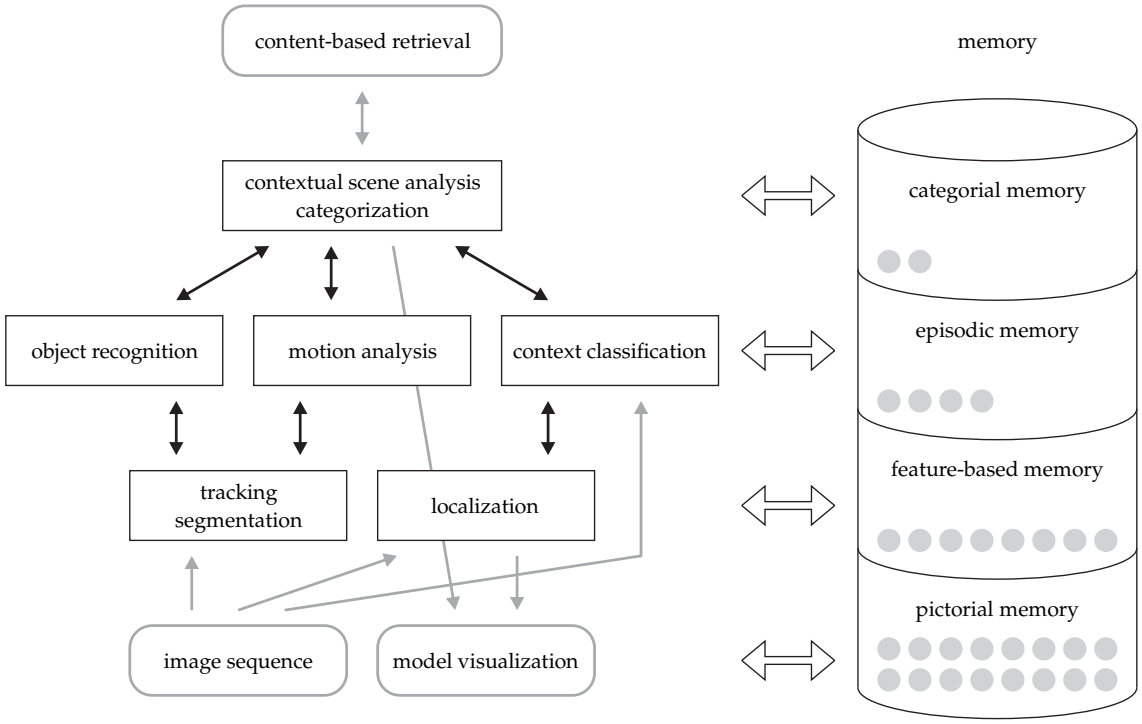


Figure 1.2: The VAMPIRE cognitive vision system consists of several interdependent components. In this figure, standard boxes represent internal system components, whereas boxes with rounded corners contain components for data input or output.

the localization of objects as well as the self-localization of the augmented reality gear. Especially for the self-localization, computing 3-D information about the environment of the user and matching this information with the video images in real-time is a vital task. Sparse 3-D reconstruction is also required for building 3-D models, which are useful for object recognition, object localization, model-based object tracking, and object revisualization. Therefore, sparse 3-D reconstruction is an important part of several components of the VAMPIRE cognitive vision system.

In this thesis, the term 3-D reconstruction subsumes all aspects of the reconstruction and the processing of the scene structure and the camera motion of an input image sequence. In particular, sparse 3-D reconstruction recovers the scene structure only for a limited number of feature points. In contrast to this, approaches that result in a dense sampling of the scene structure, for example in the form of range images, are not considered in this thesis. We categorize the algorithms for sparse 3-D reconstruction discussed in this thesis into three main work areas. In the following, we will shortly introduce these work areas and describe their relevance for sparse 3-D reconstruction in VAMPIRE.

The algorithms of the first work area are generally referred to as feature point tracking algorithms. As illustrated in Figure 1.3, their input data consists of a sequence of video images. Initially, the algorithms have to select a specified number

Figure 1.3: The basic task of feature point tracking is to estimate the motion of feature points in the images of a video sequence.

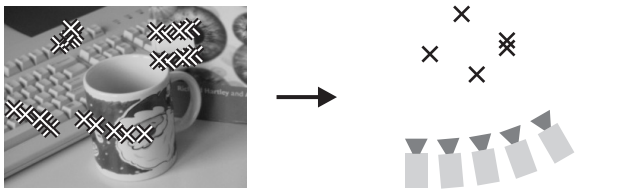


Figure 1.4: The basic task of structure and motion estimation is to recover the structure of the scene and the motion of the camera from the estimated feature trails.

of suitable feature windows, which ideally correspond to a fixed 3-D point in the captured scene. Then, the 2-D position of the feature windows is estimated in consecutive images of the video sequence. Accordingly, the output of a feature point tracking algorithm is a sequence of 2-D coordinates for each feature window. These sequences are also called feature trails. The particular importance of feature point tracking algorithms for the VAMPIRE system derives from the high number of algorithms that depend on feature trails as input data. The most prominent application of feature point tracking is the self-localization of the augmented reality gear. Without knowledge of the current position and orientation of the gear, the example use case of finding a hidden coffee cup would not be feasible. There are additional components that rely on feature point tracking, like an approach for building image mosaics of planar subscenes and several different approaches for object tracking.

The basic task of the algorithms of the second work area is illustrated in Figure 1.4. These algorithms use the feature trails generated by the feature point tracking algorithms as input data. Their output consists of the 3-D coordinates of the feature points and the position and orientation of the camera for every captured image. As the structure of the scene is deduced with the help of the camera motion, which in turn causes the feature motion in the image sequence, the problem at hand is also known as structure from motion [Oli00]. In order to emphasize the recovery of both scene structure and camera motion, we refer to this work area as structure and motion estimation. Possible applications for the structure and motion estimation algorithms in VAMPIRE include computing a 3-D map of the current scene

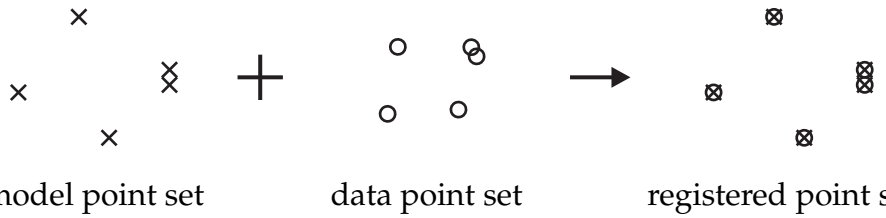


Figure 1.5: The basic task of point set registration is to align two point sets that represent the same scene or object.

and building 3-D models for object tracking and revisualization. In our example use case, the position of the coffee cup can be determined by tracking it with an object tracker when the user places it on the table. At a later date, a revisualization of the coffee cup in the stereo goggles can be used to report its current position to the user.

Finally, the algorithms of the third work area are responsible for merging different reconstructions of the same scene or object. As the reconstructions are represented by 3-D point sets in this thesis, this work area is also known as point set registration. Its basic principle is illustrated in Figure 1.5. Reconstructions computed by structure and motion estimation algorithms are only known up to a global scale factor. Therefore, we give special consideration to the registration of differently scaled reconstructions. With two 3-D point sets as input data, the registration comprises the estimation of the rotation, the translation, and the scale factor required to optimally align the two point sets.

1.2 Related Work

As cognitive vision systems have attracted considerable interest within the computer vision community, the VAMPIRE project is not the only research effort in this field. In order to give an overview of the work related to the VAMPIRE project, we present and compare other projects with a focus on cognitive vision in this section. In addition to that, we also examine the role of sparse 3-D reconstruction algorithms in these projects. Other work specifically related to one of the three main work areas of this thesis is described in the respective chapters.

The PLAYBOT project is one of the first research projects to examine cognitive vision from a systems perspective [Tso98]. Its main goal has been to develop a directable robot that enables physically disabled children to access and manipulate toys. Important goals of the PLAYBOT project are the use of vision as the primary sensor, the capability of visual search and recognition of objects, and the ability to safely work in a real and unpredictable environment. Beside their use of vision as the main input modality, both PLAYBOT and VAMPIRE share the strong interest in a technically mature and enjoyable user interface. Both projects also rely on

sophisticated object recognition algorithms. However, the VAMPIRE system additionally tries to learn new objects and categories from the contextual information in the scene. The PLAYBOT system completely avoids the use of 3-D reconstruction by making suitable adaptions to the working environment, like the use of objects and furniture with contrasting colors.

The CogVis (Cognitive Vision) project researches techniques for the construction of cognitive vision systems that perform recognition and categorization of objects and events in the context of a task-oriented mobile visual agent [Chr11]. One important assumption of the CogVis project is that a number of implicit and explicit visual tasks form the basis for visual cognition, e. g., identification, recognition, description, and detection. Like the VAMPIRE project, the CogVis project emphasizes the fundamental role of memory for the coupling of perception, control, and knowledge in a cognitive vision system. Both projects also attach great importance to the learning capabilities of their cognitive vision system in a natural environment. Due to the complexity of building a complete cognitive vision system, the CogVis project relies on several subsystems that focus on different aspects of the complete system. In contrast to this, the integrated demonstrator of the VAMPIRE project combines all developed components into a single system. The CogVis project does not require accurate 3-D information and uses appearance-based methods for low-level vision.

The CogViSys (Cognitive Vision Systems) project concentrates on the development of a virtual commentator that transforms visual information from the domains of traffic surveillance, sign language understanding, and video annotation into a textual description [Nag04]. The proposed applications of the CogViSys project do not require interaction with a human user. Therefore, issues like task-oriented processing strategies or response time limitations are not addressed. As a consequence of using static cameras as vision sensors, the CogViSys project does not perform 3-D reconstruction.

The emphasis of the LAVA (Learning for Adaptable Visual Assistants) project is on the robust and efficient categorization and interpretation of large numbers of objects, scenes, and events in real settings [Csu11]. There are two integrated demonstrators for the evaluation of the project, the Association Assistant for analyzing single digital images, and the Event Interpreter for dynamic scene understanding. Both demonstrators rely on novel appearance-based approaches for low-level vision and do not consider 3-D information of the scene.

The CHIL (Computers in the Interaction Loop) project expands the scope of the previous projects by additionally considering multi-modal input [Wai04]. Its purpose is to create environments in which computers serve humans, who focus on interacting with other humans instead of having to concentrate on the handling of the computers. The CHIL project researches computer vision algorithms for, e. g., person tracking, gesture recognition, and activity analysis, which do not depend on 3-D reconstruction.

The presented cognitive vision projects exhibit two contrasting approaches for the role of 3-D reconstruction. On the one hand, many projects completely avoid

its use. Although the theory of 3-D reconstruction from image sequences is already well understood, the development of efficient and robust algorithms that work in a wide range of environments is still a very difficult problem. Therefore, many cognitive vision projects prefer to work with appearance-based approaches. On the other hand, the VAMPIRE project uses sparse 3-D reconstruction in several components in order to provide the system with accurate 3-D data. In the VAMPIRE demonstrator, this approach enables the estimation of the position and orientation of the user's head, which yields information required for the localization of hidden objects. Furthermore, sparse 3-D reconstruction is also used for the automatic generation of object models, which facilitate the use of model-based approaches for object tracking.

1.3 Contribution of this Work

The main aim of this work is to develop and enhance algorithms for sparse 3-D reconstruction that can successfully be employed in the low-level components of the VAMPIRE cognitive vision system. Thus, the requirements for these algorithms can be directly derived from the components that either integrate them or use their results. There are two requirements that are relevant for all of the developed algorithms. First, the algorithms have to be efficient. This property is of major concern in the components of the augmented reality gear, because the visual lag between an action of the user and the reaction of the system must be as small as possible in order to ensure good usability. But efficiency is also important for offline computations in the visual active memory, where many tasks have to be performed with a limited amount of computational resources.

Second, the algorithms have to be as robust as possible. In the office scenario of the VAMPIRE project, neither the environment nor the user can be fully controlled. Therefore, the algorithms have to successfully cope with unexpected and challenging conditions. This is especially true for the low-level algorithms for sparse 3-D reconstruction, because other algorithms rely on their output as input data. As a consequence of the requirements imposed by the VAMPIRE system, we have focussed our contributions on improving the efficiency and robustness of the algorithms for sparse 3-D reconstruction from image sequences. In the following, these contributions are separately summarized for each of the three main work areas.

Feature point tracking is clearly the most active research area of this thesis. A standard Kanade-Lucas-Tomasi feature point tracker forms the starting point for our work [Luc81, Tom91]. In order to improve the efficiency of feature selection, we developed a hierarchical feature selection algorithm. The translation estimation is performed with a robustified strategy on a Gaussian image pyramid, which enhances the tolerance for large frame-to-frame movements. The combination of the inverse compositional approach for gradient descent motion estimation [Bak04] with the intensity equalization described by [Jin01] ensures both high efficiency and

high robustness. The previous enhancement also enables efficient affine motion estimation in every frame, which is used to implement a feature drift prevention scheme that is especially useful for long image sequences [Zin04].

Integrated into our feature point tracking system, these enhancements make it possible to track 300 features points at 30 frames per second on a standard personal computer. This computation speed is sufficient for applications like building image mosaics or object tracking. However, the self-localization of the augmented reality gear has different requirements. In this application, the number of tracked features is small, but the frame rate supported by the custom-built camera is very high [Mue04]. By replacing the standard translation estimation algorithm with an efficient block matching algorithm, we are able to achieve frame rates of more than 200 frames per second. We proposed this approach in [Zin05a].

Comprehensive overviews of structure from motion estimation are given in both [Har00] and [Oli00]. Although the theory of structure from motion is already well understood, developing an efficient and robust approach that works reliably and without user interaction in an unconstrained environment is still a very difficult problem. We advance the state-of-the-art of this work area with a structure and motion estimation system that is based on the approach proposed by Nister in [Nis04b]. The main design philosophy of our system is to combine efficient algorithms for the initial reconstruction with accurate algorithms for the optimization of the results. This approach yields a highly versatile system, which meets the requirements of all applications in the VAMPIRE project.

Our structure and motion estimation system comprises existing algorithms like the five-point algorithm for relative camera pose estimation described in Subsection 4.2.3, extended algorithms like the POSIT algorithm for absolute camera pose estimation using a virtual reference point proposed in Subsection 4.2.2, and new algorithms like our key frame selection algorithm detailed in Subsection 4.3.2. The versatility of our system is improved by making it possible to balance conflicting performance criteria like efficiency, accuracy, and robustness with user-defined parameters. Finally, all components of our structure and motion estimation system employ state-of-the-art techniques for robust estimation to ensure a high level of robustness to outliers.

The most widely used algorithm for 3-D point set registration is the ICP algorithm presented by Besl and McKay [Bes92]. A concise overview of different variants of this algorithm can be found in [Rus01]. In this work, we compare three different techniques for increasing the robustness of the ICP algorithm to outliers. We introduced one of these techniques in [Zin03a]. In addition to that, we propose an integrated estimation of the scale factor between the two point sets in Section 5.4. We also presented this approach in [Zin05b]. Our contribution is a drop-in replacement for the standard motion estimation, which makes it viable as an upgrade for a large number of different ICP variants. Differently scaled point sets occur when at least one of them is computed with a structure and motion algorithm, because these reconstructions are only unique up to a scale factor.

The quality of the contributions of this thesis is mirrored by the widespread adoption of the developed algorithms. For example, the feature point tracking algorithms are not only used by three different partners of the VAMPIRE project, but also by other research groups. Further information on applications of the developed algorithms can be found in Chapter 7.

1.4 Outline

This section details the structure of the remaining parts of this thesis. In order to convey a better understanding of the basic input modality of computer vision algorithms, Chapter 2 explains the basic principles of image formation. The covered topics include geometric image formation, radiometric image formation, and multiple view relations.

As the algorithms for sparse 3-D reconstruction are categorized into three main work areas in this thesis, each work area is presented in a chapter of its own. Chapter 3 covers the algorithms for feature point tracking. After describing the standard Lucas-Tomasi-Kanade feature point tracker, we detail our enhancements for efficient and robust tracking. In Chapter 4, we present our work on structure and motion estimation. The description of important state-of-the-art algorithms is followed by a detailed analysis of the enhancements in our proposed structure and motion estimation system. Chapter 5 is concerned with the algorithms for 3-D point set registration. It contains sections on the standard iterative closest point algorithm, robust correspondence estimation, and our proposed integrated scale estimation.

The experimental evaluation of the proposed algorithms is documented in Chapter 6. All proposed algorithms are separately evaluated with artificial, but realistic, data. In addition to that, the algorithms are also jointly evaluated with real input data with known ground truth in Section 6.4. For all evaluations, we also provide a thorough analysis of the obtained results. Chapter 7 is a collection of applications of the algorithms proposed in this thesis. For each application, there is a description of the problem and an explanation of how the respective algorithm is applied in its solution. Finally, a summary and an outlook conclude this thesis in Chapter 8.

Chapter 2

Basic Principles of Image Formation

Computer vision in general can be described as the automatic extraction of information from images. This description is certainly valid for the feature point tracking algorithms covered in the next chapter, because they operate directly on image sequences. But also the algorithms for structure and motion estimation, which process the output of the feature point tracking algorithms, indirectly extract information from image sequences. As most algorithms in this work directly or indirectly operate on images to extract information about the 3-D world, it is important to understand the process of creating these images. Consequently, we explain the basic principles of image formation in this chapter. In order to keep these explanations short, we focus on the aspects of image formation that are relevant for the algorithms in the following chapters. We also point out where model assumptions are not met by reality.

This chapter starts with a general description of image acquisition and its result, the digital image. The second section of this chapter presents geometric image formation, which mainly describes the interrelation between the position of an object in the 3-D world and its position in the digital image. In the section about radiometric image formation, we discuss the formation and representation of intensity and color values in a digital image. In the next section, we cover the geometric relations between two images taken from different viewpoints. The last section of this chapter contains a short summary.

2.1 Overview

2.1.1 Image Acquisition Fundamentals

The most important aspects of image acquisition can be explained with the help of the basic image acquisition setup shown in Figure 2.1. The setup contains one or more illumination sources, which are also called light sources. We are only interested in illumination sources that emanate visible light, which consists of electromagnetic waves with a wavelength between approximately 0.4 and 0.8 micrometers. As the notion of visible light is solely determined by the human visual system,

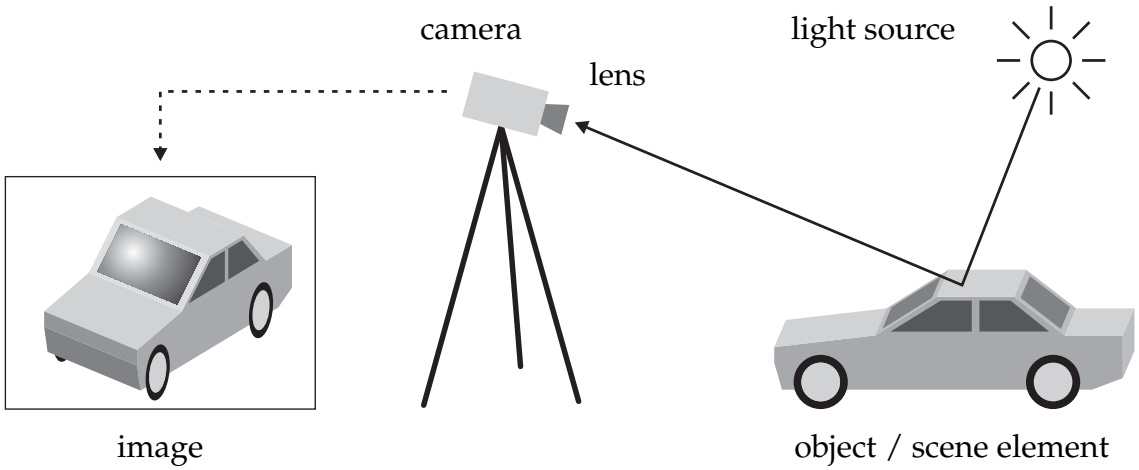


Figure 2.1: The basic image acquisition setup

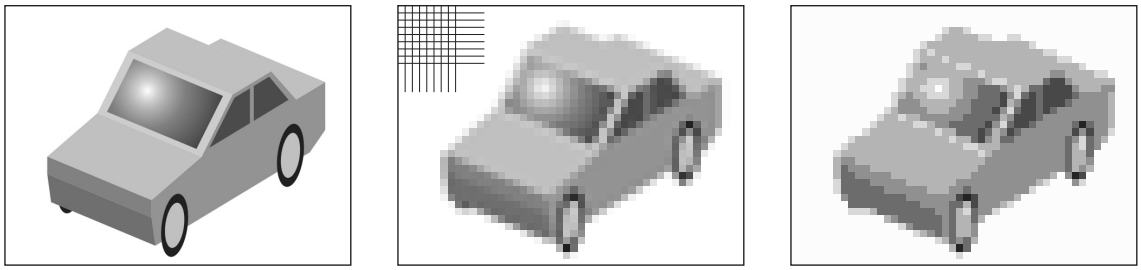
it is not surprising that there are also biological and electronic sensors for different portions of the electromagnetic spectrum.

After the light has left the illumination source, it travels in an approximately straight line, until it hits a scene element and is absorbed or reflected. The reflection properties of the scene elements determine which fraction of the incoming light is reflected in any possible direction. All scene elements together form the complete scene. Some of the light reflected by the scene elements enters the imaging device used for image acquisition. In our case, the imaging device is a camera, which consists of a lens used for gathering light from the scene, and a camera body housing an electronic sensor, which transforms the incoming light into electronic signals. A more elaborate discussion of the geometric and radiometric properties of the objects that are part of the image acquisition setup is presented in the two following sections. Before that, however, we have a look at the result of a successful image acquisition process, the digital image itself.

2.1.2 The Digital Image

Inside a standard computer, any information is stored in digital form, which means that it is represented by a sequence of bits, each of which can either be zero or one. In contrast to this digital form of information, the left image in Figure 2.2 is a continuous image of the scene corresponding to the light rays entering the camera. It can be described with a continuous function, which is defined by the amplitude of the image signal for any pair of image coordinates. In order to transform this image into a digital image, two discretization steps have to be performed. Typically, both discretization steps take place inside the digital camera.

The first discretization step consists of discretizing the coordinate values. To this end, the image is sampled at equally spaced coordinate positions. This process is



original image | image after sampling | image after quantization

Figure 2.2: This figure illustrates the effects of the discretization of a continuous image by sampling and quantization. The original image is sampled at 48×36 coordinate positions in the middle image, which contains a small section of the sampling grid in its upper left corner.

also called image sampling. Another way to describe the effects of image sampling is to say that the amplitude information for all other image coordinates is discarded. Image sampling is one of the key factors for determining the spatial resolution of an image, which is basically the smallest observable detail in an image. In accordance with [Gon01], we refer to the width and height of an image in sampled coordinate positions as the resolution of the image. The effects of image sampling are illustrated in the middle image of Figure 2.2, which has a resolution of 48×36 sampled coordinate positions.

In the second discretization step, which is also called quantization, the amplitude of the image signal is discretized. In the case of gray-level images, the amplitudes are also called intensities. Due to the storage of digital images in computer memory, the number of gray levels in an image is usually a power of two. The most common number is 256 gray levels, which can be efficiently stored in eights bits, or one byte. In order to show no visible artifacts, an image has to contain a minimum number of approximately 32 gray levels [Gon01]. In the right image of Figure 2.2, the amplitudes of the original image have been discretized into eight intensity values.

The image acquisition process implicitly contains a third kind of discretization. The scene is not captured by the camera in a continuous way, but only at discrete time steps. This behavior seems natural when only a single image is captured. Image sequences, however, are usually used to capture some kind of motion in the scene. In this case, the time interval between capturing two consecutive images is directly proportional to the amount of motion measured in image coordinates. For algorithms that search for correspondences in consecutive images, it is beneficial when the motion between these images is limited. When speaking of image sequences or video sequences, we refer to a single image as a frame. Consequently, the frequency of image captures in a video sequence is measured in frames per second.

A digital gray-level image with a resolution of $W \times H$ and L gray levels can be represented by a discrete two-dimensional function

$$\begin{aligned} f(\boldsymbol{x}) \in \{0, \ldots, L-1\}, \quad \boldsymbol{x} = (x, y)^{\mathrm{T}}, \\ x \in \{0, \ldots, W-1\}, \\ y \in \{0, \ldots, H-1\}. \end{aligned} \tag{2.1}$$

By convention, the intensity value zero denotes black, and the maximum intensity value denotes white. Furthermore, the coordinate pair $(0, 0)$ is located in the upper left corner of the image. A digital image can alternatively be represented by a matrix of intensity values. Each element of the matrix corresponds to a valid coordinate pair of the function $f(\boldsymbol{x})$. The elements of the matrix are also called image elements, picture elements, or pixels.

2.2 Geometric Image Formation

As already mentioned in the introduction of this chapter, geometric image formation determines the interrelation between the position of an object in the 3-D world and its position in the digital image. Knowledge about this interrelation is most important for understanding the principles of structure and motion estimation, which is basically the inversion of the geometric image formation process. Nevertheless, the properties of geometric image formation also influence the design of feature point tracking and 3-D registration algorithms. Consequently, the material presented in this section is relevant for all three main work areas of this thesis.

2.2.1 Coordinate Systems

For the mathematical description of the image formation process, the positions of objects, feature points, and pixels are expressed as coordinate vectors in a coordinate system. In order to efficiently describe different aspects of the image formation process, several different coordinate systems are used. The four relevant coordinate systems for our work are illustrated in Figure 2.3. The world coordinate system and the camera coordinate system are both right-handed, three-dimensional coordinate systems. The image coordinate system and the sensor coordinate system are two-dimensional coordinate systems. The coordinates of the sensor coordinate system are typically measured in pixel units.

The world coordinate system is used to specify the position of scene elements and of one or more cameras in the 3-D world. In general, it can be chosen arbitrarily. Consequently, it is possible that the world coordinate system and the camera coordinate system coincide. If this is not the case, a point $\boldsymbol{p}$ in world coordinates can be transformed into a point $\boldsymbol{c}$ in camera coordinates by applying a 3-D rotation

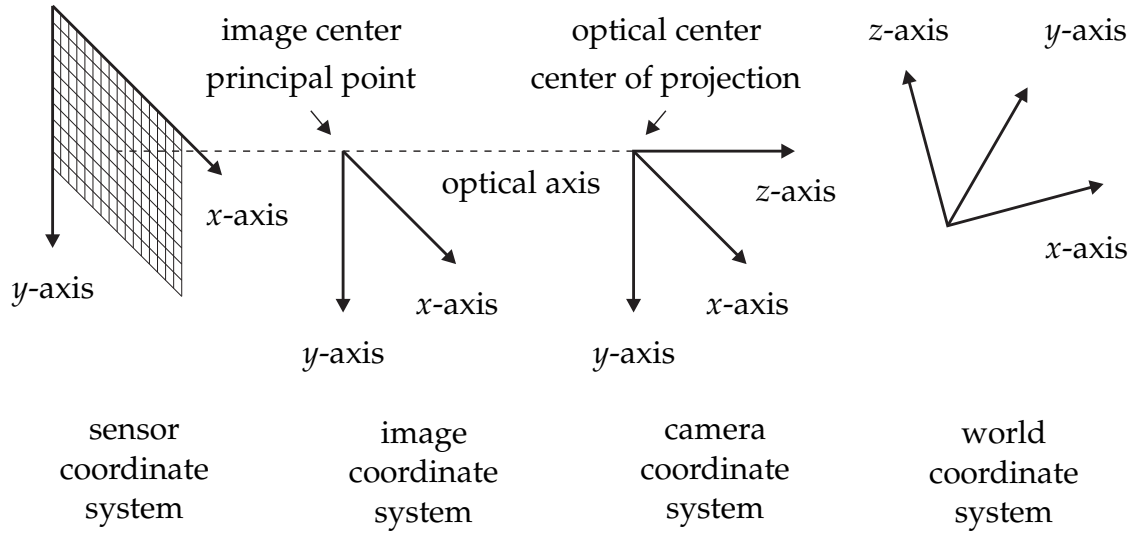


Figure 2.3: The four coordinate systems of the image formation process

matrix $\boldsymbol{R}$ and a translation vector $\boldsymbol{t}$ as follows

$$\boldsymbol{c} = \boldsymbol{R}\boldsymbol{p} + \boldsymbol{t}, \quad \boldsymbol{c}, \boldsymbol{p}, \boldsymbol{t} \in \mathbb{R}^3, \quad \boldsymbol{R} \in \mathbb{R}^{3\times 3}. \tag{2.2}$$

In this notation, the columns of the matrix $\boldsymbol{R}$ correspond to the coordinate axes of the world coordinate system in camera coordinates. Analogously, the translation vector $\boldsymbol{t}$ can be interpreted as the origin of the world coordinate system in camera coordinates.

The origin of the camera coordinate system is the optical center or center of projection, whereas the origin of the image coordinate system is the image center or principal point. The two coordinate axes of the image coordinate system point into the same direction as the first two coordinate axes of the camera coordinate system. We discuss the relative position of these coordinate systems and the transformation between them in the next subsection. The intensity values of the final image are sampled at discrete coordinates of the sensor coordinate system. The transformation between the image coordinate system and the sensor coordinate system is discussed in Subsection 2.2.3.

It is important to know that the name, orientation, and description of the presented coordinate systems vary in the computer vision literature. In one publication, our sensor coordinate system is called image coordinate system, whereas our image coordinate system is referred to as the projected camera coordinate system [Tön05]. In another publication, the term retinal coordinate system is used instead of sensor coordinate system [For03].

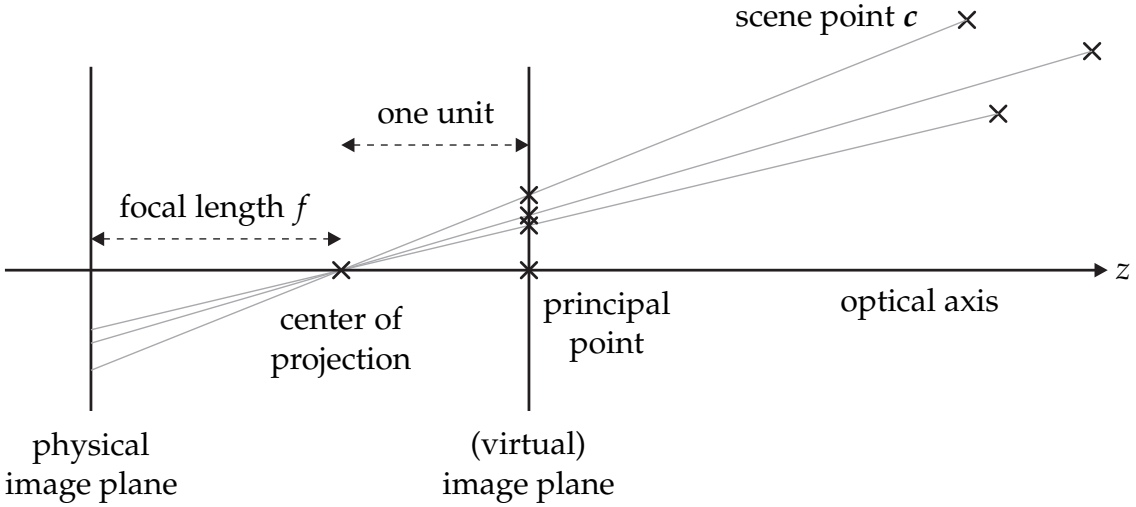


Figure 2.4: The perspective projection model

2.2.2 Projection Models

Projection models are a mathematical description of the coordinate transformation from the camera coordinate system to the image coordinate system. In the following, we cover the three projection models that are relevant for our work. The perspective projection model describes the geometry of a pinhole camera. All light rays entering a pinhole camera must pass through the pinhole, which ensures that each point on the image plane receives light from exactly one direction. By tracing a light ray from a 2-D image point through the pinhole into the scene, it is possible to determine the 3-D point corresponding to it. Both the weak-perspective projection model and the paraperspective projection model are approximations to the perspective projection model.

The perspective projection model is illustrated in Figure 2.4. In this model, the pinhole is the center of projection, which is also the origin of the camera coordinate system. In a real pinhole camera, the light rays passing through the pinhole are recorded at the physical image plane. In our opinion, it is easier to discuss projection properties in terms of a virtual image plane, which is parallel to the physical image plane, but lies on the opposite side of the center of projection. Consequently, we refer to the virtual image plane as image plane.

The optical axis is a line that is perpendicular to the image plane and goes through the center of projection. The intersection point between the optical axis and the image plane is the principal point. As we know from the previous subsection, it is also the origin of the image coordinate system. In order to simplify the mathematical notation, we define the distance between the center of projection and the principal point to be one unit. With these definitions, we can now write the basic equations for projecting a point $\boldsymbol{c} = (c_x, c_y, c_z)^{\mathrm{T}}$ in camera coordinates to a point $\boldsymbol{q} = (q_x, q_y)^{\mathrm{T}}$

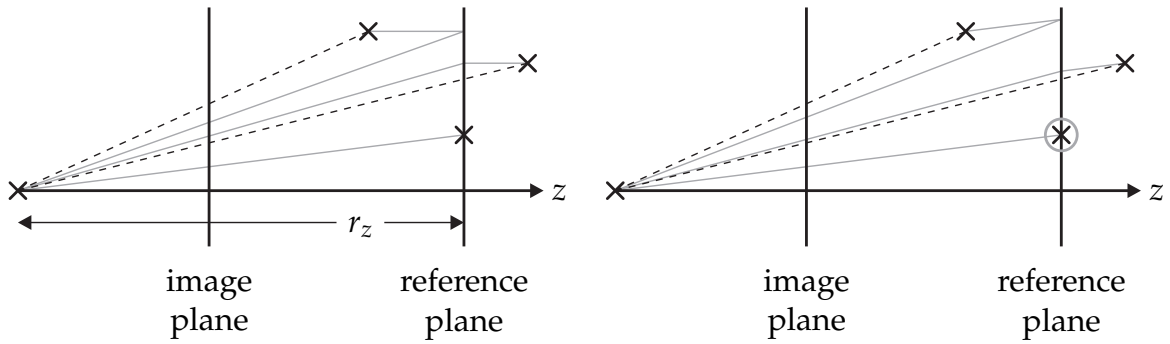


Figure 2.5: Both the weak-perspective projection model in the left illustration and the paraperspective projection model in the right illustration are affine projection models that approximate the perspective projection model. The reference point of the paraperspective projection model is marked by a circle.

in image coordinates as

$$q_x = \frac{c_x}{c_z}, \quad q_y = \frac{c_y}{c_z}. \tag{2.3}$$

The weak-perspective projection model is shown in the left illustration of Figure 2.5. Each point in camera coordinates is first projected onto a reference plane along a parallel to the optical axis. Then, the resulting points are all scaled by a constant factor derived from the perspective projection model. As a consequence, weak-perspective projection is also referred to as scaled orthographic projection. The equations of the weak-perspective projection model are

$$q_x = \frac{c_x}{r_z}, \quad q_y = \frac{c_y}{r_z}, \tag{2.4}$$

where r_z is the distance of the reference plane to the center of projection. In contrast to the equations in (2.3), the projection equations of the weak-perspective projection model are linear in the point coordinates, because r_z is a constant.

For the paraperspective projection model, which is illustrated in the right image of Figure 2.5, the orthographic projection of the weak-perspective projection model is replaced. After the selection of a reference point $(r_x, r_y, r_z)^{\mathrm{T}}$, the points in camera coordinates are projected onto the reference plane along a parallel to the line from this reference point to the center of projection. The equations for the paraperspective projection model are

$$q_x = \left(\frac{c_x}{r_z} + \frac{r_x}{r_z}\left(1 - \frac{c_z}{r_z}\right)\right), \quad q_y = \left(\frac{c_y}{r_z} + \frac{r_y}{r_z}\left(1 - \frac{c_z}{r_z}\right)\right). \tag{2.5}$$

Both the weak-perspective and the paraperspective projection are approximations to the perspective projection. As they do not model the effect that objects

which are further away appear to be smaller, they work best when the average distance of the points from the center of projection is much larger than the distance of any two points along the optical axis. Furthermore, the approximation errors are minimized when the reference plane (weak-perspective projection) or the reference point (paraperspective projection) lie close to the center of mass of all available points in camera coordinates. For more information about projection models in general, we recommend [Har00]. An elaborate discussion of the weak-perspective and the paraperspective projection can be found in [Chr96]. In this thesis, the weak perspective projection model is used by the POSIT algorithm described in Subsection 4.2.2.

2.2.3 Camera Parameters

In this subsection, we first describe the transformation between image coordinates and sensor coordinates, which is the last transformation in the geometric image formation process. Then, we summarize the parameters that are necessary to model the complete transformation from the world coordinate system to the sensor coordinate system. Finally, we show how to use homogeneous coordinates to formulate this transformation in a linear matrix equation.

The transformation between image coordinates and sensor coordinates consists of a translation and a scaling. The translation moves the origin of the sensor coordinate system to the upper left corner of the digital image. It is represented by the position $(O_x, O_y)^{\mathrm{T}}$ of the principal point in sensor coordinates. The scaling along the coordinate axes ensures that the sensor coordinates are expressed in pixel units. It depends on the focal length F, which is the distance between the center of projection and the physical image plane, and the relative sizes S_x and S_y of a pixel in the horizontal and vertical direction, respectively. With these parameters, we can represent the transformation between image coordinates and sensor coordinates as follows

$$x = \frac{F}{S_x}\, q_x + O_x\,, \quad y = \frac{F}{S_y}\, q_y + O_y\,. \tag{2.6}$$

As we have argued in this section, the geometric image formation process governs the transformation of coordinates from the three-dimensional world coordinate system to the two-dimensional sensor coordinate system. The complete transformation is usually described by two types of camera parameters, the extrinsic and the intrinsic camera parameters. Trucco and Verri define the extrinsic camera parameters as "any set of geometric parameters that identify uniquely the transformation between the unknown camera reference frame and [...] the world reference frame" [Tru98]. This definition is met by the rotation matrix $\boldsymbol{R}$ and the translation vector $\boldsymbol{t}$ in equation (2.2).

The intrinsic camera parameters are properties of the camera used for capturing an image. They describe the interrelation between camera coordinates and sen-

sor coordinates. The most important intrinsic camera parameters are the position $(O_x, O_y)^{\mathrm{T}}$ of the principal point, the relative sizes S_x and S_y of the pixels, and the focal length F. Another intrinsic camera parameter is the angle between the two axes of the sensor coordinate system, which is typically $\pi/2$ in a standard digital camera.

In order to formulate the transformation from the world coordinate system to the sensor coordinate system in a linear matrix equation, we have to work with homogeneous vectors in projective spaces. Basically, the projective space $\mathbb{P}^n$ is the union of the Euclidean space $\mathbb{R}^n$ and all n-dimensional points lying at infinity. An element of $\mathbb{P}^n$ can be described by the homogeneous vector $\underline{\boldsymbol{x}} = (x_1, \ldots, x_{n+1})$. If x_{n+1} is zero, the corresponding point $\boldsymbol{x}$ lies at infinity. Otherwise, $\underline{\boldsymbol{x}}$ corresponds to the Euclidean point $\boldsymbol{x} = (x_1/x_{n+1}, \ldots, x_n/x_{n+1})$. For a more thorough description of homogeneous coordinates, we recommend the books [Tru98] and [Har00]. Finally, we can combine the equations (2.2), (2.3), and (2.6) for the different coordinate system transformations into a linear matrix equation

$$\underline{\boldsymbol{x}} = \boldsymbol{K}\,\boldsymbol{M}\,\underline{\boldsymbol{p}} = \begin{pmatrix} F/S_x & 0 & O_x \\ 0 & F/S_y & O_y \\ 0 & 0 & 1 \end{pmatrix} (\boldsymbol{R} \mid \boldsymbol{t})\ \underline{\boldsymbol{p}}\,, \quad \underline{\boldsymbol{p}} = (\boldsymbol{p}, 1)^{\mathrm{T}}\,, \tag{2.7}$$

where the intrinsic and extrinsic camera parameters are represented by the matrices $\boldsymbol{K} \in \mathbb{R}^{3\times3}$ and $\boldsymbol{M} \in \mathbb{R}^{3\times4}$, respectively.

According to [Tru98], camera calibration is the process of determining the intrinsic camera parameters, the extrinsic camera parameters, or both. There are many algorithms for camera calibration, which differ both in the set of estimated parameters and the requirements for the input data. As a thorough review of camera calibration techniques is outside the scope of this thesis, we refer to [Fau93] and [Tru98] for further information. The algorithms for structure and motion estimation discussed in Chapter 4 require that the intrinsic parameters of the camera are known for every captured image. In contrast to this, the extrinsic camera parameters, i. e., the rotation and translation of the camera, are unknown and have to be estimated by the algorithms.

2.2.4 Real Lenses

In the previous subsections, we derived a mathematical model of the geometric image formation process, which finally resulted in the equation (2.7). In this subsection, we discuss where the model assumptions are not met by reality. On the one hand, this is an important step to assess the validity of the derived model. On the other hand, we use this approach to illustrate some of the less obvious difficulties in computer vision and to emphasize the need for robust algorithms, which are presented in the following chapters of this thesis.

The projection model derived from the pinhole camera is the main shortcoming of the preceding description of geometric image formation. Although there are real

(a) (b) (c) (d)

Figure 2.6: The small depth of field in images (a)-(c) is caused by using a wide open aperture. Due to different focus settings, different parts of the scene are in sharp focus. In image (d), the depth of field has been increased by reducing the size of the aperture.

pinhole cameras, the optical system of standard digital camera usually consists of a sophisticated combination of several differently shaped lenses. This setup allows the camera sensor to gather more light in a shorter amount of time, which makes taking pictures of moving objects possible in the first place. In addition to that, some modern compound lenses allow to change the focal length of the optical system by moving some of their internal lenses, which is also known as the ability to zoom.

Mathematical descriptions of more sophisticated optical systems than the pinhole, e. g., the thin lens model and the thick lens model, explain some of the problems caused by real lenses [For03]. The thin lens model introduces the possibility that objects may be out of focus. Only objects within a certain range of distances, which is also called depth of field, are in acceptable focus. Objects outside this range become blurred, because the light rays emanating from one point of the scene hit different points on the image plane. Modern compound lenses allow to adapt the focus setting, which can be used to keep the important parts of a scene in sharp focus. The first three images of Figure 2.6 illustrate different focus settings in the case of a small depth of field. Additionally, the depth of field of a lens can be increased by reducing the size of the aperture of the lens and vice versa. This effect can be observed by comparing image (b) and image (d) of Figure 2.6.

The thick lens model adds explanations for other causes of image degradation. There are four types of primary aberrations, which degrade the image by blurring objects. They are called astigmatism, coma, field curvature, and spherical aberration. As high quality lenses are optimized to avoid these kind of aberrations, their effects can safely be ignored in standard computer vision applications. The fifth primary aberration is distortion, which changes the shape of the image as a whole. The main reasons for distortion are imperfect lens shape, which can lead to radial distortion, and improper lens assembly, which can additionally cause tangential distortion. An example of radial distortion is shown in Figure 2.7.

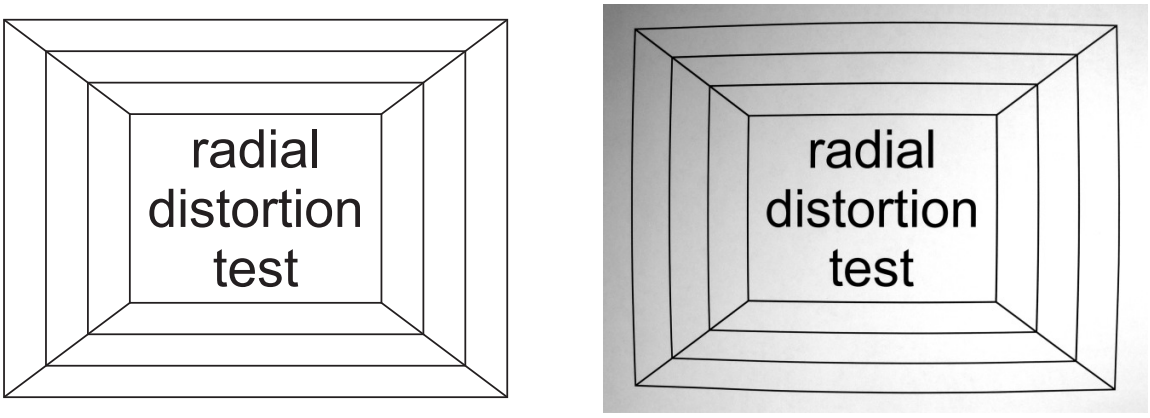
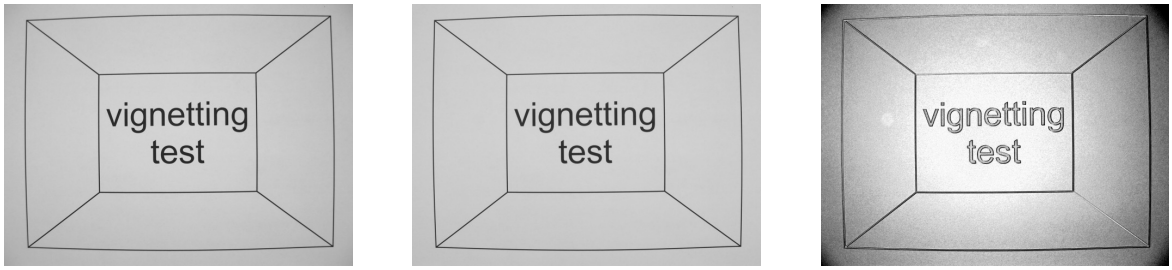


Figure 2.7: This figure illustrates the effects of radial distortion. The left image shows a downsized version of the original test pattern. The radial distortion of the lens causes the lines in the captured image to be slightly curved.



(a) (b) (c)

Figure 2.8: This figure shows the correlation between vignetting and the used aperture size. In image (a), which was captured with a wide open aperture, the corners are slightly darker than the center. Image (b) was captured with a smaller aperture and exhibits a much more uniform brightness. The difference image (c) illustrates the reduction of brightness towards the image corners caused by vignetting in image (a).

According to [Zha96], distortion in a standard compound lens is likely to be completely dominated by the radial components. In addition to that, the distortion of most standard lenses can be accurately modeled with at most two radial distortion factors D_1 and D_2. In conformance with [Zha00], we specify the distorted image coordinates $\boldsymbol{q}_d$ in terms of the undistorted coordinates $\boldsymbol{q}$, assuming that the distortion center coincides with the optical center of the camera.

$$\boldsymbol{q}_d = \left(1 + D_1 r^2 + D_2 r^4\right) \boldsymbol{q}, \quad r = \|\boldsymbol{q}\| \tag{2.8}$$

As the radial distortion factors can be regarded as additional intrinsic camera parameters, we estimate them during camera calibration using external algorithms. Good starting points for further details on algorithms for estimating the distortion factors are both [Zha00] and [Har05].

Compound lenses are prone to two additional effects, which partially belong to radiometric image formation. Firstly, light rays of different wavelengths are refracted at the lens elements in a slightly different way, which creates color fringes at contrast borders. Secondly, various apertures inside the compound lenses block some of the light entering the lens, which can lead to a drop in brightness at the image borders. This effect, which is also called vignetting, is usually not perceived by the casual observer, because the human eye is very insensitive to smooth brightness gradients. With the help of a difference image, which reduces external influences like non-uniform illumination, we illustrate the effect of vignetting in Figure 2.8.

2.3 Radiometric Image Formation

Radiometric image formation is concerned with the relation among the amounts of light energy emitted from light sources, reflected from surfaces, and registered by sensors. Other important topics of this section include the difference between incident energy and perceived energy, the intricacies of the representation of intensity and color values, and the technical properties of real cameras. The contents of this subsection are important for the discussion of feature point tracking algorithms in the following chapter, because these algorithms operate directly on the intensity values of digital images.

2.3.1 Light and Color

There are two different models for describing the phenomenon called light. On the one hand, it can be thought of as propagating sinusoidal electromagnetic waves. On the other hand, it can also be visualized as a stream of massless particles called photons, which contain a certain amount of energy. The amount of energy in a photon is inversely proportional to the wavelength of the corresponding electromagnetic wave.

Many aspects of radiometry are influenced by the human visual system. One of these aspects is the notion of visible light, which is solely determined by the properties of the human visual system. The human eye can register electromagnetic waves with a wavelength between approximately 0.4 and 0.8 micrometers. In order to faithfully reproduce the appearance of the real world, the sensors in modern digital cameras are designed to emulate this behavior.

The notion of color is another aspect of radiometry that is heavily influenced by the capabilities of the human visual system. The six to seven million cones in the human eye are responsible for color vision. They can be divided into three categories, which are sensitive to three different portions of the visible electromagnetic spectrum. These portions roughly correspond to the colors red, green, and blue. As each of the cone types only provides a single intensity as output, it is possible to describe colors by specifying a visually equivalent color created by combining

three primary colors with fixed wavelengths. This principle is used for the RGB color space, which represents colors with three separate intensities, one for the red, green, and blue channel, respectively. The RGB color space is the native description for color values presented on television sets and computer screens, which combine subpixels with the three primary colors to full color pixels. There are also numerous other color spaces besides RGB, which are not relevant for this thesis.

2.3.2 Sources, Surfaces, and Sensing

Any light captured by a camera originally emanates from a light source. The preferred quantity for measuring the amount of light in space is radiance, which is defined as the power of light traveling at a specified point in a specified direction, per unit area perpendicular to the direction and per unit solid angle. A related quantity for measuring the amount of light incident on a surface is irradiance, which is defined as the power of incident light at a specified point from a specified direction, per unit area of the surface and per unit solid angle. As standard light sources are not monochromatic, radiance and irradiance have to be measured for every possible wavelength.

When a light ray hits the surface of an object, it is either absorbed or reflected in a particular direction. A model for this behavior is called a surface reflectance model, and the bidirectional reflectance distribution function (BRDF) is the most general surface reflectance model. It describes the ratio of the radiance in the outgoing direction to the incident irradiance, depending on the angles of the incoming and outgoing light rays. As it is difficult and error-prone to measure the BRDF of a surface, simpler surface reflectance models have been developed. The Lambertian model approximates the behavior of rough non-specular surfaces like cotton cloth, matte paper, and matte paint. Its BRDF is independent of incoming and outgoing direction, which implies that each surface point appears equally bright from all viewing directions. To be more precise, the power of light reflected from a Lambertian surface is proportional to the albedo of the surface, which is a constant depending only on its material, and to the cosine of the angle between the incoming light and the surface normal.

As a summary, the radiance emitted by a Lambertian surface does not change for a moving camera. In addition to that, for flat Lambertian surfaces the radiance is always directly proportional to the albedo when the surface or the light source move. Unfortunately, many surfaces exhibit some degree of specular reflection, which does not conform to the Lambertian model. One property of specular surfaces is to produce reflections of light sources when observed from a special viewing angle. These highlights are much brighter than their surrounding and can cause strong appearance changes for small movements of the object, the camera, or the light source, which is a problem for our feature point tracking algorithms. Figure 2.9 shows a test target with two different surfaces under changing illumination conditions.

Figure 2.9: The four images of this figure show a planar test target, which has a non-specular surface on the left side and a highly specular surface on the right side. The appearance of the specular surface changes vigorously when it reflects the light rays from the light source to the camera.

With the help of the thin lens model, it is possible to verify that the irradiance at the camera sensor is proportional to the radiance leaving an object. Although the irradiance falls off as the light rays deviate from the optical axis, this phenomenon is usually much smaller than the vignetting discussed in the last subsection [For03]. At the sensor, another quantity influences the image formation process. The luminance is a measure for the power of light that an observer perceives. It is linked to irradiance by a monotonic relation depending on the sensor. For example, red light with high radiance will result in small luminance for a sensor that is most sensitive in the blue portion of the spectrum.

During the image formation process, the luminance values observed for every wavelength are accumulated by the sensor according to its sensitivity. After that, they are translated into digital intensity values during quantization. The result is either a gray-level image with one intensity value per pixel, or a color image, which is often encoded in the RGB color space with three color values per pixel. When the resulting image is inspected by a human, the term brightness is used to indicate the subjective human perception of the image intensity. Experimental evidence suggests that brightness is a logarithmic function of intensity [Gon01].

2.3.3 Gamma Correction

As we have shown in the previous subsection, the luminance values observed by a camera depend on a large number of factors, including the position, power, and spectral properties of the light sources, the reflection properties and orientation of the corresponding surface, and the position and spectral sensitivity of the imaging sensor. With some generous assumptions, like the Lambertian model for surface reflection, it is possible to deduce that surface albedo and image intensity are directly proportional [Jin01]. However, one step that is often forgotten in this context is the non-linear power-law transformation from luminance to intensity, which is commonly referred to as gamma correction.

Gamma correction is often said to be needed for compensating the non-linear response of the electron gun of standard CRT monitors. This proposition is wrong. In reality, the non-linearity of a CRT monitor is almost the inverse of the intensity

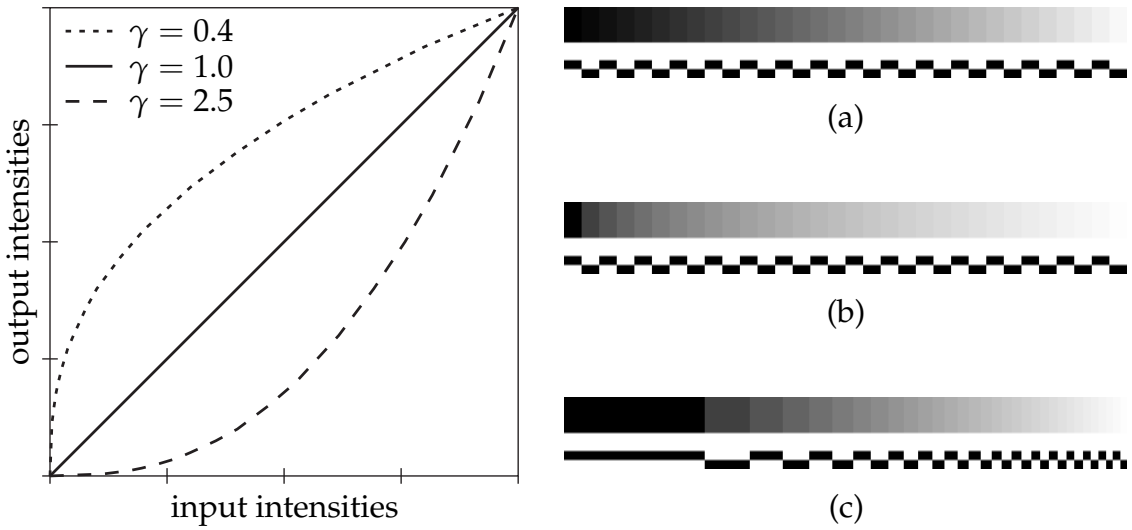
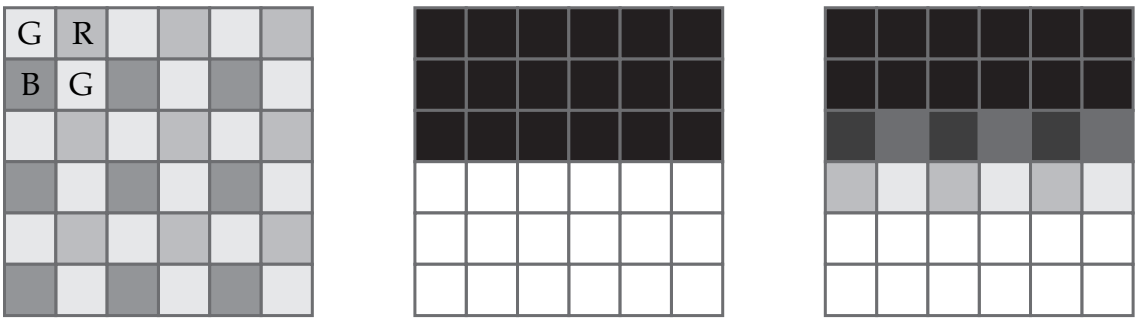


Figure 2.10: The left graph illustrates the relation between input intensities and output intensities for gamma correction with different parameters. Image (a) depicts a gamma-corrected gray-level gradient with a precision of five bits. Linearly coded intensity values with the same precision generate the gradient in image (b). When this gradient is rearranged to be perceptually uniform, the gradient in image (c) is obtained. In the dark areas of this gradient, the lower quality of linearly coded intensity values can clearly be seen.

sensitivity of human vision, so that the response of the CRT monitor is roughly perceptually uniform. The main purpose of gamma correction in computer video and graphics applications is to code the luminance values into a perceptually uniform domain, in order to optimize the perceptual performance of a limited number of bits in each intensity value. In order to reach the quality of eight bit gamma-corrected coding, linearly coded intensity values have to be stored with at least eleven bits [Poy98].

The relation between input intensity and output intensity is shown for three important gamma values in the graph in Figure 2.10. A transformation with a gamma value of 2.5 represents the typical non-linear response of a CRT monitor, whereas the transformation with a gamma value of 0.4 is often used to encode luminance values into gamma-corrected intensities, because it is the inverse of the first transformation. There are two possible choices when working with gamma-corrected intensities in computer vision. The first choice is to accept or ignore the non-linear relation between luminance values and intensity values. The second choice is to linearize the intensity values by applying an inverse gamma correction, which implies that the number of bits per value has to be increased to maintain the same precision. For our feature point tracking algorithms, which operate on small feature windows, a linearization of the intensity values does not yield significant improvements. Therefore, we deliberately chose to work with gamma-corrected intensities in this thesis.



(a) (b) (c)

Figure 2.11: Image (a) shows a standard Bayer color filter array, which contains twice as many cells for green as for red or blue. When the sample input signal in image (b) is recorded with a Bayer sensor and interpolated with a simple linear interpolation algorithm, the resulting full color image (c) contains artifacts.

2.3.4 Real Cameras

Similar to the subsection on real lenses, this subsection provides supplementary considerations for computer vision with respect to the practical aspects of the radiometric image formation process. Some important details are governed by the properties of the image sensor. Most cost-effective digital cameras use a single image sensor, which is covered by a pattern of red, green, and blue color filters. The predominant layout for these filters is the Bayer color filter array pattern, which is illustrated in Figure 2.11. In order to produce a full color image, the two missing color components for each pixel have to be interpolated by a demosaicing algorithm [Mal04, Gun05, Hir05]. This process can lead to unwanted zippering artifacts, which are demonstrated in Figure 2.11. These artifacts may introduce errors into the image that are detrimental to the accuracy of image processing algorithms, even when the obtained color images are converted to gray-level images.

There are further physical phenomena related to the sensor that influence the image formation process. A typical CCD sensor in a digital camera has one cell for each pixel, and each cell converts incoming light energy into electric charge. This charge is converted into a measurable voltage by an output amplifier. When the incoming light is too strong, the charge stored in a cell overflows to neighboring cells, and consequently alters the recorded charge there. This phenomenon is also known as blooming. For several other reasons, including fabrication defects, thermal and quantum effects, and quantization noise, the actual pixel intensity can deviate from the expected value. These effects are subsumed as image noise.

The final issue of this subsection deals with controlling the amount of light allowed to fall on the sensor, which is also called exposure in photography. Each sensor has a certain dynamic range, which is defined as the ratio between the smallest and largest measurable luminance. Luminance values outside this range are trun-

cated, and information is lost when this occurs. A digital camera offers three possibilities for controlling the exposure and keeping the luminance values inside the dynamic range:

- The aperture controls the amount of incoming light by adapting the size of a circular opening inside the lens. Incidentally, closing the aperture also increases the depth of field of the lens, which allows a larger amount of the scene to be in sharp focus.
- The shutter defines the duration of the exposure. If the camera or the objects in the scene move, a short exposure time helps to decrease the amount of motion blur in the image.
- For dark scenes, a digital camera may provide the possibility to adjust the electronic amplification of the output amplifier. This setting is also known as gain. Unfortunately, increasing the gain also considerably increases the image noise.

For our algorithms, large depth of field, short exposure time, and low noise are desirable. As a consequence, a compromise for the three settings that control the exposure has to be found. Alternatively, strong illumination helps to keep all three settings at acceptable levels.

2.4 Multiple View Relations

The image formation process records a lot of information about the captured scene. However, the distance of a scene point from the camera is not accessible in a single image. Only with two or more images, it is possible to extract the depth of a scene point by triangulation. For this purpose, the images can either be acquired simultaneously by a stereo camera head or sequentially by a moving camera, as long as the captured scene does not contain any other moving objects. This subsection explains the basic geometry between two perspective views and introduces important algebraic entities for describing this geometry. Consequently, the contents of this subsection represent the theoretical foundation for the structure and motion estimation algorithms described in Chapter 4.

2.4.1 Epipolar Geometry

Epipolar geometry describes the geometric relations of two perspective views, both of which perform a projection according to (2.7). An illustration of epipolar geometry is given in Figure 2.12. The two perspective cameras are represented by their centers of projection, $\boldsymbol{o}_l$ and $\boldsymbol{o}_r$, and by their image planes, π_l and π_r. The figure also shows a scene point $\boldsymbol{p}$, its projection $\boldsymbol{p}_l$ onto the left image plane, and

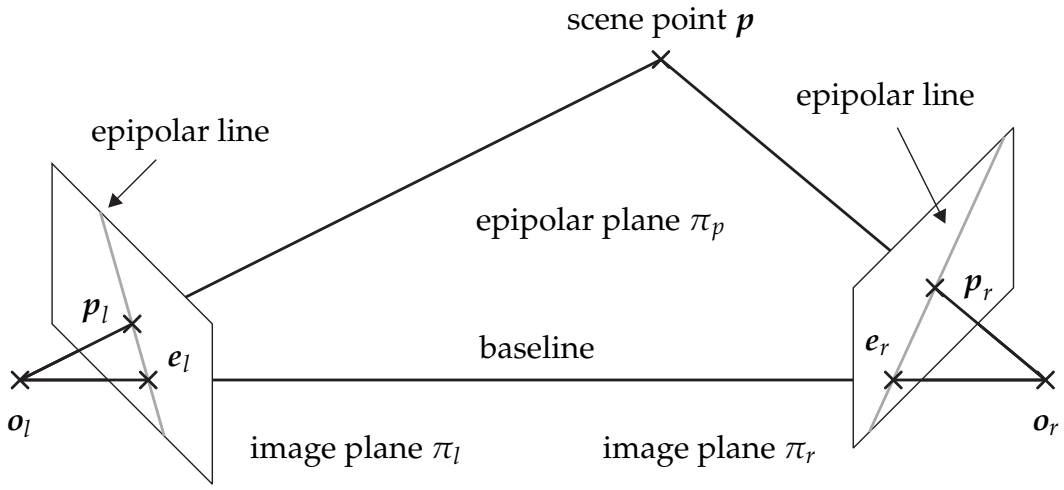


Figure 2.12: Epipolar geometry

its projection $\boldsymbol{p}_r$ onto the right image plane. For the purely geometric discussion in this subsection, the representation of points and lines is independent of any specific coordinate system. Additionally, Figure 2.12 illustrates four important entities of epipolar geometry.

- The baseline is the line joining the two centers of projection.
- An epipole is a point defined by the intersection of the baseline with one of the image planes. As there are two image planes in epipolar geometry, there are also two epipoles, $\boldsymbol{e}_l$ and $\boldsymbol{e}_r$.
- An epipolar plane π_p contains the baseline and an arbitrary scene point $\boldsymbol{p}$. Consequently, there is a whole family of epipolar planes.
- An epipolar line is defined by the intersection of an epipolar plane with one of the image planes. The two epipolar lines belonging to the same epipolar plane are called conjugated epipolar lines.

It is possible to deduce additional properties of epipolar geometry from the preceding definitions. Since both epipolar planes and image planes contain the epipole of an image, every epipolar line, which is the intersection of an epipolar plane with an image plane, goes through the epipole of the image. As a direct consequence, any image point other than the epipole lies on exactly one epipolar line.

The practical importance of epipolar geometry is largely based on another property. Given one point $\boldsymbol{p}_l$ in the left image, the point $\boldsymbol{p}$ is bound to lie somewhere on the ray from $\boldsymbol{o}_l$ through $\boldsymbol{p}_l$. Thus, $\boldsymbol{p}_r$ must lie somewhere on the image of this ray in the right image, which is exactly the epipolar line through $\boldsymbol{p}_r$. This constraint on the position of the second image point is called the epipolar constraint. It provides a geometric relation between points in one image and lines in the other

image, which simplifies the search for a corresponding point in the second image to a one-dimensional problem. Alternatively, the epipolar constraint allows to verify the validity of a pair of corresponding points.

2.4.2 The Fundamental Matrix

As we have seen in the last subsection, the epipolar constraint is useful for finding corresponding points in a pair of images. Interestingly, it is also possible to proceed in the inverse direction, i. e., to deduce epipolar geometry from point correspondences. This task calls for the fundamental matrix $\boldsymbol{F} \in \mathbb{R}^{3\times3}$, which is an algebraic representation of epipolar geometry.

The most important property of the fundamental matrix is its ability to express the epipolar constraint. Given a pair of corresponding points $\underline{\boldsymbol{x}}_l$ and $\underline{\boldsymbol{x}}_r$ in homogeneous sensor coordinates, the matrix satisfies the equation

$$\underline{\boldsymbol{x}}_r^{\mathrm{T}} \, \boldsymbol{F} \, \underline{\boldsymbol{x}}_l = 0 \, . \tag{2.9}$$

The fundamental matrix encodes information on both the intrinsic and extrinsic camera parameters. It has a rank of two and seven degrees of freedom. Further properties of the fundamental matrix are described in [Tru98] and [Har00].

As it is possible to reconstruct the fundamental matrix from eight or more pairs of corresponding points, epipolar geometry can be reconstructed without any information about the intrinsic or extrinsic parameters of the two cameras. Since we intend to work with calibrated cameras, the intrinsic camera parameters are known in advance and the generality afforded by the fundamental matrix is not required. This leads us directly to the next subsection.

2.4.3 The Essential Matrix

When the intrinsic parameters of a camera are known, the homogeneous image coordinates of a point can be computed from its homogeneous sensor coordinates as follows

$$\underline{\boldsymbol{q}} = \boldsymbol{K}^{-1} \, \underline{\boldsymbol{x}} \, . \tag{2.10}$$

In order to express the epipolar constraint for points in camera coordinates, the fundamental matrix $\boldsymbol{F}$ has to be specialized to the essential matrix $\boldsymbol{E} \in \mathbb{R}^{3\times3}$. In accordance with (2.9), but for corresponding points $\underline{\boldsymbol{q}}_l$ and $\underline{\boldsymbol{q}}_r$ in homogeneous image coordinates, the essential matrix satisfies the equation

$$\underline{\boldsymbol{q}}_r^{\mathrm{T}} \, \boldsymbol{E} \, \underline{\boldsymbol{q}}_l = 0 \, . \tag{2.11}$$

Unlike the fundamental matrix, the essential matrix encodes information on the extrinsic camera parameters only. The relative motion between the two cameras

can be represented by a rotation and a translation, which both have three degrees of freedom. Due to the global scale ambiguity inherent in a stereo system with unknown extrinsic camera parameters, the essential matrix has five degrees of freedom. The rank of the essential matrix is two. Therefore, its determinant, as well as one of its singular values, is zero.

$$\det(\boldsymbol{E}) = 0\,. \tag{2.12}$$

The two non-zero singular values of the essential matrix are equal. According to [Fau93], this property can also be formulated as a constraint on the essential matrix as follows

$$\frac{1}{2}\,(\mathrm{trace}(\boldsymbol{E}\,\boldsymbol{E}^{\mathrm{T}}))^2 - \mathrm{trace}((\boldsymbol{E}\,\boldsymbol{E}^{\mathrm{T}})^2) = 0\,. \tag{2.13}$$

Furthermore, it is possible to characterize essential matrices with a single equation, which yields nine cubic polynomials in the components of $\boldsymbol{E}$

$$\boldsymbol{E}\,\boldsymbol{E}^{\mathrm{T}}\,\boldsymbol{E} - \frac{1}{2}\,\mathrm{trace}(\boldsymbol{E}\,\boldsymbol{E}^{\mathrm{T}})\,\boldsymbol{E} = 0\,. \tag{2.14}$$

The relation between the essential matrix and the fundamental matrix can easily be deduced by comparing (2.9) with a combination of (2.10) and (2.11). When the intrinsic camera parameters of the left and right camera are denoted by $\boldsymbol{K}_l$ and $\boldsymbol{K}_r$ respectively, the fundamental matrix can be written as

$$\boldsymbol{F} = \boldsymbol{K}_r^{-\mathrm{T}}\,\boldsymbol{E}\,\boldsymbol{K}_l^{-1}\,. \tag{2.15}$$

The fundamental matrix can be reconstructed from eight pairs of corresponding points. Thus, it is not surprising that the essential matrix can be reconstructed from corresponding points, too. Since the essential matrix has less degrees of freedom, only five point pairs in camera coordinates are required. As the algorithm for reconstructing the essential matrix forms the basis of our structure and motion estimation system, it is thoroughly detailed in Chapter 4.

When the essential matrix of a stereo system is known, its inherent connection with the extrinsic camera parameters can be used to reconstruct the relative motion between the two cameras. As the choice of the world coordinate system is arbitrary in this situation, we let it coincide with the camera coordinate system of the left camera. Consequently, the extrinsic parameters $\boldsymbol{M}_l$ of the left camera do not change the world coordinates, whereas the extrinsic parameters $\boldsymbol{M}_r$ of the right camera are given by the relative motion between the two cameras

$$\boldsymbol{M}_l = (\boldsymbol{I} \mid \boldsymbol{0}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad \boldsymbol{M}_r = (\boldsymbol{R} \mid \boldsymbol{t})\,. \tag{2.16}$$

For reconstructing the extrinsic parameters $\boldsymbol{M}_r$ of the right camera from the essential matrix $\boldsymbol{E}$, the mathematical relation between these two entities has to be

known. First, the cross product of the translation vector $\boldsymbol{t}$ with a point $\boldsymbol{p}$ can be rewritten as a multiplication with a rank-deficient matrix $\boldsymbol{T}$ as follows

$$\boldsymbol{t} \times \boldsymbol{p} = \boldsymbol{T}_{\times}\, \boldsymbol{p} = \begin{pmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{pmatrix} \boldsymbol{p}\,. \tag{2.17}$$

With this notation, the essential matrix is given by

$$\boldsymbol{E} = \boldsymbol{T}_{\times}\, \boldsymbol{R}\,. \tag{2.18}$$

The preceding discussion contains all relevant information for the recovery of the extrinsic camera parameters from the essential matrix. As the actual algorithm is deeply embedded into our structure and motion estimation system, it is also presented in Chapter 4.

2.5 Summary

The main purpose of this chapter is to provide a thorough introduction to the naming conventions, basic concepts, and selected details of the image formation process. To this end, the section on geometric image formation describes the interrelation between the positions of a scene point in various reference frames, most notably the 3-D world and the digital image. Furthermore, the section on radiometric image formation shows how the intensity and color values in a digital image are generated. Finally, the section on multiple view relations lays the mathematical foundation for the recovery of structure and motion information from image sequences.

Based on the fact that human vision functions effortlessly in everyday life, the casual observer might have the impression that vision itself is an easy task. Therefore, one additional purpose of this chapter is to point out the inherent complexity of vision. This is done by providing a detailed description of the various intricacies of the image formation process. In the light of this description, it is possible to fully appreciate the amazing performance of the human visual system.

The second additional purpose of this chapter is closely coupled with the first one. The description of several details of the image formation process ends with the remark that the detail in question is not modeled in this work, either because its effect is too small to be of much concern, or because modeling the effect is too costly in terms of complexity or performance. Nevertheless, omitting the effects introduces errors, whose impact on the final result of the computer vision algorithms has to be minimized. Thus, this chapter helps to motivate the use of robust techniques for all presented computer vision algorithms.

Chapter 3

Feature Point Tracking

Feature point tracking is the first integral part of our approach for sparse 3-D reconstruction. Its main task is to compute the sensor coordinates of distinct scene elements in the images of a video sequence by analyzing the intensity values in these images. Consequently, the success of feature point tracking depends on the careful consideration of the radiometric properties of image formation, which have been described in Section 2.3. In our approach for sparse 3-D reconstruction, the computed coordinates constitute the input data for the structure and motion estimation presented in Chapter 4. Thus, feature point tracking has a strong influence on the quality of the final reconstruction.

The utility of feature point tracking is not limited to sparse 3-D reconstruction. There are a large number of additional applications for it, some of which are succinctly presented in Chapter 7. In light of the diverse requirements of the different applications, high versatility has become the most important design goal of our feature point tracking system. To this end, we facilitate the manual configuration of all components that involve a trade-off between performance criteria like accuracy, efficiency, and robustness. In addition to that, the effects of all important configuration options are described in this chapter. As the different application areas do not share any common model knowledge, we have concentrated our efforts on a fully data-driven tracking system.

In the first section of this chapter, we give an in-depth introduction to the problem of feature point tracking. We also present and classify existing algorithms that are part of this problem area. The next section contains a description of the Kanade-Lucas-Tomasi tracker and its most important extensions, which form the basis of our tracking system. In the third section of this chapter, we present our tracking system, which contains several significant enhancements. Subsequently, the section about high-speed tracking describes our adaptations of the system to the requirements of the self-localization of the VAMPIRE augmented reality gear. This chapter is concluded by a short summary.

Figure 3.1: This sequence of six images of a real-world scene illustrates common challenges for feature point tracking. A feature window is marked in the middle of the first image. In the following four images, the appearance of this feature window changes due to the rotation of the object, varying illumination conditions, the perspective distortion of the non-planar surface of the object, and the occlusion by another object. Finally, the last image indicates that tracking has to stop when the feature window leaves the field of view.

3.1 Overview

3.1.1 Problem Statement

In general, image features can be described as parts of the image with special properties. Useful image features must be detectable, which means that there are algorithms for locating instances of the feature in an image. Furthermore, image features are meaningful when they correspond to an interesting element of the captured scene. The image features used in feature point tracking are called feature points, point features, or corners. The combination of all relevant properties of a feature point is called a feature descriptor. The most important property of a feature point is its 2-D position in sensor coordinates. As the position and the single intensity value at this position do not provide enough information to detect or track a feature point, a common approach is to represent feature points by small image regions, which are also called feature windows. Their size lies between 5×5 pixels and 31×31 pixels in most applications.

Given a sequence of M digital images $(f_1(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x}))$ as input, a feature point tracking algorithm has to perform at least two basic tasks. Its first task is to detect suitable feature points in one or more of the given images, and its second task is to compute the position of corresponding feature points in other images of the se-

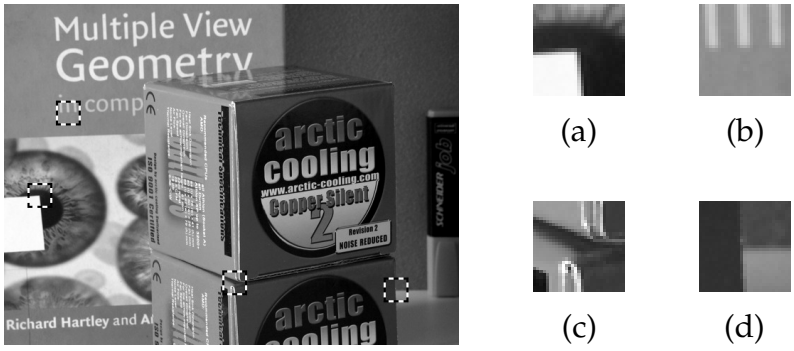


Figure 3.2: This figure illustrates potential problems in relating a 2-D feature window to a unique 3-D feature point. The feature windows (a) and (b) both lie on a planar surface. Feature window (a) also has a unique texture, which is the ideal case for feature point tracking. However, feature window (b) looks exactly like an image region to its right, which can lead to erroneous feature correspondences. Although the feature windows (c) and (d) do not lie on a planar surface, feature window (c) can be reliably attributed to a unique 3-D feature point. In contrast to this, feature window (d) spans across a depth discontinuity and does not represent a unique point in 3-D space.

quence. The main output of a feature point tracking algorithm is a set of feature trails, which specify the 2-D position of a feature point for every image in which the position has been successfully computed.

All feature point tracking algorithms rely on the validity of several basic assumptions. The first assumption concerns the detection of feature points, which requires that the intensity values of an image exhibit some kind of variance. It is obvious that an image that is uniformly colored, e. g., an image that is completely white, does not contain any suitable feature points. Another assumption is that the movement of the feature points in the sensor coordinate system does not exceed a reasonable threshold for consecutive images in the video sequence. An extreme violation of this assumption is a video sequence in which the camera turns so fast that every feature point is only visible in one single image. This example implies that the frame rate of the video camera has to be sufficiently high in relation to the motion of the camera and the scene objects.

In order to be able to compute the positions of a feature point in the images of a video sequence, feature point tracking algorithms have to assume that the appearance of the scene objects remains more or less constant throughout the whole sequence. As Figure 3.1 illustrates, there are many possible violations of this assumption in video sequences of real-world scenes. Another example are the strong appearance changes of specular and shiny surfaces discussed in Section 2.3. Consequently, successful feature point tracking is only possible when the tracking algorithms tolerate at least some degree of appearance change.

The most important assumption for the correctness of the computed feature trails is that corresponding feature points in the image sequence can all be attributed to

the same 3-D feature point in the captured scene. In addition to the specular reflections mentioned in the preceding paragraph, two more configurations are likely to cause problems with regard to this assumption. When a scene object is covered with a repeating texture, as illustrated in Figure 3.2, the feature point tracking algorithm can fail to identify the correct correspondence. Figure 3.2 also shows a feature point which spans across a depth discontinuity. Such a feature point is not guaranteed to represent a single 3-D feature point in the scene when the camera moves.

As we have detailed, the input image sequences can violate the basic assumptions of feature point tracking in many different ways. The feature point tracking problem is further complicated by the shortcomings of lenses and cameras described in Subsections 2.2.4 and 2.3.4, like distortion, vignetting, sensor noise, or motion blur. Feature point tracking algorithms are usually data-driven and work without any information about the camera motion or the scene structure. Consequently, it is not realistic to expect that all computed feature positions are absolutely correct.

Feature point tracking mainly incurs two different error types. The first kind of error is a small deviation of the computed feature position from the correct feature position. The average magnitude of the deviation strongly depends on the input data, but due to effects like discretization artifacts and sensor noise it can never be completely eliminated. The second kind of error is more severe, as it usually entails a large deviation of the computed feature position. It occurs when a feature correspondence is erroneous, i. e., when one feature trail is formed by two or more distinct feature points. Unfortunately, a data-driven feature point tracking algorithm cannot reliably detect this kind of error. Thus, all algorithms that use feature trails as input data should be prepared to encounter both error types.

As feature point tracking algorithms are an important component in many computer vision applications, they have to cope with a wide range of different input image sequences. Accordingly, there are several different properties for describing the input data. The most obvious property is the resolution of the input images. It directly affects the required memory space and computation time, and influences the selection of a suitable feature window size. Another property is the format of the image data. A pixel can either be represented by a single intensity value or by multiple color values, and each value is stored with a specified precision. We work with 8-bit gray-level images in this thesis.

The maximum number of images in a video sequence affects important components of a feature point tracking algorithm. When the maximum sequence length is small, feature detection has to be performed only once, typically in the first image of the sequence. In contrast to this, long image sequences require feature detection whenever the number of successfully tracked features falls below a certain threshold. The last important property of the input data is the frame rate of the video sequence. When the feature point tracker has to process the incoming images in real-time, the frame rate determines the maximum computation time per image. In addition to that, the frame rate directly influences the maximum movement of a feature point in the sensor coordinate system from one frame to the next.

The widespread adoption of feature point tracking not only created the need to cope with a wide range of different input data, but also generated a diversity of application requirements. In order to facilitate the systematic comparison of different feature point tracking algorithms, we propose to categorize these application requirements into four performance criteria:

- The first performance criterion is the basin of convergence. In the context of feature point tracking, it describes the maximum translation of a feature point from one frame to the next that can be reliably estimated by the feature point tracking algorithm. When it is possible to control the absolute value of translation in the video sequence, either by limiting the motion of the camera and the scene objects or by adjusting the frame rate of the video camera, a relatively small basin of convergence can be sufficient. Some tracking algorithms provide a way to directly specify their basin of convergence. Increasing the basin of convergence, either directly or indirectly, is usually achieved at the expense of the computation speed of the tracking algorithm.

- This leads us to the computational efficiency of the tracking algorithm, which is crucial for real-time systems, but less important for off-line systems. For a more detailed analysis, the operations of a tracking algorithm can be divided into operations that are performed for every processed frame and operations that are performed for every tracked feature. Consequently, the relative efficiency of a tracking algorithm depends on the average number of features that have to be tracked.

- Another important performance criterion of a tracking algorithm is the accuracy of the feature positions in the feature trails. As the feature trails are often important input data for other components of a computer vision application, their accuracy governs the accuracy of the final output.

- While accuracy can be interpreted as a measure for the average deviation from the correct result under standard conditions, the robustness of a tracking algorithm subsumes its ability to avoid large errors under adverse conditions. In the case of feature point tracking, an adverse condition is any strong violation of the model assumption that a feature window only moves in the sensor coordinate system without changing its appearance. This can happen for a large number of reasons, including strong noise in the input images, the distortion or occlusion of a feature window, or changes in the illumination of the scene.

Finally, different applications have different requirements for the output data of a tracking algorithm. Normally, all feature trails are passed to the subsequent components of the computer vision application. Some algorithms for structure and motion estimation, however, can only work with complete feature trails, which include a position for every frame of the processed video sequence. It is also possible that additional information about the tracked features, like a confidence measure for the

correctness of the estimated position, is needed by other components of the computer vision application. These requirements can also prove to be important for the choice of a suitable tracking algorithm.

3.1.2 Related Work

Most existing feature point tracking algorithms can be classified into two groups. The algorithms of the first group detect feature points in every frame of the video sequence and then build the feature trails by establishing correspondences between the detected features. In contrast to this, the second group of algorithms detects feature points only once per feature trail. The feature positions in the other frames of the sequence are directly computed from the intensity values in the known and tentative feature windows. Despite their differences, both groups of algorithms use feature detection as an important processing step.

A large number of feature point detection algorithms compute a quality measure for every pixel of an image. Subsequently, the resulting interest image is scanned for local maxima of the feature quality measure. Moravec proposed one of the first algorithms of this kind in [Mor80]. His algorithm computes the similarity of a feature window to slightly translated copies of itself. The feature quality is determined by the minimum dissimilarity for all tested directions.

Harris and Stephens describe an auto-correlation detector that enhances several key aspects of Moravec's algorithm [Har88]. For example, the new detector makes the quality measure isotropic and decreases its sensitivity to edges. Tomasi and Kanade present a feature quality measure that is mathematically derived from the tracking equations of the Kanade-Lucas-Tomasi tracker [Tom91]. Interestingly, the resulting feature detector is closely related to the Harris detector. Both feature detection algorithms will be presented more thoroughly in Subsection 3.2.2.

Unlike the two previous algorithms, the SUSAN (smallest univalue segment assimilating nucleus) feature detector proposed by Smith and Brady does not require the computation of image derivatives [Smi97]. Instead, the quality measure is defined by the number of pixels with similar intensities in a local window around the center pixel. The parametric feature detector presented in [Bak98] represents different kinds of features in a low dimensional subspace of Hilbert space. Rosten and Drummond employ a machine learning approach to speed up their FAST (feature from accelerated segment test) algorithm for real-time feature detection [Ros06]. An evaluation of a large number of feature point detectors can be found in [Sch00].

The mathematical properties of feature detection algorithms are analyzed with the help of condition theory in [Ken03]. Other recent topics in the area of feature detection include scale invariance and affine invariance. One way to achieve scale invariance is to convolve the image with a Difference of Gaussians kernel at multiple scales, as demonstrated by Lowe with his SIFT (scale invariant feature transform) feature framework [Low04]. Mikolajczyk and Schmid combine the computationally more expensive Laplacian of Gaussian method with an affine invariant algorithm

in [Mik04]. A comparison of several different affine invariant feature detectors is presented in [Mik05b].

The choice of a suitable feature detector strongly depends on the used tracking algorithm. For the first group of tracking algorithms described above, it is highly important that feature points detected in one image are detected at corresponding positions in subsequent images. The detected feature points should also be distinguishable from non-corresponding feature points by examining a suitable feature descriptor.

There are a large number of algorithms that only use the positions of detected feature points for correspondence estimation [Che99b, Vee01, Sha05]. However, when the images containing the features points are available, it is possible to devise much more powerful feature descriptors. For example, the intensity values in a small window around the feature are a straightforward addition to the feature descriptor. Thus, with the help of a similarity measure for feature windows, a correspondence can be established by finding the feature with the most similar feature window. Nister uses this technique together with a mutual consistency check in [Nis04c]. Bretzner and Lindeberg present another feature tracker, which incorporates automatic scale selection and a more elaborate multi-cue correspondence measure in [Bre98]. Along with an evaluation of feature detectors, two more tracking algorithms are described in [Tis04]. Finally, [Mik05a] contains an evaluation of a comprehensive collection of feature descriptors with respect to correspondence estimation.

In contrast to the first group of tracking algorithms, the second group computes the position of corresponding feature points directly from the intensity values of the images. This approach is closely related to the computer vision problem of optical flow, which aims to compute a dense motion field between two images for all image pixels. In order to compute the two dimensional motion vectors from one dimensional intensities, Horn and Schunck introduced the assumption of a smooth motion field [Hor81]. A comprehensive evaluation of optical flow techniques can be found in [Bar94]. Another overview, albeit without evaluation, is given by Mitiche and Bouthemy in [Mit96].

Contrary to global optical flow estimation, a feature point tracking algorithm computes the constant local optical flow, i. e., the translation of a feature window. On the one hand, this can be done by iterative gradient-based algorithms like the Kanade-Lucas-Tomasi tracker [Luc81, Tom91], which will be discussed in more detail in Section 3.2. On the other hand, it is possible to determine the translation of a feature window by matching it to all windows in a specified search region, using a suitable similarity measure. This approach, which is commonly referred to as block matching, is mainly used for disparity estimation in stereo image pairs [För86, Kan94].

Despite their apparent differences, iterative gradient-based algorithms and block matching have been proven to be mathematically equivalent [Dav95]. Recently, new approaches have also bridged the gap between local and global optical flow estimation. In [Süh02] the global optical flow of an echocardiogram is determined

with the help of a Kanade-Lucas-Tomasi tracker adapted to estimate local affine motion. Bruhn et al. present a hybrid method that combines local and global optical flow estimation [Bru05]. It is more robust to noise than a standard optical flow algorithm and provides a confidence measure for the correctness of the computed flow field.

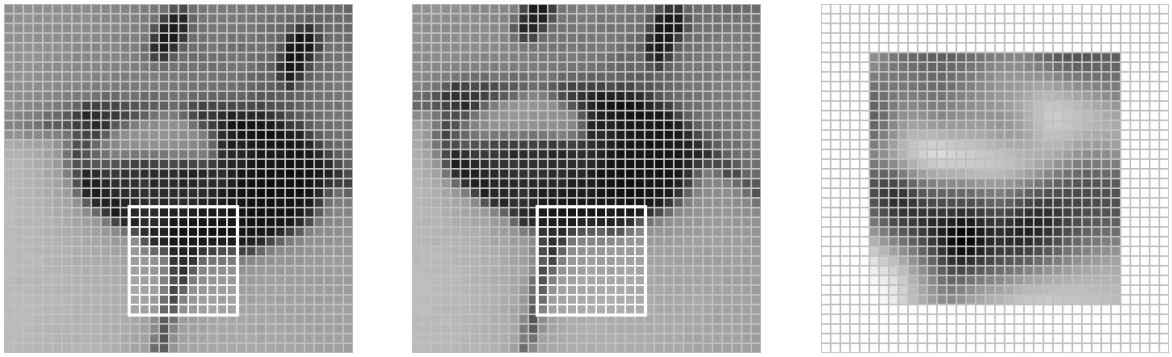
Apart from the basic tracking algorithms themselves, there are further interesting advances in the area of feature point tracking. Toyama and Hager present a framework that incorporates a hierarchy of tracking algorithms for increased robustness in [Toy99]. Algorithmic fusion is proposed to achieve the same effect in [McC02]. Increased robustness is also reported for a tracker that employs a spline-based global deformation model to guide the feature point tracking process [Kan97]. In [Hei99] the Kanade-Lucas-Tomasi tracker is extended to work with color images. A mathematical analysis of the performance bounds for gradient-based tracking algorithms is presented in [Rob04].

The integration of model knowledge into the feature point tracking algorithm is another active research area. When the scene is static, a global image registration can be used to compensate for camera motion [Zhe95]. In this case, it is also possible to detect outliers by an online reconstruction of the 3-D scene structure [Nis00]. Other approaches that exploit model knowledge include feature point tracking for faces with Gabor wavelets [Wie02], extending interrupted trails under the assumption of an affine camera model [Sug04], and grouping of feature points belonging to independently moving objects [Siv06].

Algorithms for tracking a small number of larger regions are commonly referred to as object tracking, region tracking, or template matching algorithms. They are closely related to feature point tracking algorithms, but due to the larger size of the tracked regions, they have to cope with stronger appearance variation and are more prone to severe occlusions. As failing to track an object is usually more severe than losing a feature point, object tracking algorithms employ a wide range of techniques to make them more robust. Consequently, recent publications on object tracking like [Bla98, Scl98, Hag98, Jur02, Bue04, Geo04, Buc06] provide valuable insight that can also be applied to feature point tracking algorithms, e. g., techniques for coping with partial occlusions of the tracked region. Furthermore, it is also possible to combine both tracking types. For example, a block matching algorithm provides input for a model-based object tracking algorithm in [Nic02], and SIFT features are employed in the initialization of another model-based object tracking algorithm in [Grä05].

3.2 The Kanade-Lucas-Tomasi Tracker

The Kanade-Lucas-Tomasi tracker is an iterative approach for correlation-based feature point tracking. Compared to the other feature point tracking algorithms described in the preceding section, it provides a high level of computational efficiency,



(a) (b) (c)

Figure 3.3: This figure illustrates the basic principle of correlation-based feature point tracking. The reference image $f_r(\boldsymbol{x})$ in image (a) contains an exemplary feature window $\mathcal{W}$. Image (b) shows that the feature window is out of place in the current image $f_c(\boldsymbol{x})$, when the scene objects move relative to the camera. Image (c) illustrates the values of the error function $\epsilon(\boldsymbol{d})$ for all possible positions of the feature window inside the current window. In this context, dark pixels denote a small value of the error function, i. e., a small difference between the intensity values of the feature window in the reference image and the translated feature window in the current image.

accuracy, and robustness. Furthermore, it is known to be very versatile and highly customizable, which is confirmed by the large number of existing enhancements to the basic algorithm. Thus, we have chosen the Kanade-Lucas-Tomasi tracker as the basis of our tracking system.

3.2.1 Basic Principles

The basic tracking principle of the Kanade-Lucas-Tomasi tracker (KLT tracker) was proposed by Lucas and Kanade in [Luc81]. As their paper is concerned with solving an image registration problem, where the region of interest is usually specified in advance, it does not address the problem of feature detection. Accordingly, we assume that a suitable feature point is given, and defer the problem of detecting feature points to the next subsection.

The reference image $f_r(\boldsymbol{x})$ in Figure 3.3 contains a feature window $\mathcal{W}$. We define $\mathcal{W}$ to be the set of sensor coordinates $\boldsymbol{x} = (x, y)^{\mathrm{T}}$ of all pixels inside the feature window. Usually, the feature window is a square region around the corresponding feature point. The task of the tracking algorithm is to compute the position of the feature window in the current image $f_c(\boldsymbol{x})$. More precisely, the task is to find the motion parameters which minimize a specified measure for the difference of the intensity values at the corresponding pixel positions of the feature windows in both images. When the motion parameters represent a translation and the difference measure is the sum of squared differences, the following error function has to be

minimized

$$\epsilon_{\text{klt}}(\boldsymbol{d}) = \sum_{\boldsymbol{x} \in \mathcal{W}} \left(f_r(\boldsymbol{x}) - f_c(\boldsymbol{x} + \boldsymbol{d})\right)^2, \quad \boldsymbol{d} \in \mathbb{R}^2. \tag{3.1}$$

The right image in Figure 3.3 illustrates the values of the error function for every possible translation. The block matching algorithms mentioned in the preceding section actually compute the error function for a large number of discrete translations in a defined search region. In contrast to this, the Kanade-Lucas-Tomasi tracker iteratively minimizes the error function. In order to reflect the algorithm's iterative nature, the error function is reformulated as

$$\epsilon_{\text{klt}}(\Delta\boldsymbol{d}) = \sum_{\boldsymbol{x} \in \mathcal{W}} \left(f_r(\boldsymbol{x}) - f_c(\boldsymbol{x} + \boldsymbol{d} + \Delta\boldsymbol{d})\right)^2, \quad \Delta\boldsymbol{d} \in \mathbb{R}^2. \tag{3.2}$$

Between successive iteration steps, the motion parameter vector $\boldsymbol{d}$ is updated according to the rule $\boldsymbol{d} \leftarrow \boldsymbol{d} + \Delta\boldsymbol{d}$. Different ways of setting the initial value of the motion parameter vector $\boldsymbol{d}$ will be discussed in the next section.

For the computation of the vector $\Delta\boldsymbol{d}$, the error function $\epsilon_{\text{klt}}(\Delta\boldsymbol{d})$ is transformed with a first-order Taylor expansion on $f_c(\boldsymbol{x} + \boldsymbol{d})$

$$\tilde{\epsilon}_{\text{klt}}(\Delta\boldsymbol{d}) = \sum_{\boldsymbol{x} \in \mathcal{W}} \Big(f_r(\boldsymbol{x}) - \left(\nabla \boldsymbol{f}_c(\boldsymbol{x} + \boldsymbol{d})\right)^{\mathrm{T}} \Delta\boldsymbol{d} - f_c(\boldsymbol{x} + \boldsymbol{d})\Big)^2, \tag{3.3}$$

where $\nabla \boldsymbol{f}_c(\boldsymbol{x})$ is the image gradient defined by

$$\nabla \boldsymbol{f}_c(\boldsymbol{x}) = \left(\frac{\partial f_c(\boldsymbol{x})}{\partial x}, \frac{\partial f_c(\boldsymbol{x})}{\partial y}\right)^{\mathrm{T}}. \tag{3.4}$$

Now, setting the derivative of the revised error function $\tilde{\epsilon}_{\text{klt}}(\Delta\boldsymbol{d})$ to zero

$$\sum_{\boldsymbol{x} \in \mathcal{W}} \nabla \boldsymbol{f}_c(\boldsymbol{x} + \boldsymbol{d}) \Big(f_r(\boldsymbol{x}) - \left(\nabla \boldsymbol{f}_c(\boldsymbol{x} + \boldsymbol{d})\right)^{\mathrm{T}} \Delta\boldsymbol{d} - f_c(\boldsymbol{x} + \boldsymbol{d})\Big) = \boldsymbol{0} \tag{3.5}$$

and solving for the update vector $\Delta\boldsymbol{d}$ yields

$$\Delta\boldsymbol{d} = \boldsymbol{H}_{\text{klt}}^{-1} \sum_{\boldsymbol{x} \in \mathcal{W}} \nabla \boldsymbol{f}_c(\boldsymbol{x} + \boldsymbol{d}) \left(f_r(\boldsymbol{x}) - f_c(\boldsymbol{x} + \boldsymbol{d})\right) \tag{3.6}$$

with the matrix $\boldsymbol{H}_{\text{klt}} \in \mathbb{R}^{2\times 2}$ defined as

$$\boldsymbol{H}_{\text{klt}} = \sum_{\boldsymbol{x} \in \mathcal{W}} \nabla \boldsymbol{f}_c(\boldsymbol{x} + \boldsymbol{d}) \left(\nabla \boldsymbol{f}_c(\boldsymbol{x} + \boldsymbol{d})\right)^{\mathrm{T}}. \tag{3.7}$$

The equations for the motion parameter update vector show that the tracking algorithm of the Kanade-Lucas-Tomasi tracker is a specialized Gauss-Newton gradient descent optimization algorithm. A detailed analysis of the computational cost of the tracking algorithm and a comparison with other approaches for minimizing the error function are given in [Bak04].

In order to apply the tracking algorithm described above, a number of technical details have to be resolved. The first problem is the computation of the image gradient, which consists of the partial derivatives of an image in the direction of the two coordinate axes. For this purpose, we rely on numerical differentiation with the Sobel operator. It consists of two 3×3 kernels

$$S_x = \frac{1}{8}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \text{and} \quad S_y = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \tag{3.8}$$

which are convolved with the original image to yield the respective partial derivatives. Interestingly, a Sobel kernel can be decomposed into two or four primitive kernels

$$\begin{aligned} S_x &= \frac{1}{8}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \\ &= \frac{1}{8}\begin{bmatrix} 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \\ &= \frac{1}{8}\begin{bmatrix} 1 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \end{aligned} \tag{3.9}$$

It is a well-known fact that derivative computations emphasize high-frequency noise [Gup97]. The last line of (3.9) illustrates that the Sobel operator counteracts this problem by using three kernels for averaging and smoothing the results of the derivative computation with the first kernel.

As the motion parameter vector is not limited to integer values, it is possible that the pixel positions of a feature window lie between the pixel positions of the image. The most efficient way to solve this problem is to use the intensity value of the nearest pixel, which is also called nearest neighbor interpolation. However, this approach seriously impairs the accuracy of the tracking algorithm. In order to achieve more accurate results, we compute the intensity values with the help of a bilinear interpolation. This technique computes a weighted average of the intensities of the four nearest pixels, which turns out to be a good compromise between accuracy and efficiency. The same method is also applied to the interpolation of the partial derivatives. Many more sophisticated interpolation techniques are described and evaluated in [Mei01]. As these techniques incur a much higher computational cost than bilinear interpolation, they are not considered in this thesis.

Like every iterative algorithm, the Kanade-Lucas-Tomasi tracker requires the definition of one or more stopping criteria. When the algorithm is successful, the computed sequence of motion parameter vectors converges. Consequently, the iterative computation can be stopped when all components of the update vector $\Delta \boldsymbol{d}$ fall below a specified minimum threshold $\theta_{\text{tra\_stop}}$. It is also possible that the sum of

squared differences of the two feature windows increases after an iteration step. In this case, the iteration loop is terminated as well, and the previous motion parameter vector is returned. In addition to that, the maximum number of iterations $\delta_{\text{tra\_iter}}$ can be specified as a safeguard. Finally, the tracking of a feature has to be stopped when it moves out of the image.

3.2.2 Feature Detection

Many different algorithms for detecting feature points have been mentioned in Subsection 3.1.2. One of the strong points of the Kanade-Lucas-Tomasi tracker is the existence of a custom-made feature detection algorithm, which was first presented by Tomasi and Kanade in [Tom91]. Instead of relying on some conceived assumptions about the nature of a good feature, it is directly based on the mathematical properties of the tracking algorithm.

The most important condition for successful tracking is that the computation of the motion parameter update vector described in (3.6) yields dependable results. This is not the case when the entries of the matrix $\boldsymbol{H}_{\text{klt}}$ are dominated by noise. Consequently, the two eigenvalues of $\boldsymbol{H}_{\text{klt}}$ should be large. When the matrix $\boldsymbol{H}_{\text{klt}}$ is not well-conditioned, solving the system in (3.6) becomes unstable. Therefore, the eigenvalues of $\boldsymbol{H}_{\text{klt}}$ should be roughly equal. As the maximum value of the eigenvalues is bounded by the maximum intensity value, Tomasi and Kanade propose to use the minimum eigenvalue of $\boldsymbol{H}_{\text{klt}}$ as a measure for the quality of the corresponding feature window.

The quality of prospective feature windows is typically computed for every pixel of an image. Figure 3.4 illustrates the resulting interest images for different window sizes. Interestingly, the response of suitable image features like corners is not limited to a single pixel. Instead, the interest images show that the feature quality is high in an extensive area around the corner. It can also be seen that the size of this area directly depends on the size of the feature window. This observation motivates the augmentation of the feature detection algorithm with an appropriate feature selection strategy.

Tomasi and Kanade propose to ignore feature windows whose quality lies below a threshold $\delta_{\text{ftr\_qual}}$, which is determined by the quality of the feature windows in an approximately uniform region of the image [Tom91]. Another possibility to prevent the selection of unsuitable feature windows is to apply a local non-maximum suppression as shown in [Nis04c]. There are different strategies for selecting the best remaining features. Tomasi and Kanade build a list of features that is sorted according to feature quality. Then, the best remaining features are selected in an iterative process, while features candidates that overlap existing features are deleted from the list. A more efficient algorithm for feature selection will be presented in Subsection 3.3.2.

A close examination of the selected features reveals that corners are very likely to be positioned at the edges of the feature windows. In addition to that, these

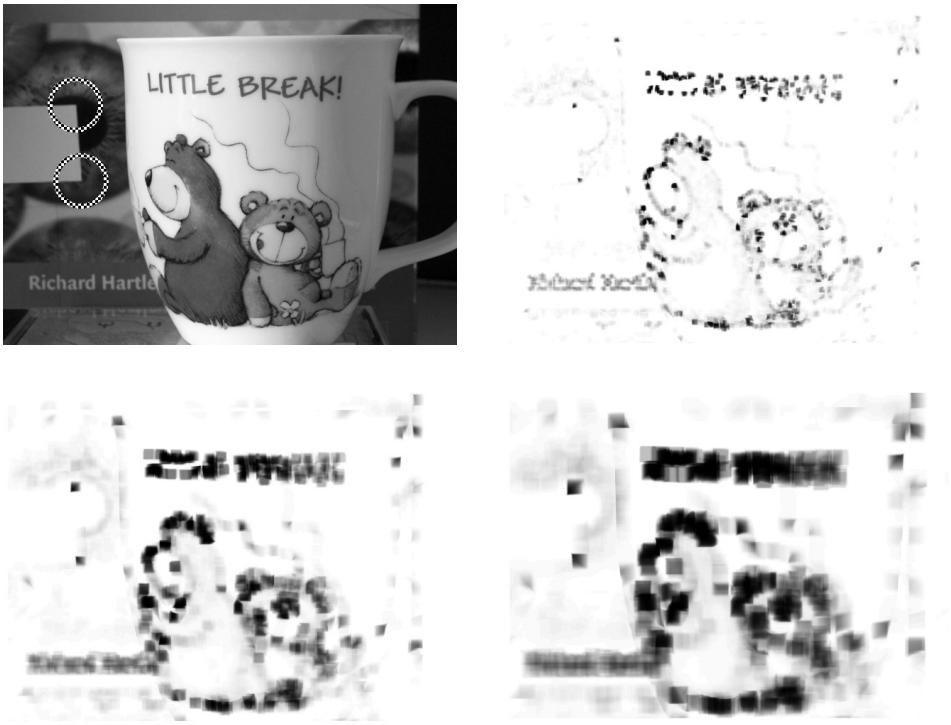


Figure 3.4: An exemplary gray-level image and three interest images for feature windows of 7×7, 15×15, and 23×23 pixels. The feature quality, which is represented by the intensity of the pixels in the interest images, increases from white to black. The areas highlighted by the circles in the first image illustrate that two-dimensional variations in the image intensities, e. g., corners, result in a high feature quality. In contrast to this, feature windows that only include the edge connecting the two highlighted corners are not visibly better than feature windows in homogeneous areas of the image.

feature windows are filled with the brighter part of the image region in an overwhelming majority of the cases. This noteworthy behavior of the feature detection algorithm is illustrated in Figure 3.5 and has already been observed by Tomasi and Kanade [Tom91]. It is explained by the fact that, as long as the corner is inside the feature window, the feature quality is only influenced by the remaining intensity variations, which are usually higher in the brighter regions of the image. This behavior can lead to suboptimal performance of the feature point tracking algorithm. A surprisingly simple and effective solution to this problem is to decrease the size of the feature windows for feature detection. Consequently, even if a corner lies at the edge of the detection window, it lies safely inside the actual tracking window. Another possibility is to emphasize the inner pixels of the detection window by applying Gaussian weights, but our experiments indicate that this method does not further improve the performance of the tracking algorithm.

As the feature quality has to be computed for every pixel of an input image, the efficiency of this computation is very important for real-time applications. The

Figure 3.5: The results of feature selection for the interest images of Figure 3.4. All features windows have a minimum distance of 12 pixels from each other. The two highlighted areas illustrate the phenomenon that corners inside a feature window are often positioned at its very edge, which is especially prominent for the large feature windows in the lower right image.

entries of the matrix $\boldsymbol{H}_{\text{klt}}$ can be determined from the image derivatives by simple operations that involve only summation and multiplication. When the matrix $\boldsymbol{H}_{\text{klt}}$ is given by

$$\boldsymbol{H}_{\text{klt}} = \begin{pmatrix} \sum \frac{\partial f(\boldsymbol{x})}{\partial x}\frac{\partial f(\boldsymbol{x})}{\partial x} & \sum \frac{\partial f(\boldsymbol{x})}{\partial x}\frac{\partial f(\boldsymbol{x})}{\partial y} \\ \sum \frac{\partial f(\boldsymbol{x})}{\partial x}\frac{\partial f(\boldsymbol{x})}{\partial y} & \sum \frac{\partial f(\boldsymbol{x})}{\partial y}\frac{\partial f(\boldsymbol{x})}{\partial y} \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix}, \tag{3.10}$$

its smaller eigenvalue can be written in closed form as

$$q_T = \frac{1}{2}\left(h_{11} + h_{22} - \sqrt{(h_{11} - h_{22})^2 + 4\,h_{12}^2}\right). \tag{3.11}$$

In [Har88], Harris and Stephens present a similar quality measure that avoids the explicit computation of the eigenvalues. It is defined by

$$q_H = \det(\boldsymbol{H}_{\text{klt}}) - K_H \operatorname{trace}(\boldsymbol{H}_{\text{klt}})^2 = h_{11}h_{22} - h_{12}^2 - K_H\,(h_{11} + h_{22})^2. \tag{3.12}$$

Interestingly, the original work contains no advice on a suitable value for the constant K_H. However, values between 0.04 and 0.08 are recommended for K_H in other

publications. We use a value of 0.04. As the quality measure of the Harris detector requires fewer operations than the quality measure proposed by Tomasi and Kanade, its computation is slightly more efficient. Nevertheless, our experiments suggest that both computation speed and detection results of the two quality measures differ only by negligible amounts.

Zivkovic and van der Heyden propose a new feature detection technique, which is based on the size of the convergence region of the feature point tracking algorithm [Ziv04]. By tracking the feature from different starting points near its actual position, the size of the convergence region is estimated. Unfortunately, this technique requires a lot of computation time, which makes it unsuitable for a large number of applications.

3.2.3 Outlier Rejection

The Kanade-Lucas-Tomasi tracker estimates the translation of a feature point from one frame to the next. In other words, the current frame $f_c(\boldsymbol{x})$ becomes the reference frame $f_r(\boldsymbol{x})$ when the tracking is finished, and the next frame in the video sequence becomes the current frame. Consequently, the reference feature window of a feature is constantly changing. In addition to that, a tracking error can never be recovered in later frames, because an erroneous feature window becomes the new reference feature window, which is the de facto standard for the appearance of the feature.

When a severe tracking error occurs, the dissimilarity of the two corresponding feature windows is often conspicuously high. This case can be identified by setting a maximum threshold for the sum of squared differences of the two feature windows. In contrast to this, the accumulation of small tracking errors, which is also known as the feature drift problem, cannot be detected reliably in this way. Unfortunately, small tracking errors are unavoidable, because the feature windows in two consecutive frames will never be identical in a video sequence of a real-world scene. A discussion of several reasons for this phenomenon can be found in Subsection 3.1.1.

In order to enhance the performance of outlier rejection, Shi and Tomasi propose to measure the feature dissimilarity between the original frame and the current frame [Shi94]. With the help of the original feature window, even incremental appearance changes can be detected reliably. Unfortunately, this approach gives rise to a new problem. Although the motion model of pure translation is sufficiently accurate for tracking small feature windows from one frame to the next, it is inadequate for describing the motion of a feature window over longer periods of time. This problem is illustrated by the images of the feature window in the top row of Figure 3.6.

Features that lie on a planar surface of a scene object change their appearance in the image based on the properties of perspective projection. As the feature windows used for feature point tracking are rather small, Shi and Tomasi recommend to ignore the effects of perspective distortion and to use an affine motion model for

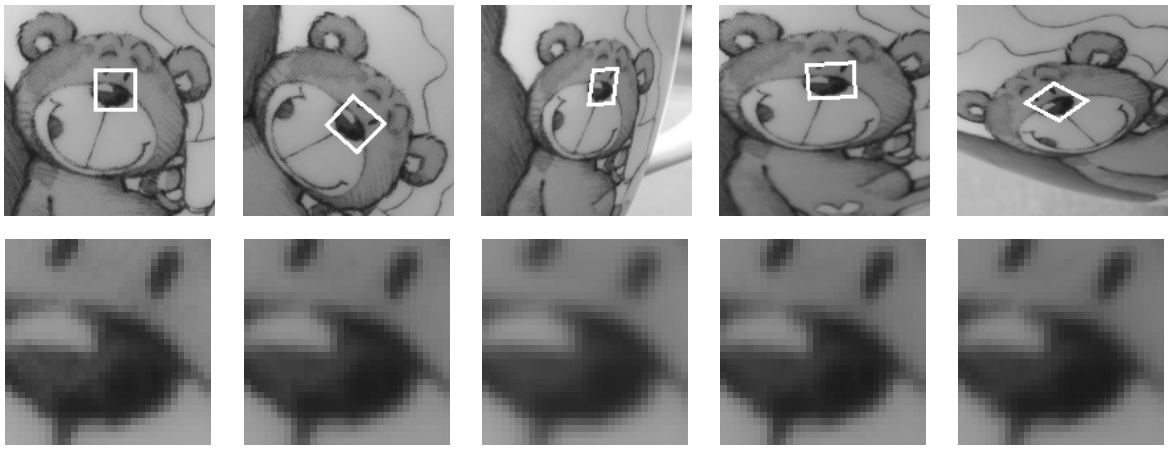


Figure 3.6: The images in the top row illustrate how the appearance of a feature can change when the scene object moves relative to the camera. Although the feature window lies on a curved surface and the camera performs a perspective projection, the affine motion model is sufficiently accurate for describing the motion of the feature window. This finding is confirmed by the images in the bottom row, which contain the feature windows that were reconstructed from the respective input images with the computed motion parameters.

measuring feature dissimilarity. In order to consolidate the different motion models in our feature point tracking system, we work with the parameterized warp function $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\phi})$. It represents either a translation

$$\boldsymbol{g}_t(\boldsymbol{x}, \boldsymbol{\phi}_t) = \boldsymbol{x} + \boldsymbol{d}\,, \quad \boldsymbol{d} \in \mathbb{R}^2, \quad \boldsymbol{\phi}_t = (\, d_1,\, d_2\,)^{\mathrm{T}}$$

or an affine motion

$$\begin{aligned} \boldsymbol{g}_a(\boldsymbol{x}, \boldsymbol{\phi}_a) &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{d}\,, \quad \boldsymbol{A} \in \mathbb{R}^{2\times 2}, \quad \boldsymbol{d} \in \mathbb{R}^2, \\ \boldsymbol{\phi}_a &= (\, a_{11}-1,\, a_{12},\, a_{21},\, a_{22}-1,\, d_1,\, d_2\,)^{\mathrm{T}}. \end{aligned} \tag{3.13}$$

In both cases, the parameter vector is defined such that the zero vector yields the identity transformation. We can now rewrite the error function of the feature tracker as

$$\epsilon_{\mathrm{faa}}(\Delta\boldsymbol{\phi}) = \sum_{\boldsymbol{x}\in\mathcal{W}} \left(f_r(\boldsymbol{x}) - f_c(\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\phi} + \Delta\boldsymbol{\phi}))\right)^2. \tag{3.14}$$

The update vector $\Delta\boldsymbol{\phi}$ that minimizes the error function $\epsilon_{\mathrm{faa}}(\Delta\boldsymbol{\phi})$ can be derived with the same approach that was applied in (3.3) - (3.7). First, the error function is transformed with a first-order Taylor expansion on $f_c(\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\phi}))$. Then, the derivative of the revised error function is set to zero and the resulting system is solved for $\Delta\boldsymbol{\phi}$. After the introduction of the vector

$$\boldsymbol{h}_{\mathrm{faa}}(\boldsymbol{x}) = \left(\left(\boldsymbol{\nabla} f_c(\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\phi}))\right)^{\mathrm{T}} \frac{\partial \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\phi})}{\partial \boldsymbol{\phi}} \right)^{\mathrm{T}} \tag{3.15}$$

and the matrix

$$\boldsymbol{H}_{\text{faa}} = \sum_{\boldsymbol{x} \in \mathcal{W}} \boldsymbol{h}_{\text{faa}}(\boldsymbol{x})\, \boldsymbol{h}_{\text{faa}}(\boldsymbol{x})^{\mathrm{T}}\,, \tag{3.16}$$

the update vector $\Delta\boldsymbol{\phi}$ can be written as

$$\Delta\boldsymbol{\phi} = \boldsymbol{H}_{\text{faa}}^{-1} \sum_{\boldsymbol{x} \in \mathcal{W}} \boldsymbol{h}_{\text{faa}}(\boldsymbol{x})\, (f_r(\boldsymbol{x}) - f_c(\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\phi})))\,. \tag{3.17}$$

When the parameterized warp function $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\phi})$ represents a translation, the vector $\boldsymbol{h}_{\text{faa}}(\boldsymbol{x})$ is defined as

$$\begin{aligned} \boldsymbol{h}_t(\boldsymbol{x}) &= \left((\nabla f_c(\boldsymbol{g}_t(\boldsymbol{x}, \boldsymbol{\phi}_t)))^{\mathrm{T}} \frac{\partial \boldsymbol{g}_t(\boldsymbol{x}, \boldsymbol{\phi}_t)}{\partial \boldsymbol{\phi}_t} \right)^{\mathrm{T}} \\ &= \left((\nabla f_c(\boldsymbol{x} + \boldsymbol{d}))^{\mathrm{T}} \frac{\partial (\boldsymbol{x} + \boldsymbol{d})}{\partial \boldsymbol{d}} \right)^{\mathrm{T}} \\ &= \left((\nabla f_c(\boldsymbol{x} + \boldsymbol{d}))^{\mathrm{T}} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right)^{\mathrm{T}}. \end{aligned} \tag{3.18}$$

As can easily be seen, this notation yields the same results for the computation of the motion parameter update vector $\Delta\boldsymbol{\phi}$ as (3.6). On the other hand, for an affine motion, the vector $\boldsymbol{h}_{\text{faa}}(\boldsymbol{x})$ is given by

$$\begin{aligned} \boldsymbol{h}_a(\boldsymbol{x}) &= \left((\nabla f_c(\boldsymbol{g}_a(\boldsymbol{x}, \boldsymbol{\phi}_a)))^{\mathrm{T}} \frac{\partial \boldsymbol{g}_a(\boldsymbol{x}, \boldsymbol{\phi}_a)}{\partial \boldsymbol{\phi}_a} \right)^{\mathrm{T}} \\ &= \left((\nabla f_c(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{d}))^{\mathrm{T}} \frac{\partial (\boldsymbol{A}\boldsymbol{x} + \boldsymbol{d})}{\partial \boldsymbol{\phi}_a} \right)^{\mathrm{T}} \\ &= \left((\nabla f_c(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{d}))^{\mathrm{T}} \begin{pmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{pmatrix} \right)^{\mathrm{T}}. \end{aligned} \tag{3.19}$$

Consequently, the matrix $\boldsymbol{H}_{\text{faa}}$ for solving the equation in (3.17) is a 6×6 matrix. In accordance with Subsection 3.2.1, the general rule for updating the motion parameter vector $\boldsymbol{\phi}$ between two iterations of the motion estimation is

$$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \Delta\boldsymbol{\phi}\,. \tag{3.20}$$

The second row of images in Figure 3.6 illustrates the reconstructions of a tracked feature window for the affine motion model. It is obvious that only the affine motion model is able to cope with the types of motion that occur over longer periods of time, e. g., rotation, scaling, and shearing. Shi and Tomasi estimate the affine motion from the first frame to the current frame of a feature trail and reject outliers by monitoring the corresponding error function. Different uses for the affine motion estimation and more elaborate techniques for outlier rejection will be presented in the next section.

3.3 The Feature Point Tracking System

This section details our contributions in the area of feature point tracking. To this end, we present several independent enhancements to the Kanade-Lucas-Tomasi feature point tracker in the five subsections 3.3.2 to 3.3.6. Furthermore, we describe the cooperation of these enhancements in our feature point tracking system in Subsection 3.3.7. We start this section with an overview of the effects of the proposed enhancements on the performance of the feature point tracking system with respect to the criteria defined in Subsection 3.1.1.

3.3.1 Overview

As our feature point tracking system is used by a number of components of the VAMPIRE system, like self-localization of the augmented reality gear, building image mosaics, and object tracking, its main goal is to be sufficiently versatile to satisfy the diverse requirements. In order to achieve this versatility, we have taken care that the enhancements presented in the following subsections are both independent and optional. This design decision is especially important for enhancements that yield both positive and negative effects with respect to the four performance criteria defined in Subsection 3.1.1.

The associated trade-offs of the proposed extensions concerning the defined performance criteria are discussed in their respective subsection. In addition to that, the following general observations can be made:

- The basin of convergence of the standard Kanade-Lucas-Tomasi tracker is approximately half the size of the feature windows, which is not sufficient for many applications. The hierarchical translation estimation described in Subsection 3.3.5 considerably increases the basin of convergence of the translation estimation.

- The computational efficiency of the standard KLT tracker is already very good. The enhancements in Subsections 3.3.2 and 3.3.3 further improve its efficiency without interfering with the other performance criteria. In addition to that, the efficient implementation of all algorithms of the tracking system is essential to obtain real-time capabilities.

- Although the requirements on the accuracy of the computed feature positions vary widely for the different applications of our tracking system, the accuracy of the standard tracker is usually sufficient. With the help of the feature drift prevention approach presented in Subsection 3.3.3, the tracker gains the ability to maintain its subpixel accuracy in the case of long video sequences.

- The low robustness against adverse conditions in video sequences of real-world scenes is a weak point of the standard Kanade-Lucas-Tomasi tracker.

First and foremost, the intensity equalization approaches discussed in Subsection 3.3.4 greatly alleviate this weakness. However, they also result in a moderately reduced basin of convergence. The approaches for outlier rejection described in Subsection 3.3.6 indirectly increase the robustness of the tracker by rejecting erroneous feature positions. These approaches have no measurable impact on the other performance criteria.

3.3.2 Efficient Feature Selection

In the literature on the standard Kanade-Lucas-Tomasi tracker, feature selection is only performed in the first frame of the video sequence [Tom91, Shi94]. Thus, the efficiency of this operation has never been a pressing issue. In our work, the tracking system is also used to process long video sequences, where some features are invariably lost due to one of the following reasons:

- Feature points are occluded by other objects.
- Feature points leave the field of view of the camera.
- Feature points are discarded by outlier rejection.

Therefore, in order to replace lost feature points regularly, feature selection also has to be performed in intermediate frames. As a consequence, the employed feature selection strategy has to be very efficient to retain the real-time performance of the tracking system.

The input data for our feature selection strategy is an interest image, which stores a feature quality measure for every pixel. In our tracking system, the feature quality measure is computed according to either (3.11) or (3.12). In both cases, the computation of the interest image requires two derivative images, which are generated with the Sobel operator as detailed in Subsection 3.2.1. There are three design goals for our feature selection strategy. It has to find the best remaining features in the interest image, avoid the selection of overlapping features, and operate in a highly efficient way.

In order to achieve the desired goals, our feature selection strategy uses the hierarchical search structure illustrated in Figure 3.7. The pixels of the interest image are grouped into logical blocks, which have a typical size of 8×8 pixels. These blocks are hierarchically grouped into higher-level logical blocks, until the interest image is completely covered by one block on the highest level. For each block, the position and the quality measure of the best feature are stored. This operation requires slightly more computation time than scanning the entire interest image for the best feature once. When all blocks are initialized, the best remaining feature is stored in the top-level block.

After the selection of the best remaining feature, its quality measure is set to zero in the interest image. When the selection of overlapping features is not desired, the

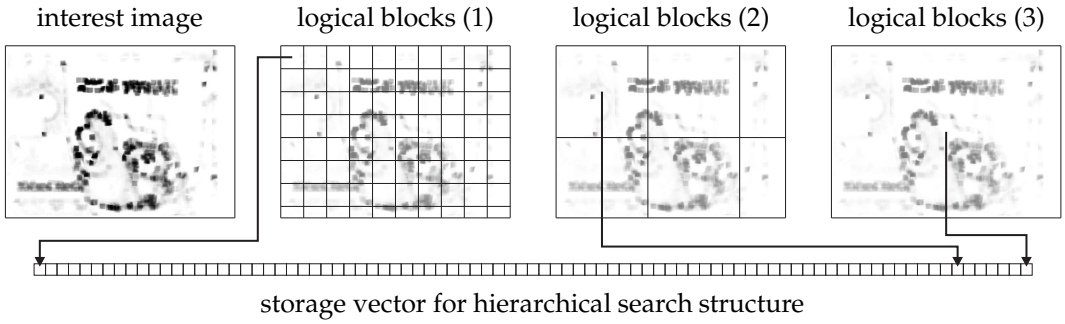


Figure 3.7: The hierarchical search structure partitions the interest image into several levels of logical blocks. The only physical representation of each block is an element in the storage vector, which contains the position and the quality measure of the best feature in this block. Metadata like the address of a block in the storage vector or the set of its elements can be directly computed from the resolution of the image, the size of the blocks, and the position of the block.

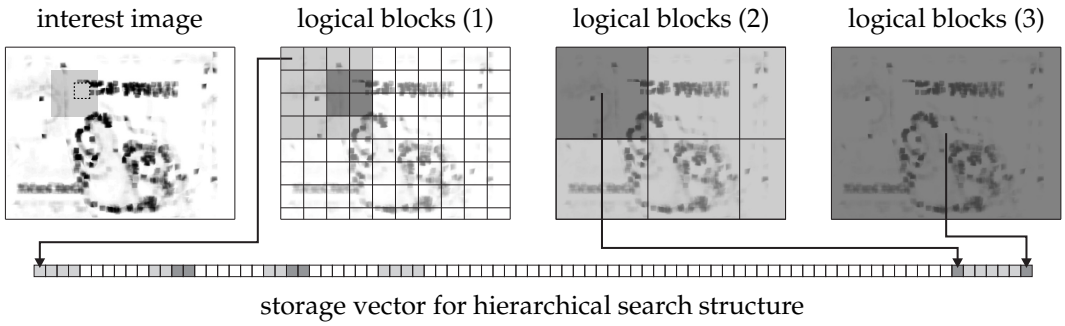


Figure 3.8: Exemplary update of the hierarchical search structure. A dashed rectangle marks the values around the selected feature that are set to zero to prevent overlapping feature windows. Pixels and logical blocks that have to be examined for the update of blocks on the next higher level are marked in light gray, whereas blocks that are potentially updated are shown in dark gray.

quality measure of all pixels within the distance $\delta_{\text{ftr\_dist}}$ from the selected feature has to be set to zero. As a side note, it is also important to perform this action for all currently active features when the feature selection is used on intermediate frames in a video sequence. In this case, it is most efficient to edit the interest image before the hierarchical search structure is initialized.

In order to find the next best remaining feature, an update of the hierarchical search structure is necessary. As the interest image has only been changed in a known local area around the last selected feature, only a small number of blocks have to be examined and updated. This process is demonstrated in Figure 3.8, where examined pixels and blocks are shown in light gray and updated blocks are shown in dark gray. It can easily be seen that only a small fraction of the pixels and blocks has to be examined during an update of the hierarchical search structure. All

in all, feature selection with the presented strategy is much faster than the algorithm proposed in [Tom91]. As it is even faster than the highly optimized computation of the interest image, further speed improvements would only show a small effect on the complete system.

3.3.3 Efficient Motion Estimation

When the basin of convergence of the tracker is increased with our robustified strategy for hierarchical translation estimation, which is described in Subsection 3.3.5, the motion parameter vector $\boldsymbol{\phi}$ has to be computed iteratively for every starting position on every level of the image pyramid for every feature point in every frame of the video sequence. Due to the very high number of invocations, the computation of the motion parameter update vector $\Delta\boldsymbol{\phi}$, which is specified in (3.17), has to be as efficient as possible to achieve real-time performance. Baker and Matthews propose an algorithm with improved efficiency in [Bak04]. Furthermore, they also present a consistent framework for several closely related feature point tracking algorithms.

In their framework, the original algorithm by Lucas and Kanade is classified as a forwards additive algorithm. The term "forwards" denotes that the motion of the feature window in the current image is estimated with respect to its position in the reference image. Furthermore, the algorithm is labelled "additive", because the motion parameter vector $\boldsymbol{\phi}$ is updated by adding the motion parameter update vector $\Delta\boldsymbol{\phi}$. Baker and Matthews propose the inverse compositional algorithm. Compared to the original algorithm, it estimates the inverse motion, i. e., the motion of the feature window in the reference image with respect to the position of the feature window in the current image. In addition to that, the motion parameter update vector $\Delta\boldsymbol{\phi}$ represents an incremental motion, which has to be composed with the current motion.

The error function of the inverse compositional algorithm is given by

$$\epsilon_{\text{ica}}(\Delta\boldsymbol{\phi}) = \sum_{\boldsymbol{x}\in\mathcal{W}} \left(f_r(\boldsymbol{g}(\boldsymbol{x},\Delta\boldsymbol{\phi})) - f_c(\boldsymbol{g}(\boldsymbol{x},\boldsymbol{\phi}))\right)^2 . \tag{3.21}$$

A first-order Taylor expansion on $f_r(\boldsymbol{g}(\boldsymbol{x},0))$ yields

$$\begin{aligned}\tilde{\epsilon}_{\text{ica}}(\Delta\boldsymbol{\phi}) &= \sum_{\boldsymbol{x}\in\mathcal{W}} \left(f_r(\boldsymbol{g}(\boldsymbol{x},0)) + \left(\boldsymbol{\nabla} f_r(\boldsymbol{g}(\boldsymbol{x},0))\right)^{\mathrm{T}} \frac{\partial \boldsymbol{g}(\boldsymbol{x},\boldsymbol{\phi})}{\partial\boldsymbol{\phi}}\Delta\boldsymbol{\phi} - f_c(\boldsymbol{g}(\boldsymbol{x},\boldsymbol{\phi}))\right)^2 \\ &= \sum_{\boldsymbol{x}\in\mathcal{W}} \left(f_r(\boldsymbol{x}) + \left(\boldsymbol{\nabla} f_r(\boldsymbol{x})\right)^{\mathrm{T}} \frac{\partial \boldsymbol{g}(\boldsymbol{x},\boldsymbol{\phi})}{\partial\boldsymbol{\phi}}\Delta\boldsymbol{\phi} - f_c(\boldsymbol{g}(\boldsymbol{x},\boldsymbol{\phi}))\right)^2 .\end{aligned} \tag{3.22}$$

When the derivative of the revised error function is set to zero, solving for the motion parameter update vector $\Delta\boldsymbol{\phi}$ results in

$$\Delta\boldsymbol{\phi} = \boldsymbol{H}_{\text{ica}}^{-1} \sum_{\boldsymbol{x}\in\mathcal{W}} \boldsymbol{h}_{\text{ica}}(\boldsymbol{x}) \left(f_c(\boldsymbol{g}(\boldsymbol{x},\boldsymbol{\phi})) - f_r(\boldsymbol{x})\right) , \tag{3.23}$$

with the vector $\boldsymbol{h}_{\text{ica}}$ defined as

$$\boldsymbol{h}_{\text{ica}}(\boldsymbol{x}) = \left((\boldsymbol{\nabla} f_r(\boldsymbol{x}))^{\mathrm{T}} \frac{\partial \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\phi})}{\partial \boldsymbol{\phi}} \right)^{\mathrm{T}}, \tag{3.24}$$

and the matrix $\boldsymbol{H}_{\text{ica}}$ given by

$$\boldsymbol{H}_{\text{ica}} = \sum_{\boldsymbol{x} \in \mathcal{W}} \boldsymbol{h}_{\text{ica}}(\boldsymbol{x})\, \boldsymbol{h}_{\text{ica}}(\boldsymbol{x})^{\mathrm{T}}. \tag{3.25}$$

The main advantage of the inverse compositional algorithm is that both the vector $\boldsymbol{h}_{\text{ica}}$ and the matrix $\boldsymbol{H}_{\text{ica}}$ are independent of the current frame and the current motion parameter vector. Consequently, they have to be computed at most once for every iterative optimization, instead of in every step of the iteration like in the forwards additive algorithm. As the matrix $\boldsymbol{H}_{\text{ica}}$ has to be inverted for computing the motion parameter update vector, the number of required matrix inversions is also considerably reduced.

The rule for updating the motion parameter vector in the inverse compositional algorithm is

$$\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\phi}) \leftarrow \boldsymbol{g}(\boldsymbol{g}(\boldsymbol{x}, \Delta\boldsymbol{\phi})^{-1}, \boldsymbol{\phi}). \tag{3.26}$$

In accordance with its introduction in Subsection 3.2.3, the parameterized warp function $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\phi})$ represents either a translation or an affine motion. In the case of translation, the new motion is given by

$$\begin{aligned} \boldsymbol{g}_t(\boldsymbol{g}_t(\boldsymbol{x}, \Delta\boldsymbol{\phi}_t)^{-1}, \boldsymbol{\phi}_t) &= \boldsymbol{x} - \Delta\boldsymbol{d} + \boldsymbol{d} \\ &= \boldsymbol{x} + (\boldsymbol{d} - \Delta\boldsymbol{d})\ . \end{aligned} \tag{3.27}$$

In the case of affine motion, the composition of the motions yields

$$\begin{aligned} \boldsymbol{g}_a(\boldsymbol{g}_a(\boldsymbol{x}, \Delta\boldsymbol{\phi}_a)^{-1}, \boldsymbol{\phi}_a) &= \boldsymbol{A} \left((\Delta\boldsymbol{A})^{-1} (\boldsymbol{x} - \Delta\boldsymbol{d}) \right) + \boldsymbol{d} \\ &= \left(\boldsymbol{A}(\Delta\boldsymbol{A})^{-1} \right) \boldsymbol{x} + \left(\boldsymbol{d} - \boldsymbol{A}(\Delta\boldsymbol{A})^{-1} \Delta\boldsymbol{d} \right)\ . \end{aligned} \tag{3.28}$$

The only disadvantage of the inverse compositional algorithm is the more costly procedure for updating affine motions, but that is easily outbalanced by the more efficient computation of the motion parameter update vector. For further details, we refer to the thorough analysis of the computational cost of both algorithms in [Bak04] and to the proof of their equivalence to first order in [Bak01].

The outstanding efficiency of the inverse compositional algorithm facilitates the integration of our robustified strategy for hierarchical translation estimation, which is described in Subsection 3.3.5, into our real-time feature point tracking system. In addition to that, it also extends the scope of affine motion estimation, which can now be continuously performed in every frame of a video sequence. The resulting structure of the motion estimation is illustrated in Figure 3.9. For any given

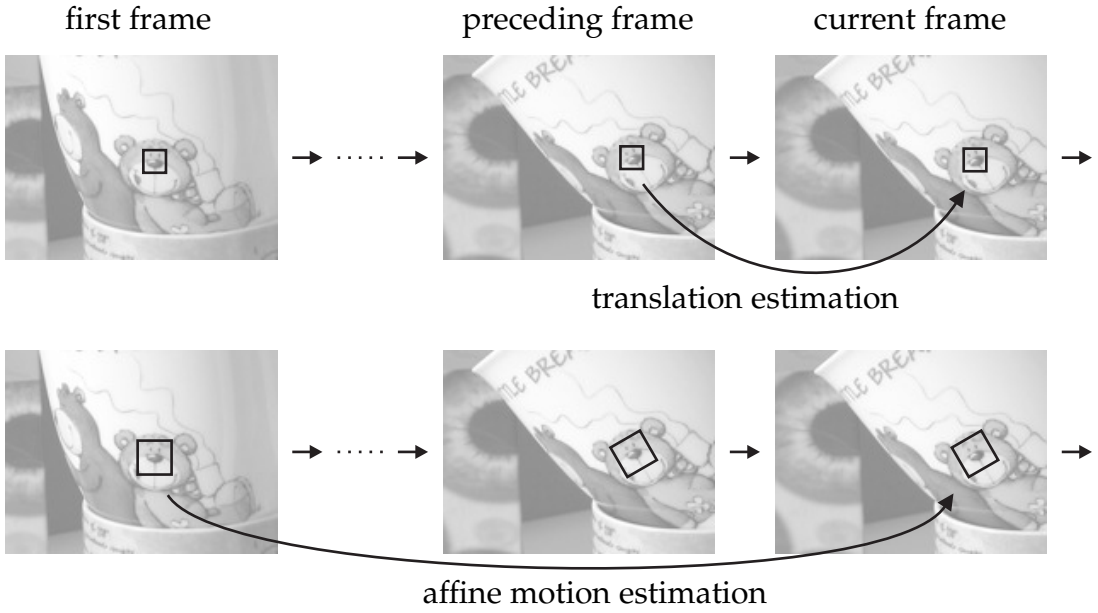


Figure 3.9: The motion estimation of our feature point tracking system comprises a translation estimation followed by an affine motion estimation. The result of the translation estimation from the preceding frame to the current frame is used to initialize the displacement vector $\boldsymbol{d}$ of the affine motion estimation from the first frame to the current frame. In contrast to this, the affine distortion matrix $\boldsymbol{A}$ is initialized with the matrix obtained by the affine motion estimation in the preceding frame.

feature and any current frame, the translation estimation is performed from the preceding frame to the current one. In order to retain a large basin of convergence for strongly distorted scene objects, the translation of a feature is always computed with a square feature window. Afterwards, the affine motion of the feature window is estimated with the first frame of the feature trail as reference frame. To this end, the affine distortion matrix $\boldsymbol{A}$ is initialized with the matrix estimated in the preceding frame, whereas the displacement vector $\boldsymbol{d}$ is initialized with the result of the translation estimation in the current frame. It is also possible to specify different window sizes $\delta_{\text{tra\_size}}$ and $\delta_{\text{aff\_size}}$ for the estimation of translation and affine motion, respectively. As the affine motion model has more motion parameters, the feature windows for its estimation are commonly larger.

Compared to the approach for outlier rejection described in Subsection 3.2.3, where the affine motion of a feature point is only estimated in the last frame of the feature trail, our approach for motion estimation offers several clear advantages. The first advantage is that outlier rejection can be performed in every frame of the video sequence. Consequently, it is possible to detect and reject outliers much earlier than in the last frame of the feature trail, i. e., before more computational resources are wasted on tracking them. Different methods for detecting outliers with the help of our motion estimation component are discussed in Subsection 3.3.6.

Another advantage of our approach for motion estimation originates from the incremental estimation of the affine distortion matrix. The gradient descent optimization algorithms for motion estimation have a basin of convergence for every estimated parameter. For translation estimation, the basin of convergence only pertains to the translation of the feature window. For affine motion estimation, the basin of convergence additionally applies to the entries of the affine distortion matrix. Consequently, the affine motion estimation only converges to the correct result when this result is not too far from the starting position. For example, it is very unlikely that a rotation of the feature window by more than 90 degrees can be reliably estimated in one step. Nevertheless, this problem often arises when the affine motion is only estimated at the end of the feature trail. With our approach, the affine motion parameters are updated in every frame, so that affine distortions that are too strong occur much less frequently.

Finally, our approach for motion estimation offers a solution to the feature drift problem, which becomes apparent in video sequences of more than 100 frames. There are several reasons why a feature window is never exactly identical in two different frames of a video sequence of a real-world scene:

- geometric distortions like rotation or non-rigid deformation,
- image noise and sampling artifacts caused by the image sensor,
- intensity changes induced by variation in illumination or exposure.

These effects are usually very small in consecutive frames. Nevertheless, they tend to introduce small errors in the frame-to-frame translation estimation. Over time, these estimation errors accumulate and invariably cause the feature windows to drift from their true position. As the feature drift problem is mainly limited to long video sequences, it has not been observed in early work on feature point tracking [Luc81, Tom91, Shi94].

Our solution to the feature drift problem is to use the translation parameters of the affine motion estimation as the final values of the feature position. In order to prevent small errors in the affine distortion matrix from negatively affecting the computed translation parameters, we center the coordinate system for the affine motion estimation on the original feature window. As the estimation is always performed with the first feature window of the feature trail as a reference, the feature drift problem is practically eliminated. It is also possible to use affine motion estimation exclusively [Jin01], but this approach abandons the much larger basin of convergence of the preceding translation estimation. Further strategies for preventing feature drift in the context of region tracking are discussed in [Mat04].

3.3.4 Integrated Intensity Equalization

It is a basic assumption of feature point tracking that the appearance of the scene objects remains approximately constant throughout the video sequence. However,

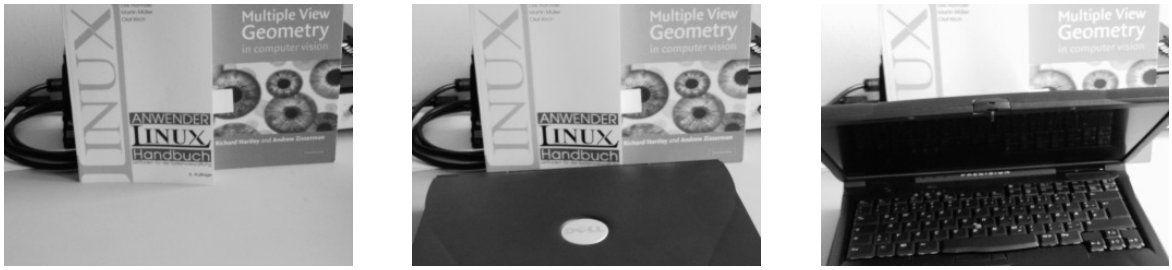


Figure 3.10: The automatic exposure correction of digital cameras is based on the assumption that a correctly exposed scene yields an average intensity value of medium gray. When a very dark object is introduced into the scene, the camera counteracts by making the whole image brighter, which affects the appearance of all scene objects.

especially the assumption of constant image intensity is often violated in video sequences of real-world scenes, which has a number of different reasons:

- Movements of the scene objects, the light sources, or the camera change the angle of incidence and the angle of irradiance of the light rays.
- Movements of the scene objects, the light sources, or the camera affect and alter the shadows in the scene.
- The lighting conditions change because light sources are added, removed, or otherwise modified.
- The automatic exposure correction of the digital camera adjusts the shutter time, the aperture size, or the electronic amplification.

As an example, the effects of automatic exposure correction of a digital camera on the appearance of the scene objects are illustrated in Figure 3.10.

The existence of various violations of the constant image intensity assumption suggests that the robustness of our feature point tracking system can be considerably improved by performing an intensity equalization on the involved feature windows. One promising approach is dynamic histogram warping, which models the intensity deviations of two images with a non-linear monotonically increasing function [Cox95]. It is mainly used for equalizing the intensities of complete images and disregards any independent local intensity deviations. Consequently, we prefer alternative approaches, which use less elaborate intensity equalization models, but are better suited for operating on corresponding feature windows.

The affine linear model $\alpha f(\boldsymbol{x}) + \beta$ is used as an intensity equalization model in many feature point tracking algorithms. It has the ability to adapt both the contrast and the brightness of the feature windows, which is a rather coarse adjustment, especially in light of the non-linear encoding of the intensity values described in Subsection 2.3.3. In practice, however, the affine linear model has proven to be more than adequate, which is mainly due to the small size of the involved feature

windows. We always compute the intensity equalization parameters α and β for the reference feature window in our feature point tracking system. In line with the handling of the motion parameters in the inverse compositional algorithm, the estimated intensity equalization parameters are inverted and applied to the current feature window for performance reasons

$$\hat{f}_c(\boldsymbol{x}) = \frac{1}{\alpha} f_c(\boldsymbol{x}) - \frac{\beta}{\alpha}. \tag{3.29}$$

There are several different methods for computing the intensity equalization parameters. The first method is based on the average value μ and the standard deviation σ of the intensity values in both feature windows. The equations for the reference feature window are

$$\mu_r = \frac{1}{|\mathcal{W}|} \sum_{\boldsymbol{x} \in \mathcal{W}} f_r(\boldsymbol{x}) \quad \text{and} \quad \sigma_r^2 = \frac{1}{|\mathcal{W}|} \sum_{\boldsymbol{x} \in \mathcal{W}} f_r^2(\boldsymbol{x}) - \mu_r^2 . \tag{3.30}$$

The average value μ_c and the standard deviation σ_c of the intensity values in the current feature window are defined analogously. It is possible to normalize the feature windows independently by subtracting the mean value and dividing by the standard deviation as proposed in [Fus99]. However, the same result can be achieved more efficiently with the affine linear model, when its parameters are computed according to

$$\alpha_{\text{die}} = \frac{\sigma_c}{\sigma_r} \quad \text{and} \quad \beta_{\text{die}} = \mu_c - \frac{\sigma_c}{\sigma_r} \mu_r , \tag{3.31}$$

where the index "die" is an abbreviation of the term "distribution normalization intensity equalization".

The second method directly estimates the intensity equalization parameters by minimizing the sum of squared differences of the intensity values in the adapted reference feature window and the current feature window

$$\epsilon_{\text{sie}}(\alpha, \beta) = \sum_{\boldsymbol{x} \in \mathcal{W}} \left(\alpha f_r(\boldsymbol{x}) + \beta - f_c(\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\phi})) \right)^2 . \tag{3.32}$$

Setting the derivative of the error function to zero and solving for the intensity equalization parameters yields

$$\begin{pmatrix} \alpha_{\text{sie}} \\ \beta_{\text{sie}} \end{pmatrix} = \left(\sum_{\boldsymbol{x} \in \mathcal{W}} \begin{pmatrix} f_r(\boldsymbol{x}) \\ 1 \end{pmatrix} \begin{pmatrix} f_r(\boldsymbol{x}) \\ 1 \end{pmatrix}^{\mathrm{T}} \right)^{-1} \left(\sum_{\boldsymbol{x} \in \mathcal{W}} \begin{pmatrix} f_r(\boldsymbol{x}) \\ 1 \end{pmatrix} f_c(\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\phi})) \right) . \tag{3.33}$$

As this method compares the intensity values at corresponding positions in the feature windows, it is more sensitive to strong misalignments of the feature windows than the first method. Both methods have in common that they are performed in alternation with the motion estimation. Consequently, the convergence rate of the combined iterative estimation is low.

A more efficient solution is achieved by combining the estimation of motion and intensity equalization into a single algorithm. This approach is described for the forwards additive algorithm in [Jin01]. In this thesis, we propose the integration of the affine linear model for intensity equalization into the inverse compositional algorithm for motion estimation. Let us define the corresponding error function as

$$\epsilon_{\text{iie}}(\Delta\boldsymbol{\phi}, \alpha, \beta) = \sum_{\boldsymbol{x} \in \mathcal{W}} \left(\alpha f_r(\boldsymbol{g}(\boldsymbol{x}, \Delta\boldsymbol{\phi})) + \beta - f_c(\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\phi}))\right)^2 . \tag{3.34}$$

A first-order Taylor expansion on $f_r(\boldsymbol{g}(\boldsymbol{x}, 0))$ yields the revised error function

$$\tilde{\epsilon}_{\text{iie}}(\Delta\boldsymbol{\phi}, \alpha, \beta) = \sum_{\boldsymbol{x} \in \mathcal{W}} \left(\alpha f_r(\boldsymbol{x}) + \alpha \left(\boldsymbol{\nabla} \boldsymbol{f}_r(\boldsymbol{x})\right)^{\mathrm{T}} \frac{\partial \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\phi})}{\partial \boldsymbol{\phi}} \Delta\boldsymbol{\phi} + \beta - f_c(\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\phi}))\right)^2 .$$

This optimization problem can be solved with the help of the intermediate parameter vector

$$\boldsymbol{\phi}_{\text{iie}} = \left(\alpha\, \Delta\boldsymbol{\phi}^{\mathrm{T}},\, \alpha\,,\, \beta\right)^{\mathrm{T}} . \tag{3.35}$$

After setting the derivative of the revised error function to zero, the solution of the optimization problem is given by

$$\boldsymbol{\phi}_{\text{iie}} = \boldsymbol{H}_{\text{iie}}^{-1} \sum_{\boldsymbol{x} \in \mathcal{W}} \boldsymbol{h}_{\text{iie}}(\boldsymbol{x})\, f_c(\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\phi}))\,, \tag{3.36}$$

where the vector $\boldsymbol{h}_{\text{iie}}$ and the matrix $\boldsymbol{H}_{\text{iie}}$ are defined as

$$\boldsymbol{h}_{\text{iie}}(\boldsymbol{x}) = \left(\left(\boldsymbol{\nabla} \boldsymbol{f}_r(\boldsymbol{x})\right)^{\mathrm{T}} \frac{\partial \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\phi})}{\partial \boldsymbol{\phi}}\,,\, f_r(\boldsymbol{x})\,,\, 1\right)^{\mathrm{T}} , \tag{3.37}$$

and

$$\boldsymbol{H}_{\text{iie}} = \sum_{\boldsymbol{x} \in \mathcal{W}} \boldsymbol{h}_{\text{iie}}(\boldsymbol{x})\, \boldsymbol{h}_{\text{iie}}^{\mathrm{T}}(\boldsymbol{x})\,. \tag{3.38}$$

The resulting algorithm inherits the improved robustness of the intensity equalization and the outstanding efficiency of the inverse composition algorithm for motion estimation. As the parameters for motion and intensity equalization are estimated together, this algorithm has a higher convergence rate than the algorithms with alternating estimation. Compared to an algorithm without intensity equalization, the number of parameters increases from two to four for translation estimation and from six to eight for affine motion estimation. The larger parameter space potentially decreases the basin of convergence with respect to the motion parameters. However, when strong intensity variations occur, an algorithm without intensity equalization is very unlikely to converge to the correct motion parameters at all.

Translation estimation is commonly performed without applying intensity equalization, because the intensity variations from one frame to the next are negligible in most video sequences. In contrast to this, the estimation of affine motion from the

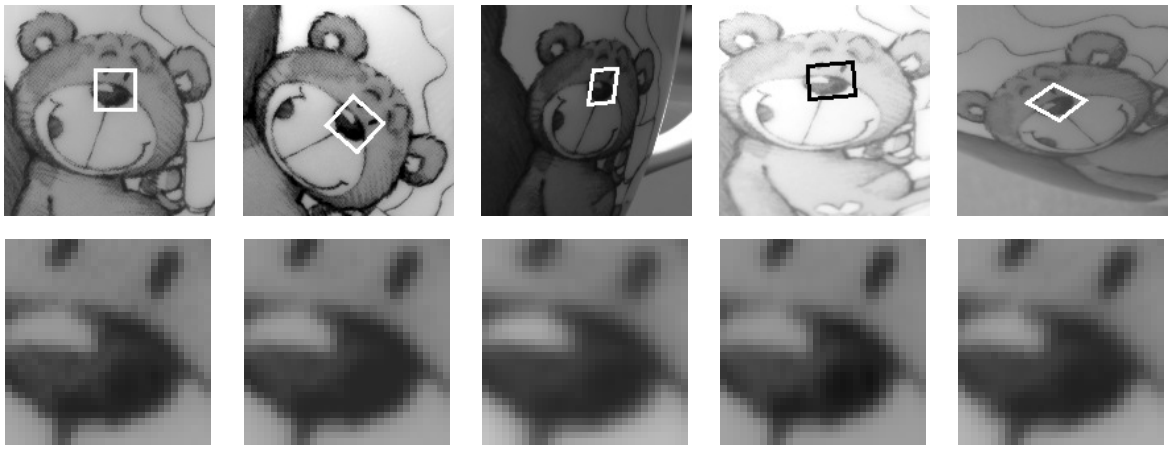


Figure 3.11: The images in the top row illustrate the effects of intensity changes on the appearance of a scene object. In order to allow an easy comparison with the results of the affine motion estimation in Figure 3.6, the changes were performed in an image editing program by modifying the contrast and the brightness of the images, as well as applying non-linear intensity transfer functions. The reconstructed feature windows in the bottom row demonstrate the performance of the integrated intensity equalization with the affine linear model in this test case.

first frame of a feature trail to the current frame often suffers from large intensity variations. Consequently, the algorithm with integrated intensity equalization is mostly used for affine motion estimation. In this case, the reduced basin of convergence with respect to the motion parameters is not critical, because the necessary updates are usually quite small. The robustness of our tracking system with integrated intensity equalization is illustrated in Figure 3.11.

3.3.5 Hierarchical Translation Estimation

The basin of convergence for successful translation estimation with the Kanade-Lucas-Tomasi tracker strongly depends on the specific appearance of each feature window. In general, our experimental evaluation suggests that displacements of approximately half the size of the feature window can be reliably estimated with the standard error function, which is defined in (3.21). The basin of convergence for translation estimation with integrated intensity equalization, as defined by the error function in (3.34), is even smaller. Thus, feature windows with a size of 11×11 pixels allow the estimation of a maximum displacement of 5 pixels, which is less than one percent of the width of an image with a size of 640×480 pixels. For many applications of our tracking system, this basin of convergence is too small.

There are several different approaches for improving the basin of convergence of the translation estimation. One solution is to increase the size of the feature windows. However, this approach yields only a small improvement for the basin of

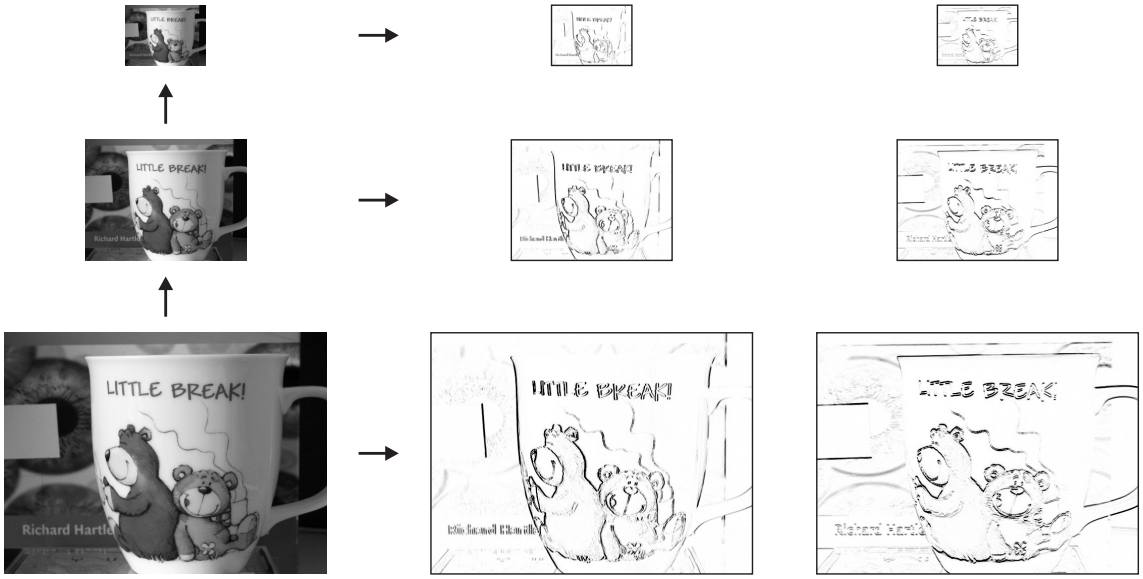


Figure 3.12: Gaussian image pyramids for one gray-level image and two derivative images. For efficiency reasons, we compute the derivative images from the gray-level images on the corresponding level in the pyramid. As the size of the additional images decreases very rapidly, the Gaussian image pyramid for an image requires at most 35% more memory than the original image.

convergence, but considerably reduces the computational efficiency of the feature point tracker. In addition to that, a larger feature window has a higher risk of straddling a depth discontinuity and is more prone to occlusions by other scene objects. Another solution, which was proposed by Lucas and Kanade in [Luc81], is to increase the basin of convergence by using a low-pass filter to smooth the image. As small details are suppressed by the filter, this approach decreases the accuracy of the estimated feature positions.

Based on the observation that a low-pass filtered image can be resampled at a lower resolution without loss of information, it is possible to devise an approach that yields a much larger basin of convergence. In our tracking system, the input image is first smoothed with a 5×5 Gaussian filter

$$\boldsymbol{G} = \frac{1}{16}\begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix} * \frac{1}{16}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{3.39}$$

and then downsampled by a factor of two along both coordinates axes by removing all odd rows and columns of the image. The result of the recursive application of this procedure is a Gaussian image pyramid, which is illustrated in Figure 3.12.

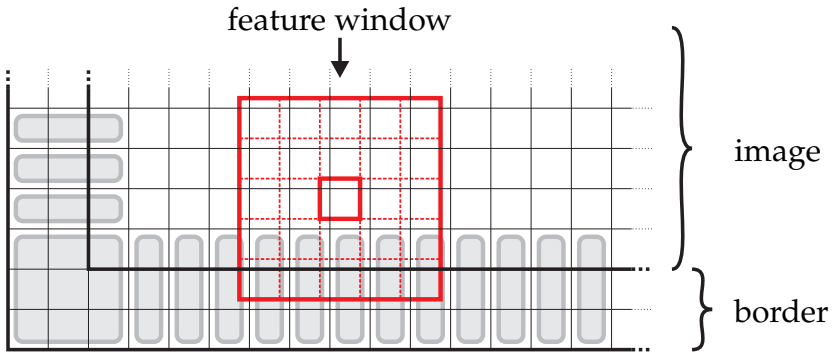


Figure 3.13: The images in our image pyramid are extended with a border in order to improve the efficiency of the translation estimation. This approach ensures that every pixel of the feature window lies inside the extended image, as long as the central pixel of the feature window lies inside the original image. For example, the 5×5 feature window in this figure requires an additional border of two pixels. During the initialization of the border, the intensity value of every pixel in the additional border is copied from its nearest neighbor in the image. As a consequence, the intensity values of the pixels in any gray rounded box are identical.

For feature point tracking, image pyramids with two to five levels are used. As an example, the size of the image on the fourth level of the pyramid is 80×60 pixels for an input image with a size of 640×480 pixel.

The standard way of applying the Gaussian image pyramid is to start the hierarchical translation estimation at the highest level of the pyramid and to iteratively refine the estimated feature translation on the lower levels of the pyramid. When the physical feature window size in pixels is held constant, the effective feature window size doubles from one level to the next higher level. On the one hand, the effective feature window size is large for the initial estimation of the translation at the highest level of the pyramid, which improves the basin of convergence of the tracking algorithm. On the other hand, the refinement of the feature position at the lower levels of the pyramid helps to maintain the high accuracy of the standard algorithm. This approach is detailed by Bouguet in [Bou00]. As our experimental evaluation suggests that every level of the image pyramid approximately doubles the basin of convergence, the additional computation time required for computing the Gaussian image pyramid represents a negligible drawback.

When the size of the feature windows is held constant on all levels of the image pyramid, the image borders have to be treated with special attention. For a standard input image with a size of 640×480 pixels, the requirement that a feature window with a size of 11×11 pixels has to lie completely inside all images of a four level image pyramid results in a border of $5 \times 2^3 - 5 = 35$ pixels where no features can be tracked. This problem can be eliminated by requiring that only the center pixel of a feature window has to lie inside the current image. Bouguet

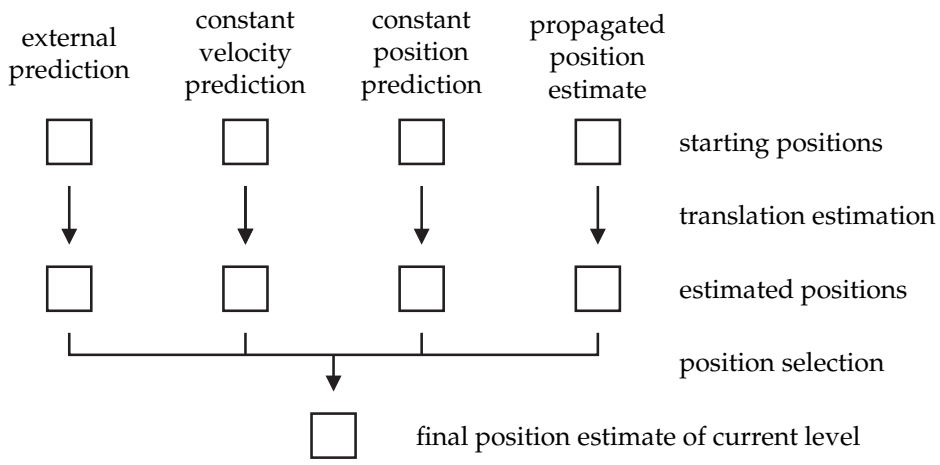


Figure 3.14: This figure illustrates our robustified result propagation strategy for hierarchical translation estimation on one level of the Gaussian image pyramid. The three starting positions on the left are optional, but one of them has to be present at the highest level of the pyramid, because the fourth starting position is not available there. A standard hierarchical translation estimation can be simulated by using one of the optional starting positions at the highest level and only the propagated position estimate on all other levels.

proposes to exclude those pixels of the feature window that lie outside the current image [Bou00]. For this approach, the location of every pixel has to be checked in every computation. In contrast to this, we extend the images of the image pyramid with an additional border. As illustrated in Figure 3.13, every pixel in the additional border has the same intensity value as the nearest pixel inside the image. Consequently, all computations can be done without explicitly checking the position of every pixel.

Estimation errors on the higher levels of the Gaussian image pyramid are a common problem of the hierarchical translation estimation. One reason is that the implicit blurring of the input image can erase small details and possibly make whole features disappear. Another reason is that the large effective feature window size increases the risk of drastic appearance changes, which has already been detailed above. In both cases, the estimation error can become so large that it cannot be recovered on the lower levels of the pyramid. However, for small translations, simply starting the estimation on one of the lower levels might have been sufficient to estimate the correct translation. In the following, we put forward a strategy that eliminates the described estimation errors.

Figure 3.14 illustrates the process of translation estimation on one level of the Gaussian image pyramid with our robustified result propagation strategy. Basically, the translation estimation is run with a number of different starting positions, and afterwards one translation vector is chosen as the final result for the current level. Thus, the estimated translation is given by the final result on the lowest level of the pyramid. The quality measure for selecting the best translation vector on

one level of the pyramid is defined by the final value of the error function of the tracking algorithm.

The four possible starting positions for translation estimation are shown in Figure 3.14. The external prediction is only available in special applications, where model knowledge allows to compute the approximate feature position in advance. The prediction for constant feature velocity can be computed internally, as long as the feature position is known in the two preceding frames. When the velocity of the feature points is indeed more or less constant, this prediction also increases the basin of convergence of the feature point tracker. The third starting position is given by the position of the feature in the preceding frame. Consequently, it is especially useful for small or erratic motions of the feature points, and whenever none of the other predictions are available. The fourth starting position is the propagated result of the hierarchical estimation, which is obviously available on any level other than the highest one.

Although the translation estimation is performed with a very efficient algorithm, computing it several times on every level of the image pyramid for every feature point can have a negative impact on the computational efficiency of the tracker. Consequently, our result propagation strategy can be configured for maximum robustness or for maximum computation speed. When computation speed is not an issue, the predictions for constant velocity and for constant position can be used in parallel. It is also possible to activate only one of the predictions in order to improve the computation speed of the tracker. For maximum computation speed, the use of the predicted starting positions can be limited to the higher levels of the image pyramid, as the probability for large estimation errors decreases on the lower levels.

3.3.6 Revised Outlier Rejection

The standard approach for outlier rejection by Shi and Tomasi has already been discussed in Subsection 3.2.3. Corresponding feature windows in the first frame and the current frame of a feature trail are aligned with the help of an affine motion model. Subsequently, the dissimilarity between corresponding feature windows as defined by the error function in (3.21) is monitored. This approach also forms the basis of the outlier rejection in our feature point tracking system. As a first enhancement to the standard approach, we improve the accuracy of the affine motion estimation and the significance of the associated dissimilarity measure by integrating the three approaches for intensity equalization described in Subsection 3.3.4.

In order to reject features based on their dissimilarity values, a threshold on the maximum dissimilarity value has to be specified. Shi and Tomasi work with a user-defined threshold that is determined after manual inspection of the computed dissimilarity values [Shi94]. In contrast to this, the threshold is computed automatically with the X84 rejection rule in [Fus99]. The rule requires that the inliers have a Gaussian distribution and determines the threshold with the help of the median

absolute deviation. Our experiments show that the integrated intensity equalization in our tracking system considerably widens the gap between the dissimilarity values of correctly tracked features and outliers, because most violations of the assumption of constant image intensity are eliminated. Consequently, we rely on a predefined value $\theta_{\text{out\_mse}}$ for the threshold on the maximum dissimilarity value.

Both [Shi94] and [Fus99] perform outlier rejection after processing all frames of a video sequence. This approach is perfectly valid for tracking in short video sequences, but has some problems when working with longer video sequences, where features are continually lost and lost features are regularly replaced with new features. In order to check all tracked features, the outlier rejection component has to consider more than the first and the last frame of the video sequence. In addition to that, rejecting outliers as soon as possible reduces the computation time wasted on tracking them and allows new features to replace them. Therefore, our tracking system performs outlier rejection after every frame of the video sequence.

Another interesting idea for outlier rejection is put forward in [Jin01]. As the scale of the current feature window is determined by the affine distortion matrix, it may become much smaller or much larger than the reference feature window. Both cases are strong indications for an estimation error, and thus an outlier. In addition to that, when the current feature window becomes too small, the information it contains is not sufficient for successful motion estimation. Consequently, a minimum and a maximum threshold on the relative scale of the reference feature window can be used for outlier rejection.

We extend this approach by monitoring the singular values of the affine distortion matrix, which represent the scale of the feature window along the principal axes of the affine transformation. Thus, a feature point is rejected when at least one of the singular values of its affine distortion matrix lies outside the range $[1/\theta_{\text{out\_sv}}, \theta_{\text{out\_sv}}]$. Our default value for $\theta_{\text{out\_sv}}$ is four. In contrast to the original approach, this allows us to reject features that are extremely distorted, but have approximately retained their original area. Our experiments confirm that high distortion is a strong sign for an erroneous feature window.

Although the outlier rejection in our feature point tracking system helps to increase the quality of the computed feature trails, some outliers simply cannot be detected by data-driven approaches. Fortunately, the number of remaining outliers is rather small in most applications. As an important consequence, all algorithms that use feature trails as input data must always be able to cope with a certain number of outliers.

3.3.7 System Structure

In this subsection, we provide a detailed description of the structure of our feature point tracking system, which is illustrated in Figure 3.15. In addition to that, we explain the application of the enhancements presented in the preceding subsections.

FOR every image c in the image sequence		
apply Gaussian filter of size $\delta_{\text{fil\_size}}$ to reduce noise (cf. text below)		
compute Gaussian image pyramid with $\delta_{\text{tra\_lvl}}$ levels (cf. Subs. 3.3.5)		
FOR every feature point in the preceding image		
perform (hierarchical) translation estimation (cf. Subs. 3.3.5) - with the inverse compositional algorithm (cf. Subs. 3.3.3) - with optional intensity equalization (cf. Subs. 3.3.4) - or with block matching (cf. Subs. 3.4.2)		
perform affine motion estimation - with the inverse compositional algorithm (cf. Subs. 3.3.3) - with optional intensity equalization (cf. Subs. 3.3.4)		
perform outlier rejection (cf. Subs. 3.3.6)		
IF	tracking has been successful	
THEN	store feature point in corresponding feature trail	
IF	(c mod $\delta_{\text{ftr\_step}} = 0$) AND (# of tracked feature points $< \delta_{\text{ftr\_num}}$)	
THEN	detect and select new features (cf. Subs. 3.3.2)	

Figure 3.15: The structure of our feature point tracking system

We also disclose important technical details of our feature point tracking system. Finally, we analyze its user-defined parameters, which are summarized in Table 3.1.

In our tracking system, the images of an image sequence are processed one after another in chronological order, which is the standard approach for data-driven feature point tracking. However, our tracking algorithms also provide an interface for processing images in an arbitrary order, which is utilized by the application presented in Subsection 7.2.2. As this approach depends on additional information on the position and the orientation of the camera for every captured image, it is not considered any further in this chapter.

The first operation of our feature point tracking system is to smooth the input image with a Gaussian filter of size $\delta_{\text{fil\_size}}$. This preprocessing step reduces the noise in the input images and increases the basin of convergence of the translation estimation. However, using a large window size for the filter blurs the relevant features in the images. In addition to that, the application of the filter increases the computation time. Consequently, we recommend to omit the filtering when the input images contain only a small amount of noise or when computational efficiency is of utmost importance. In all other cases, a small 3×3 filter window is usually sufficient. The second preprocessing step generates a Gaussian image pyramid containing one gray-level image and two derivative images as illustrated in Figure 3.12. The default value for the number of hierarchy levels $\delta_{\text{tra\_lvl}}$ is usually

name	type	description	def
$\delta_{\text{fil\_size}}$	preprocessing	filter window size	3
$\delta_{\text{ftr\_bord}}$	feature selection	minimum distance to image border	7
$\delta_{\text{ftr\_dist}}$	feature selection	minimum feature distance	6
$\delta_{\text{ftr\_num}}$	feature selection	maximum number of features	200
$\theta_{\text{ftr\_qual}}$	feature selection	minimum feature quality	0.1
$\delta_{\text{ftr\_size}}$	feature selection	feature window size	3
$\delta_{\text{ftr\_step}}$	feature selection	select features every $\delta_{\text{ftr\_step}}$ frames	1
$\delta_{\text{ftr\_type}}$	feature selection	feature type	$*$
$\delta_{\text{tra\_iter}}$	translation est.	maximum number of iterations	16
$\delta_{\text{tra\_lvl}}$	translation est.	number of hierarchy levels	3
$\delta_{\text{tra\_min}}$	translation est.	minimum level with multiple est.	0
$\delta_{\text{tra\_pred}}$	translation est.	motion prediction type	$*$
$\delta_{\text{tra\_size}}$	translation est.	feature window size	7
$\theta_{\text{tra\_stop}}$	translation est.	minimum translation update	0.1
$\delta_{\text{tra\_type}}$	translation est.	translation estimation type	$*$
$\delta_{\text{bm\_chk}}$	block matching	check for multiple local maxima	false
$\delta_{\text{bm\_rfn}}$	block matching	refine estimated translation	true
$\delta_{\text{bm\_rng}}$	block matching	search range	8
$\delta_{\text{aff\_iter}}$	affine motion est.	maximum number of iterations	16
$\delta_{\text{aff\_size}}$	affine motion est.	feature window size	15
$\theta_{\text{aff\_stop}}$	affine motion est.	minimum relative error reduction	0.1
$\delta_{\text{aff\_type}}$	affine motion est.	affine motion estimation type	$*$
$\theta_{\text{out\_mse}}$	outlier rejection	maximum error for affine window	225.0
$\theta_{\text{out\_sv}}$	outlier rejection	maximum distortion of affine window	4.0

Table 3.1: This table summarizes the parameters of our feature point tracking system. Its fourth column contains the default values of the parameters. Default values shown as $*$ are discussed in the text below.

a good compromise between the basin of convergence of the translation estimation and its computational efficiency.

The cooperation of translation estimation and affine motion estimation in our feature point tracking system is explained in Subsection 3.3.3 and illustrated in Figure 3.9. Furthermore, our robustified result propagation strategy for hierarchical translation estimation is presented in Subsection 3.3.5. It can be customized by specifying a minimum level for performing multiple translation estimations with the parameter $\delta_{\text{tra\_min}}$, whose default value enables multiple translation estimations on all levels of the image pyramid. For most input image sequences, multiple translation estimations can be disabled on the lowest levels of the image pyramid without reducing the success rate of the translation estimation. However, this only results in a minor increase of the computational efficiency of the tracking system. Two start-

ing positions of our translation estimation are computed from the previous feature positions by assuming either a constant position or a constant motion of the feature point. A discussion of both prediction types can be found in Subsection 3.3.5. It is possible to activate them independently with the parameter $\delta_{\text{tra\_pred}}$. By default, our tracking system uses only the constant motion prediction.

Our tracking system supports several options for the type $\delta_{\text{tra\_type}}$ of the translation estimation algorithm. The default algorithm is based on the inverse compositional approach and minimizes the error function in (3.21). The robustness of the tracking system with respect to strong intensity changes between consecutive images can be improved by using the error function defined in (3.34), which provides integrated intensity equalization. However, this approach also reduces the basin of convergence and the computational efficiency of the translation estimation. In both cases, the iterative estimation is stopped when the elements of the translation update vector $\Delta\boldsymbol{d}$ fall below the threshold $\theta_{\text{tra\_stop}}$, which is measured in pixels. As a safeguard, the parameter $\delta_{\text{tra\_iter}}$ limits the maximum number of iterations of one estimation.

In addition to the gradient descent algorithms, our tracking system offers a block matching algorithm for translation estimation. This algorithm has been developed for a high-speed feature point tracking application and is described in Subsection 3.4.2. It supports three difference measures, which are defined in equations (3.40), (3.41), and (3.42), respectively. In contrast to the gradient descent algorithms, the block matching algorithm allows the direct specification of its basin of convergence by adjusting its search range $\delta_{\text{bm\_rng}}$. As a consequence, this algorithm does not benefit from an image hierarchy, and the associated parameters $\delta_{\text{tra\_lvl}}$ and $\delta_{\text{tra\_min}}$ are not relevant for it. Further details on block matching, including the meaning of the parameters $\delta_{\text{bm\_rfn}}$ and $\delta_{\text{bm\_rng}}$, are given in Subsection 3.4.3.

There are four types $\delta_{\text{aff\_type}}$ of affine motion estimation in our tracking system. The inverse compositional approach with or without integrated intensity equalization is based on the error functions in (3.34) and (3.21), respectively. In addition to that, our tracking system provides two algorithms that perform motion estimation and intensity equalization in alternation. These algorithms estimate the intensity equalization parameters according to (3.31) and (3.33). The iterative estimation of the affine motion parameters is stopped when the relative reduction of the error function falls below $\theta_{\text{aff\_stop}}$. Furthermore, the number of iterations can be limited with the parameter $\delta_{\text{aff\_iter}}$. The parameters for outlier rejection are explained in Subsection 3.3.6.

Every $\delta_{\text{ftr\_step}}$ images, our tracking system checks if less than $\delta_{\text{ftr\_num}}$ features were successfully tracked in the current image. If this is the case, a feature detection algorithm is used to compute an interest image as described in Subsection 3.2.2. To this end, the parameter $\delta_{\text{ftr\_type}}$ activates either the Tomasi-Kanade detector (3.11) or the Harris detector (3.12). Subsequently, our efficient feature selection algorithm presented in Subsection 3.3.2 selects new features until the sum of tracked and newly selected features reaches $\delta_{\text{ftr\_num}}$, or the quality measure of all remaining

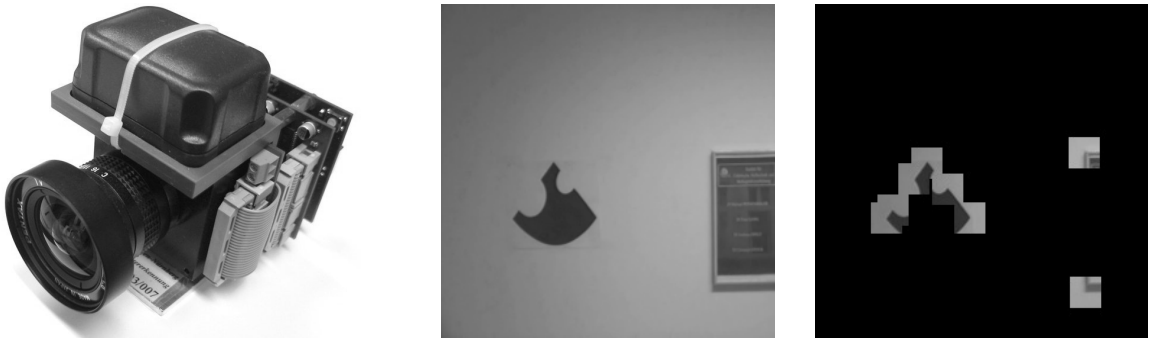


Figure 3.16: The left image shows the custom-built high-speed CMOS camera of the augmented reality gear with attached inertial sensor. The camera captures complete images like the middle image at a resolution of 1024×1024 pixels and a frame rate of 7.5 fps. It is possible to increase the frame rate by decreasing the resolution. The right image illustrates the high-speed mode of the camera, which provides subwindows of selectable size and position.

pixels in the interest image lies below $\theta_{\text{ftr\_qual}}$. The parameter $\delta_{\text{ftr\_bord}}$ defines the minimum distance of the center of the feature window to the image border. In order to obtain feature windows that lie completely inside the image, it should be at least half as large as the largest feature window, which is usually the feature window of the affine motion estimation.

3.4 High-Speed Feature Point Tracking

3.4.1 Motivation

As stated in the introduction of this thesis, the self-localization of the custom-built augmented reality gear is the most prominent application of feature point tracking in the VAMPIRE project. Its task is to estimate the position and the orientation of the user's head. This task is solved with a hybrid tracking system that combines a vision-based component with an inertial tracker [Rib02]. The required hardware is shown in the left image of Figure 3.16. It consists of a helmet-mounted CMOS camera for the vision-based component and an inertial sensor, which is attached on top of the camera with a lacing cord. In this configuration, even small rotations of the helmet can cause large movements in the image plane. In order to limit the inter-frame movements of the feature points, the custom-built high-speed CMOS camera has been designed to support very high frame rates [Mue04].

The requirements of the vision-based component differ in several aspects from most other applications of feature point tracking:

- The number of tracked feature points is very small. As six non-collinear feature points with known 3-D positions are sufficient to estimate the position

and orientation of a calibrated camera, there is no need to track more than 30 feature points at a time.

- The frame rate of the video sequence is very high. In contrast to most other digital cameras, which deliver frame rates of up to 30 frames per second, the custom-built CMOS camera supports frame rates of up to 2500 frames per second.

- Due to limits in bandwidth and processing power, the CMOS camera does not provide complete images at higher frame rates, but only subwindows of individually selectable size and position. An example of this operating mode is given in the right image of Figure 3.16.

- An external prediction of the current feature position is available. The inertial tracker estimates the position and the orientation of the camera with an update rate of more than 500 Hertz. This data can be used to select suitable subwindows and to initialize the current feature position.

Although high versatility is a very important design goal of our feature point tracking system, this list of requirements is hard to satisfy. First of all, the external prediction can be incorporated easily, thanks to the robustified result propagation strategy described in Subsection 3.3.5. However, it is difficult to achieve the described type of computational efficiency. An analysis of our feature point tracking system shows that the very low computation cost per tracked feature is balanced by the relatively high computation cost for processing the input images. This configuration is optimal for tracking a large number of features at a moderate frame rate, but problematic for the differing requirements of the self-localization of the augmented reality gear. Finally, working with subwindows as input data completely prevents the use of the hierarchical translation estimation, because the subwindows quickly become too small in higher levels of the Gaussian image pyramid. As a direct result of the inability to employ the hierarchical translation estimation, the basin of convergence of the feature point tracker becomes very small.

Considering the preceding discussion, we have to conclude that translation estimation with the Kanade-Lucas-Tomasi gradient descent algorithm is not the optimal solution for the self-localization application. Consequently, it is necessary to replace the current hierarchical translation estimation with an algorithm that entails less overhead per frame and works with subwindows, though possibly at the cost of longer computation times per feature. These properties are fulfilled by the block matching approach, which has already been briefly mentioned in Subsection 3.1.2. The following subsection contains a thorough explanation of block matching and the final subsection gives further details about the adapted tracking system.

3.4.2 Block Matching

In a sense, block matching can be considered to be a predecessor of the Kanade-Lucas-Tomasi tracking algorithm. This notion is substantiated by the fact that the section on existing techniques in the original work of Lucas and Kanade mainly deals with block matching algorithms [Luc81]. The close relationship between the Kanade-Lucas-Tomasi tracking algorithm and block matching is further demonstrated in [Dav95], which contains a proof of the mathematical equivalence of the two approaches.

As explained in Subsection 3.2.1, the task of block matching is identical to the task of the Kanade-Lucas-Tomasi tracking algorithm. Both algorithms estimate the motion of the feature window $\mathcal{W}$ from the reference frame $f_r(\boldsymbol{x})$ to the current frame $f_c(\boldsymbol{x})$. To this end, a difference measure between the reference feature window and the prospective current feature windows is minimized. In contrast to the Kanade-Lucas-Tomasi tracking algorithm, where the difference measure is expressed as an error function that is iteratively minimized by refining the initial motion estimate, a standard block matching algorithm computes the difference measure for all possible translations in a defined search region. Consequently, standard block matching algorithms forfeit the subpixel accuracy of the gradient descent tracking algorithm, but gain the ability to reliably find the optimal value of the difference measure in the defined search region.

The difference measure can be exchanged without affecting the other parts of a standard block matching algorithm. Consequently, a large variety of measures is employed for block matching. Typical difference measures are the sum of absolute differences and the sum of squared differences, which also forms the basis of the error function in the Kanade-Lucas-Tomasi tracking algorithm. In this thesis, we evaluate three more elaborate difference measures. The first one is the normalized sum of squared differences, which is given by

$$\epsilon_{\mathrm{nssd}}(\boldsymbol{d}) = \frac{\sum\limits_{\boldsymbol{x} \in \mathcal{W}} \left(f_r(\boldsymbol{x}) - f_c(\boldsymbol{x}+\boldsymbol{d})\right)^2}{\sqrt{\sum\limits_{\boldsymbol{x} \in \mathcal{W}} \left(f_r(\boldsymbol{x})\right)^2 \sum\limits_{\boldsymbol{x} \in \mathcal{W}} \left(f_c(\boldsymbol{x}+\boldsymbol{d})\right)^2}}\,. \tag{3.40}$$

It adds a normalization factor that removes the influence of the mean intensity of two similar feature windows on the difference measure, but does not cope with feature windows that require intensity equalization.

In contrast to the preceding difference measures, the following two measures are similarity measures, which means that they have to be maximized for block matching. The normalized cross correlation is invariant to changes in contrast, i. e., intensity changes of a feature window described by the linear model $\alpha f(\boldsymbol{x})$. It is

computed according to

$$\epsilon_{\text{corr}}(\boldsymbol{d}) = \frac{\sum\limits_{\boldsymbol{x} \in \mathcal{W}} f_r(\boldsymbol{x}) f_c(\boldsymbol{x}+\boldsymbol{d})}{\sqrt{\sum\limits_{\boldsymbol{x} \in \mathcal{W}} \left(f_r(\boldsymbol{x})\right)^2 \sum\limits_{\boldsymbol{x} \in \mathcal{W}} \left(f_c(\boldsymbol{x}+\boldsymbol{d})\right)^2}}. \tag{3.41}$$

The normalized correlation coefficient, which is also known as the zero mean normalized cross correlation, is invariant to changes in contrast and brightness, which are described by the affine linear model $\alpha f(\boldsymbol{x}) + \beta$. The equation of the normalized correlation coefficient is

$$\epsilon_{\text{coef}}(\boldsymbol{d}) = \frac{\sum\limits_{\boldsymbol{x} \in \mathcal{W}} \left(f_r(\boldsymbol{x}) - \bar{f}_r\right) \left(f_c(\boldsymbol{x}+\boldsymbol{d}) - \bar{f}_c\right)}{\sqrt{\sum\limits_{\boldsymbol{x} \in \mathcal{W}} \left(f_r(\boldsymbol{x}) - \bar{f}_r\right)^2 \sum\limits_{\boldsymbol{x} \in \mathcal{W}} \left(f_c(\boldsymbol{x}+\boldsymbol{d}) - \bar{f}_c\right)^2}}, \tag{3.42}$$

where $\bar{f}_r$ and $\bar{f}_c$ denote the mean intensity values of the feature windows.

In the following, we discuss the properties of block matching in comparison to the Kanade-Lucas-Tomasi tracking algorithm and with respect to the desired application in the translation estimation of a high-speed feature point tracking system. As block matching exclusively considers the intensity values of the feature windows, there is no need for the computation of derivative images. In addition to that, the basin of convergence of block matching directly depends on the user-defined size of the search region. Thus, block matching does not require a Gaussian image pyramid in order to achieve a sufficiently large basin of convergence. Without the need for derivative images and image pyramids, no processing of the input images is performed at all, except for the optional filtering of the input images. Consequently, the achievable frame rate is only limited by the computation time per feature window. Furthermore, it is easily possible to adapt the block matching algorithm to work with the subwindows captured by the CMOS camera in high-speed mode.

Another advantage of block matching is that the invariance to intensity deviations does not come at the cost of a reduced basin of convergence. However, invariant similarity measures like the normalized correlation coefficient take longer to compute than more basic difference measures. In addition to that, compared to the very efficient Kanade-Lucas-Tomasi tracking algorithm, block matching has a relatively long computation time per feature for any difference measure. Consequently, several approaches for improving the efficiency of block matching have been proposed.

The sequential similarity detection algorithm of [Bar72] stops the computation of the difference measure at a certain position as soon as it detects that the examined position is not the best one. However, this algorithm is restricted to cumulative difference measures, like the sum of absolute differences or the sum of squared differences. Another possibility is to employ a search strategy that reduces the number

of tested positions in the search region. As the search strategies take the risk of missing the best position, they are mainly used in video coding applications. A recent example of an efficient search strategy is given in [Toi02]. In order to perform efficient block matching without the limitations of the approaches discussed above, we employ a software library of optimized algorithms, which support the SIMD (single instruction / multiple data) processing capabilities of modern processors. In light of the small number of features that have to be tracked, this solution is sufficiently efficient for our desired application.

Although block matching looks like a perfect fit for our high-speed feature point tracking application, it also has some disadvantages compared to the original gradient descent motion estimation algorithm. In its current form, block matching only compares the feature windows at integer pixel positions. Consequently, it is less accurate than the replaced gradient descent algorithm. In addition to that, the affine motion estimation, which plays a central role in our tracking system, cannot be performed by the block matching algorithm. Finally, block matching has a tendency to mismatch features, especially when several similar windows lie within the search region. As these problems have to be solved in the context of the complete tracking system, the respective solutions are described in the following subsection.

3.4.3 The Adapted Tracking System

The most important step towards high-speed tracking with our tracking system is the substitution of the gradient descent translation estimation with a block matching algorithm. In order to achieve maximum performance, we propose several further steps, which can be divided into two groups. The first group consists of enhancements to the block matching algorithm, which were also integrated into the feature point tracking system described in Subsection 3.3.7. The second group consists of adaptations of implementation details of the tracking system, which do not affect its basic structure. Thus, the structure of the adapted tracking system coincides with the structure presented in Figure 3.15. We start our discussion with the second group of enhancements.

After the substitution of the translation estimation algorithm, derivative images are only required for feature detection and affine motion estimation. Due to the use of the inverse compositional algorithm, the gradient information for the affine motion estimation can be precomputed in the first frame of a feature trail. Consequently, it is sufficient to compute derivative images for input images that are used for feature detection. This is one of the main reasons for the high efficiency of our adapted feature point tracking system. In order to minimize the average computational overhead per input image, feature detection should be performed as rarely as possible.

In order to achieve very high frame rates, the high-speed mode of the CMOS camera has to be used most of the time, so that complete images are only acquired for feature detection. Unfortunately, the original idea of copying the captured sub-

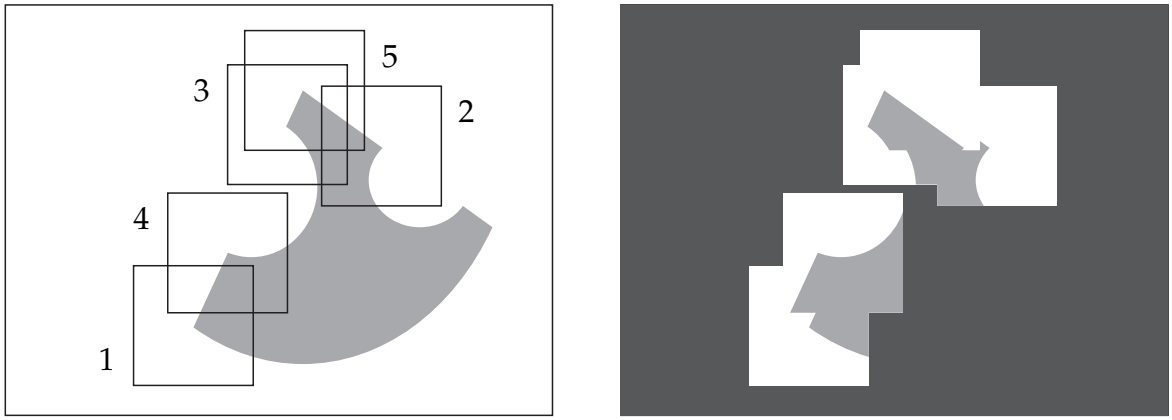


Figure 3.17: The left image shows five subwindows around feature points, which are captured by the CMOS camera in sequential order. Copying the subwindows into a common virtual input image results in unwanted artifacts when the subwindows overlap and the feature points move during the capturing process.

windows into a virtual input image has proven to be problematic. As Figure 3.17 illustrates, overlapping subwindows lead to an inconsistent composite image when the camera or the scene objects move during the capturing process. Consequently, the subwindows have to be stored separately, together with additional information about their position in the full image.

Another problem concerns the storage of the feature data. Although it is possible to inspect the tracking results after every frame, many applications query the feature data en bloc when the complete video sequence has been processed. Therefore, our standard feature point tracking system stores all feature data until the main program terminates. Due to the extremely high frame rate of the CMOS camera and the open-ended nature of the self-localization application, this strategy slowly but surely consumes the complete main memory of the computer. Consequently, the adapted tracking system only stores the data of currently tracked feature points.

After the description of the implementation details of our adapted feature point tracking system, we are finally able to address the disadvantages of stand-alone block matching discussed in the last paragraph of the preceding subsection. In our adapted tracking system, most of these disadvantages are mitigated or even eliminated by the subsequent estimation of the affine motion with a gradient descent algorithm. Obviously, the inability to estimate affine motion with our block matching algorithm is rendered irrelevant. Furthermore, the low accuracy of our block matching algorithm does not negatively affect the final feature position, because this position is refined during the affine motion estimation. However, due to the small basin of convergence of the gradient descent algorithm for affine motion estimation, a more accurate initialization of the feature position is often beneficial. Therefore, we propose to improve the accuracy of block matching by biquadratic interpolation of the difference measure, which yields subpixel accuracy at a neg-

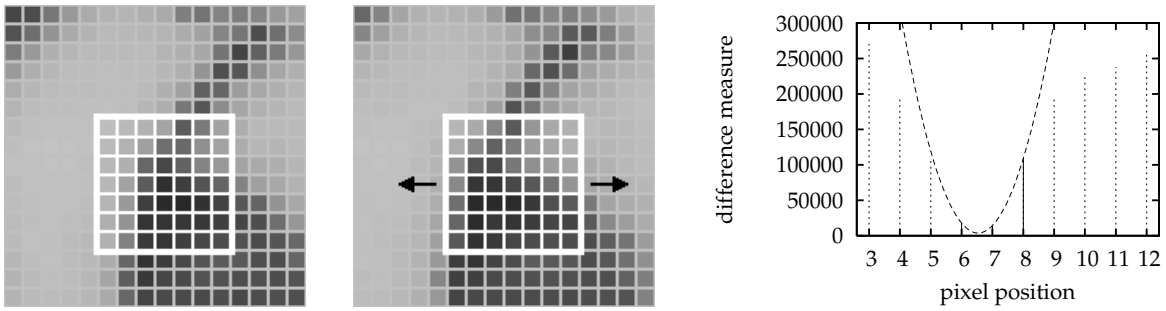


Figure 3.18: Block matching with biquadratic interpolation for achieving subpixel accuracy. In order to simplify the explanation, we only cover the one-dimensional case. The feature window in the first image lies at pixel position eight and the intensities in the second image have been shifted by 1.5 pixels to the left. The computed values of the difference measure are shown in the graph on the right. Instead of using pixel position seven, which yields the smallest difference measure, a quadratic curve is fitted through the three data points closest to the optimum. The final estimate is determined by the position of the vertex of the parabola, which lies at pixel position 6.53 in our example.

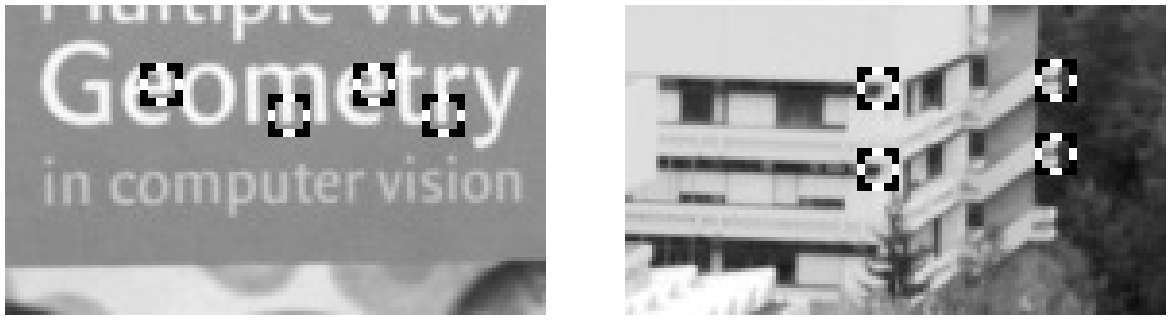


Figure 3.19: The occurrence of similar feature windows in an image can cause the block matching algorithm to produce erroneous feature trails. As shown in the left image, similar feature windows are quite common in images of any type of text. Another example for an increased probability of similar feature windows are regular man-made objects like the building in the right image.

ligible computational cost. This approach is illustrated in Figure 3.18. It can be activated with the parameter $\delta_{\text{bm\_rfn}}$.

The only remaining disadvantage of block matching is its propensity to mismatch features. This problem occurs when the feature window and a window with similar appearance both lie within the search region. Similar windows are common in images that show man-made objects, like office buildings, and artificial textures, like block letters. Figure 3.19 demonstrates the occurrence of similar windows in images of real-world scenes. The probability of a feature mismatch can be reduced by decreasing the size of the search region or by increasing the size of the feature windows, but both possibilities negatively affect more important properties of the block matching algorithm.

In contrast to block matching, gradient descent translation estimation only suffers from feature mismatch when the frame-to-frame displacement is too large, i. e., when a similar feature window is closer to the starting position than the correct feature window. In order to simulate the behavior of the replaced gradient descent algorithm, we propose the following mismatch prevention strategy. First, the difference measure is computed for all positions in the search region and its optimum is determined. Then, a dynamic threshold is computed as the mean value of the difference measure in the 4-neighborhood around the optimum. Finally, from the set of all local optima in the search window whose difference value is below the threshold, we choose the optimum that is closest to the expected position. This strategy effectively reduces the number of mismatches, especially when the current feature window is near its expected position, and requires only a small amount of additional computation time. It can be enabled with the parameter $\delta_{\text{bm\_chk}}$.

3.5 Summary

This chapter is dedicated to the problem of feature point tracking, which represents the first part of our approach for sparse 3-D reconstruction. An important part of the first section is the discussion of the performance criteria of feature point tracking, which make it possible to objectively compare different algorithms. The high importance of feature point tracking in the computer vision tool box is emphasized by the presentation of the large amount of related work. The second section describes the Kanade-Lucas-Tomasi tracker, which forms the basis of our feature point tracking system.

Our contribution to the state-of-the-art of feature point tracking is presented in the third and forth section of this chapter. The third section details the advances in areas like efficient feature selection, hierarchical translation estimation, integrated intensity equalization, and effective outlier rejection. It is concluded by a detailed explanation of the structure of our versatile feature point tracking system. Finally, the forth section contains the description of a special-purpose high-speed feature point tracking system, which was custom-made to meet the specific requirements of the self-localization of the VAMPIRE augmented reality gear.

The high versatility of our standard feature point tracking system is one of the reasons for its successful adoption in a large number of applications, some of which are presented in Chapter 7. However, in order to achieve optimum performance for a specific application, the tracking system can be tuned with more than 20 user-specified parameters. Although all parameters are listed and explained in Subsection 3.3.7, the high number of parameters prevents a comprehensive discussion of all possible combinations of parameter values. As a starting point for further information on the key parameters of our feature point tracking system, we highly recommend to check its experimental evaluation in Chapter 6.

Chapter 4

Structure and Motion Estimation

The second integral part of our approach for sparse 3-D reconstruction consists of a system for structure and motion estimation. Its main purpose is the simultaneous estimation of the scene structure and the camera motion from the feature positions computed by the feature point tracking system described in Chapter 3. The theoretical foundation for what amounts to a successful inversion of the image formation process is laid in the sections on geometric image formation and multiple view relations in Chapter 2.

As the estimation of the scene structure requires a movement of the camera relative to the scene, the problem discussed in this chapter is also known as structure from motion. It has undisputedly been the most active research area of computer vision for more than two decades. In addition to that, photogrammetry has been dedicated to determining the geometric properties of objects from one or more photographic images since the invention of modern photography in the nineteenth century. Despite the large amount of research, the automatic reconstruction of structure and motion is a very difficult problem to date.

The main goal of our structure and motion estimation system is to work as a reliable part of real-world applications, especially within the VAMPIRE project. Such applications include model-based object tracking and visualization with image-based models, which are both presented in Chapter 7. Consequently, our emphasis lies on versatility, efficiency, and robustness in a variety of common scenarios. In contrast to this, the special treatment of unusual scene configurations has been deemed less important.

The first section of this chapter contains an introduction to structure and motion estimation, as well as an overview of the related work in this problem area. In the second section, we describe the state-of-the-art algorithms that lie at the core of our structure and motion estimation system. The system itself as well as the incorporated enhancements are presented in the third section. The final section reiterates the most important aspects of this chapter.

4.1 Overview

4.1.1 Problem Statement

As we have already stated in the introduction of this chapter, the main purpose of structure and motion estimation is the simultaneous estimation of the scene structure and the camera motion from an image sequence. In this work, we rely on our feature point tracking system to process the image sequence and use its output, the 2-D coordinates of distinct 3-D feature points, as the input of our structure and motion estimation system. Other possible input types include line features, dense motion fields, and single image cues. Some approaches with alternate input types are presented in Subsection 4.1.2.

As the input of our structure and motion estimation system consists of the trails of the tracked feature points, the estimated scene structure is given by the 3-D position of these feature points in an arbitrary world coordinate system. The reconstructed camera motion is represented by the extrinsic camera parameters in the same world coordinate system, which usually coincides with the camera coordinate system of the first camera. As detailed in Subsection 2.2.3, one set of extrinsic camera parameters is defined by a rotation matrix $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ and a translation vector $\boldsymbol{t} \in \mathbb{R}^3$. Our structure and motion estimation system computes a Euclidean reconstruction of the scene, which is allowed to differ from the true reconstruction by a similarity transformation, i. e., by a rotation, a translation, and a uniform scaling. When the image sequence contains M images and the feature tracker has detected N feature points, the input consists of at most MN 2-D feature positions. Furthermore, the output comprises M sets of extrinsic camera parameters and N 3-D feature positions.

Depending on the selected approach for structure and motion estimation, the intrinsic camera parameters, which encode properties of the camera like the focal length and the position of the principal point, are either input or output. When the intrinsic camera parameters are required as input, these parameters have to be estimated before using a camera for capturing image sequences for structure and motion estimation. This process is also called camera calibration. It can be performed with an effective algorithm that requires only a simple planar calibration pattern [Zha00]. An extension to this algorithm makes it possible to estimate the radial distortion parameters of the lens [Har05].

A disadvantage of the calibrated framework is that it is not possible to process arbitrary video sequences from unknown cameras. In addition to that, changing the focal length of a camera lens during the capture of an image sequence is not supported. However, these restrictions are not relevant for the planned applications of our structure and motion estimation system. What is more, working with an uncalibrated camera has several disadvantages:

- Most approaches for uncalibrated reconstruction are not able to estimate the non-linear radial distortion parameters of the camera lens, which makes these algorithms less accurate.

- Camera calibration benefits from camera rotation, whereas structure estimation requires sufficient camera translation. It is difficult to reconcile both constraints in every input image sequence.

- Euclidean reconstruction with an uncalibrated camera is more susceptible to special configurations of the scene structure. The most prominent example is the impossibility to recover the intrinsic camera parameters in the case of a planar but otherwise unknown scene.

- Ignoring the calibration information when it is available needlessly increases the number of unknowns to be estimated. In general, this causes the estimation to become more sensitive to noise and less robust with respect to difficult configurations of scene geometry and camera motion.

As one of our main design goals is a high versatility with respect to different configurations of scene geometry and camera motion, the calibrated framework is a natural choice for our structure and motion estimation system.

The a priori knowledge of the intrinsic camera parameters is an important requirement of our structure and motion estimation system. In addition to that, our system requires the input data to satisfy several other assumptions. The most important assumption for all structure and motion estimation algorithms is that the camera moves relative to the scene, which is also implied by the alternative designation "structure from motion". Most of the work on structure and motion estimation, including the algorithms in this thesis, also rely on the fact that scene objects are rigid and do not move independently from each other. Other assumptions directly concern the tracked feature points. It is obvious that every image of the image sequence must contain at least a small number of feature points. In addition to that, the 2-D position of every feature point has to be known in at least two images, because a feature point with only one known position contains no useful information. In fact, most algorithms for structure and motion estimation benefit from feature trails that are as long as possible.

Figure 4.1 illustrates three scenarios for structure and motion estimation. The first scenario represents a configuration obtained by capturing a scene with a hand-held camera. In the given example, the depth of the complete scene is approximately as large as the distance of the scene to the camera. The camera motion mainly consists of a translation perpendicular to the optical axis of the camera and the translation distance is approximately as large as the distance of the camera to the scene objects. This configuration of scene structure and camera motion is perfectly suited for structure and motion estimation, because the resulting feature positions determine both the scene geometry and the camera motion in a reliable way. When either the depth of the scene or the camera translation distance decreases, it becomes more difficult to separate the effects of camera translation and camera rotation on the feature positions.

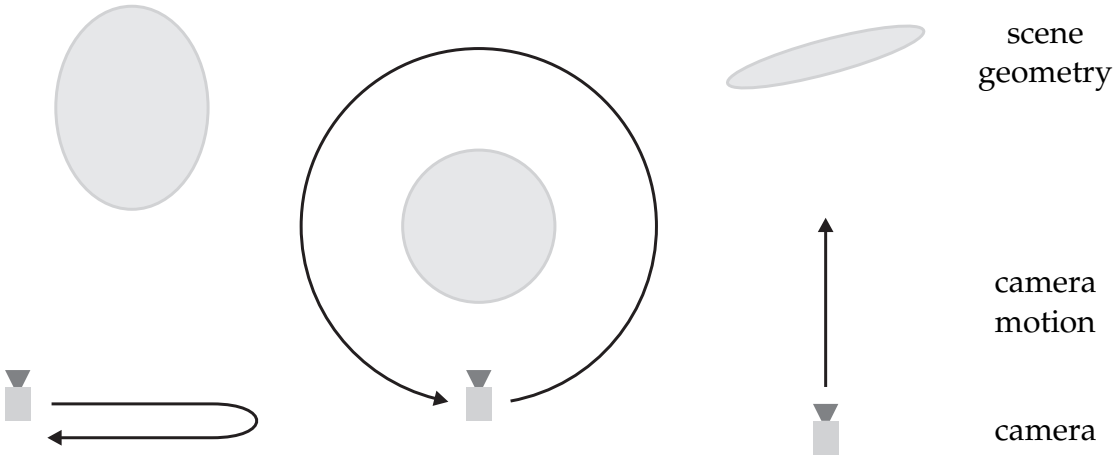


Figure 4.1: Three scenarios for structure and motion estimation demonstrate possible configurations of scene geometry and camera motion.

The second scenario is typical of a turntable setup, where an object is rotated on a motorized disc and the camera remains stationary. When the rotating disc is chosen as the frame of reference, the camera appears to revolve around the object on a circle of fixed radius. As can be seen, the relevant aspects of scene geometry and camera motion of the second scenario closely resemble those of the first scenario. However, there is an important difference in the average length of the resulting feature trails. In the first scenario, most feature points are visible from all camera positions. In contrast to this, a large fraction of the feature points on the front of the object is invisible after half a turn of the turntable in the second scenario. Consequently, the average length of the feature trails is much smaller in the second scenario. This phenomenon can cause problems for algorithms that only work with complete feature trails.

The third scenario represents one of the most difficult configurations of scene geometry and camera motion. It might occur in an augmented reality application when the user moves towards a far wall of the room. The forward motion of the camera makes it difficult to recover the scene structure accurately, because the view rays to any one feature point are close to parallel. As detailed above, the small depth of the scene impedes the accurate estimation of the camera motion, because the effects of translation and rotation are very difficult to separate in this case. Finally, the almost planar nature of the scene structure introduces additional ambiguities for many algorithms.

The categorization of different configurations of scene geometry and camera motion into scenarios is only the first step towards a systematic evaluation of different algorithms for structure and motion estimation. As the next step, we identify three performance criteria, which can be used to compare the performance of an algorithm to the requirements of its intended application:

- The computational efficiency of an algorithm for structure and motion estimation has two distinct aspects. On the one hand, the computation time for processing a given number of frames is a measure for the general efficiency of the algorithm. On the other hand, the delay between the input of the data for a frame and the output of the corresponding reconstruction defines the latency of the algorithm. While the general efficiency is relevant for all possible applications, the latency of an algorithm is only important for a subset of real-time applications, like augmented reality or robot navigation.

- The accuracy of an algorithm for structure and motion estimation can be measured independently for the scene structure and the camera motion. Incidentally, the relative importance of both results is determined by the application. For example, object modeling is interested in the structure of the object, whereas self-localization only depends on the recovery of the camera motion. There are some algorithms for structure and motion estimation that use a simplified mathematical model of the problem in order to exchange optimum accuracy for computability or higher efficiency. Even when the algorithm is optimal for perfect input data, the final accuracy of the reconstruction depends on the quality of the feature positions provided as input.

- The robustness of an algorithm describes its ability to avoid large errors under difficult conditions. In the context of structure and motion estimation, this is first and foremost a concern of dealing with outliers in the feature positions. The numerous reasons for the occurrence of outliers are detailed in Subsection 3.1.1. Another aspect of the robustness of an algorithm is its ability to work with input data representing unusual configurations of scene geometry and camera motion. Our broad definition of the robustness of an algorithm prevents the specification of a unique measure for this performance criterion. In Section 6.2, we evaluate the robustness of our structure and motion estimation system with respect to several different aspects. To this end, we analyze the accuracy of the reconstruction for input data with varying levels of difficulty.

The preceding definition of three performance criteria enables us to state the design goals for our structure and motion estimation system more precisely. Computational efficiency is an important goal of our system. However, the self-localization of the augmented reality gear, which is the only application of structure and motion estimation with stringent latency requirements in the VAMPIRE project, employs a highly specialized hybrid ego-motion estimation system. Therefore, we do not require low latency and are free to focus on higher accuracy and better robustness instead. As there are many different applications for our system, the reconstruction of scene structure and camera motion has to be performed with equal accuracy. Finally, the robustness of the employed algorithms is an important aspect for the fully automatic operation of our structure and motion estimation system. Consequently,

all relevant components of our system have to be designed to successfully cope with a reasonable amount of outliers. In this context, we expect the relative frequency of erroneous feature positions to lie below 50% in every frame. The desired versatility of our system will be achieved by employing algorithms that work for a wide range of configurations, but without resorting to algorithms that are specialized for specific scene structures or specific types of camera motions.

One important limitation of structure and motion estimation is the inability to recover the global scale of the scene. Using the equations in Chapter 2, we can easily verify that an arbitrary scaling of the scene structure and the camera motion does not change the captured images. In this chapter, the unknown scale factor has to be estimated when two partial reconstructions of the same scene have to be merged. In addition to that, we present a solution for the registration of two complete reconstructions with unknown correspondences and unknown relative scale factor in Chapter 5.

4.1.2 Related Work

Since the publication of the first computer algorithms for structure and motion estimation, for example by Longuet-Higgins in [LH81], a large variety of theoretical approaches, practical algorithms, and complete systems have been proposed in this active area of computer vision. Oliensis gives a comprehensive overview of the most important approaches for structure and motion estimation in [Oli00]. He also analyzes the differences between algorithms for calibrated and uncalibrated cameras.

In order to obtain a Euclidean reconstruction for an input sequence captured with an uncalibrated camera, the camera calibration has to be performed by the reconstruction algorithm for every input sequence. This process is also called self-calibration. Classical algorithms for self-calibration like [Fau92, Oli99b, Pol99b] compute the extrinsic and intrinsic camera parameters before the scene structure is estimated. Stratified self-calibration is a more recent approach, which first recovers the projective structure and then upgrades it to the Euclidean structure by using available constraints [Har93, Hey97, Pol99a]. Both types of self-calibration are reviewed by Fusiello in [Fus00].

Apart from the distinction between the calibrated and the uncalibrated framework, it is very instructive to categorize the algorithms for structure and motion estimation by their general structure. One large category is formed by the factorization algorithms, whose basic assumption is that the feature positions can be explained as a bilinear combination of the structure and motion parameters. Consequently, recovering these parameters involves the factorization of a measurement matrix, which contains the feature positions of the feature trails. The first factorization algorithm, which assumes an orthographic projection model, was presented by Tomasi and Kanade in [Tom92]. Poelman and Kanade introduced a factorization algorithm for paraperspective projection [Poe94]. The algorithm of Christy and Ho-

raud computes a perspective reconstruction by iteratively updating the measurement matrix to represent an affine projection [Chr96]. Sturm and Triggs proposed a factorization algorithm for perspective projection with an uncalibrated camera in [Stu96]. Iterative extensions of this algorithm are put forward in [Mah01] and [Oli07]. A general overview of factorization algorithms is given by Kanade in [Kan98].

The factorization algorithms described above share some common limitations. As batch methods, they process a complete video sequence at a time, which makes them unsuitable for real-time applications with low latency requirements. What is more, the measurement matrix must not have unknown entries. In other words, the factorization algorithms can only use complete feature trails. For the second scenario of Figure 4.1, this limitation entails the choice between discarding a large number of images and discarding a large number of incomplete feature trails, both of which reduce the available input data considerably. A final drawback of the described factorization algorithms is their inability to cope with strong outliers in the input data.

Most limitations of the original factorization algorithms have been addressed by enhanced algorithms. [Mor97] presents a sequential factorization algorithm for orthographic projection that is capable of real-time operation. The iterative approach in [Hey99], which employs subspace methods, is another example of a factorization algorithm with low latency. The problem of recovering missing features has already been treated in [Tom92]. A more efficient solution using an expectation maximization algorithm is presented in [Gue02]. Another area of interest is the incorporation of information on the uncertainty of the feature positions. Aguiar and Moura propose a factorization algorithm that attaches weights to the feature points in [Agu03], whereas Anandan and Irani model the directional uncertainty of every feature point, which allows their algorithm to work in the extreme case of normal flow data [Ana02]. Finally, [Aan02] and [Huy03] describe different approaches for the iterative correction of outliers.

Although most limitations of factorization have been addressed, there is no single algorithm that incorporates all necessary enhancements. What is more, long video sequences with highly incomplete feature trails are still a weak point of all factorization algorithms. As a consequence, we focus our attention on another category of algorithms for structure and motion estimation, the invariant-based algorithms. Their basic principle is to derive constraints on the image data by explicit algebra. In the case of two-view motion estimation, the properties of the matrix encoding the relative camera motion can be used to recover this matrix from a number of corresponding features. The most prominent two-view algorithm is the eight-point algorithm for estimating the fundamental matrix, which was presented by Hartley in [Har92]. Its performance was improved with the help of a normalization step in [Har97a], whose effects are examined with statistical methods in [Cho03].

[Har94b] details an algorithm for estimating the fundamental matrix from seven feature points, which yields at most three possible solutions. This algorithm is

compared with other techniques for estimating the fundamental matrix in [Zha98]. As the focus of this thesis lies on the calibrated framework, we are interested in invariant-based algorithms that estimate the camera motion with the help of the essential matrix. An efficient algorithm for estimating the essential matrix from five feature points was given by Nister in [Nis04b]. It is conceptually similar to the seven-point algorithm by Hartley and yields up to ten possible solutions. As it is one of the key algorithms of our structure and motion estimation system, we discuss it thoroughly in Subsection 4.2.3. Another variety of invariant-based algorithms estimate the relative motion between three views with an algebraic entity that is known as the trifocal tensor. Such algorithms can be found in [Fit98] and [Nis00].

For successful structure and motion estimation, invariant-based motion estimation algorithms have to be complemented with additional methods. When the relative camera motion between at least two frames is known, it is possible to triangulate the feature points as described in Subsection 4.2.1. With the obtained 3-D feature positions, the pose of additional cameras can be estimated by the absolute pose estimation algorithms detailed in Subsection 4.2.2. There are also several different generic approaches for making invariant-based motion estimation algorithms more robust to outliers in the input data. An experimental evaluation of these approaches is presented in [Tor97]. Our efforts for improving the robustness of our structure and motion estimation system are documented in Subsection 4.3.4.

Most invariant-based motion estimation algorithms are defined for a small number of features in two or three frames. Even though some of these algorithms, like the five-point algorithm, are able to incorporate a larger number of features into their computations, the resulting solution optimizes no meaningful cost function. Thus, the accuracy of these motion estimation algorithms with respect to noisy input data is suboptimal. A discussion of relevant optimization criteria used in two-view algorithms can be found in [Zha02]. The standard way for improving the accuracy of the reconstruction is to refine the results with an additional optimization algorithm. These algorithms, which constitute our third category of algorithms for structure and motion estimation, can be divided into two distinct groups. The first group consists of specialized algorithms for one problem domain and one cost function, like the exact two-view motion estimation for the directional least-squares error put forward in [Oli02]. The second group uses standard techniques for non-linear minimization and its members are generally known as bundle adjustment algorithms. An elaborate overview of bundle adjustment is given in [Tri99]. As bundle adjustment plays an important role in our structure and motion estimation system, we provide further information on this technique in Subsection 4.2.4.

At this point, it should have become apparent that a structure and motion estimation system that combines a number of different approaches to benefit from the relative advantages of the individual algorithms requires an elaborate design and a fair amount of auxiliary components. Relevant examples of such systems can be found in [Bou95, Fit98, Nis00, Gib02]. Many of these systems share a number of recurring

features, like the careful selection of key frames for the initial motion estimations, the use of a defined process for merging the resulting partial reconstructions, and the refinement of intermediate and final reconstructions with the help of optimization algorithms. Recently, real-time structure and motion estimation systems have been proposed for augmented reality [Lou05a] and robot navigation [Nis06] applications. The details of our proposed system are discussed in Section 4.3.

After the preceding overview of the most important categories of structure and motion estimation, we briefly present other related approaches in this work area. There are several specialized algorithms for dealing with difficult scene configurations. Both [Sze98] and [Pol02] are concerned with the reconstruction of a mostly planar scene, whereas [Oli99a] and [Ved07] propose algorithms for small camera translations and pure forward motion of the camera, respectively. Another class of algorithms makes use of established probabilistic methods like the extended Kalman filter [Soa98a, Soa98b] or the variable state dimension filter [McL95, McL00] to incrementally recover the unknown structure and motion. The approach in [Cha02] employs a sampling-based uncertainty representation to increase its robustness to outliers.

Other approaches for structure and motion estimation are somewhat difficult to classify. [Las98] proposes a coordinate-free approach that is based on a geometric algebra. In [Oli01], Oliensis presents algorithms that do not fit into any of the previous categories. Different problems of structure and motion estimation are solved with the quasiconvex optimization approach in [Ke05, Ke06]. A generic framework for structure and motion estimation from images of arbitrary camera types is described in [Ram06]. Another interesting problem is the reconstruction of non-rigid scenes, which either contain non-rigid surfaces or independently moving objects [Car02, Han03, Tor04, Xia06]. In some applications, different types of input data like lines [Bar05] or single image cues [Sax07] are used to provide additional information. Finally, interesting applications of structure and motion estimation, including plenoptic modeling and robot navigation, are described in [Koc99, Pol04, Dav07].

4.2 State-of-the-Art Algorithms

As we have observed in the preceding section, structure and motion estimation with an invariant-based approach requires the combination of different algorithms for several isolated subproblems. In order to separate the discussion of the state-of-the-art algorithms from the description of their combination into our structure and motion estimation system, we focus our attention on the algorithms themselves in this section. To this end, every subsection contains a short problem statement, a concise overview of possible alternatives, and a thorough description of the selected algorithms.

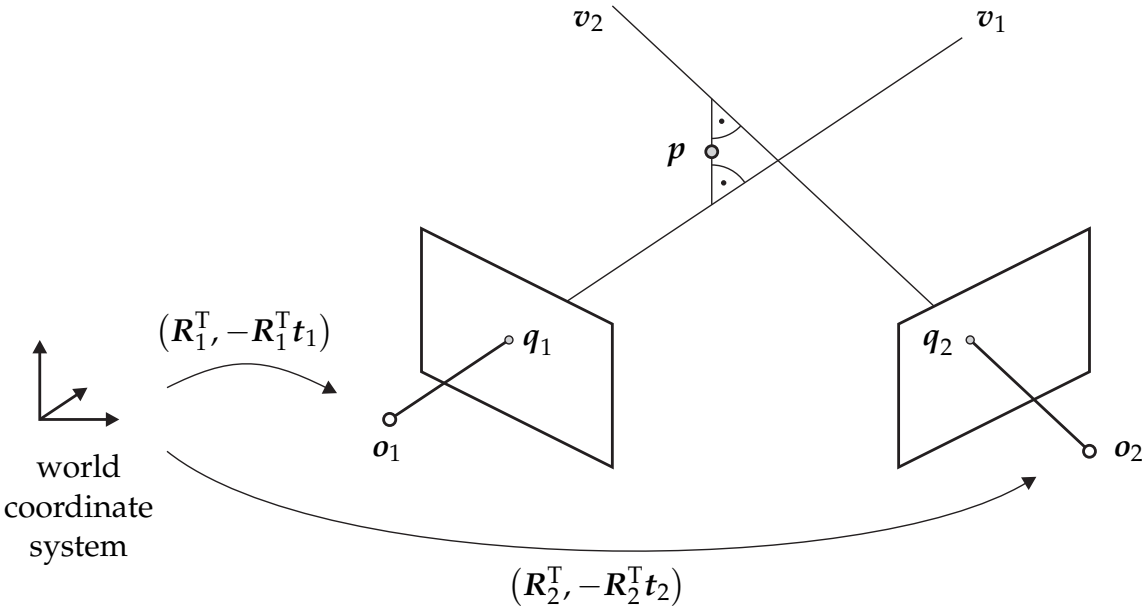


Figure 4.2: The triangulation of a feature point with non-intersecting view rays requires a suitable approximation to the point of intersection.

4.2.1 Triangulation

When the intrinsic and extrinsic camera parameters are known for at least two camera positions, it is possible to reconstruct the 3-D position of a feature point from the corresponding feature positions in image coordinates. Solutions to this problem require the intersection of two rays in space and are widely known as triangulation algorithms. As neither the camera parameters nor the feature positions are absolutely accurate in practice, there is a very high probability that the two rays will not actually intersect in space. Thus, the approach for finding a suitable approximation to the point of intersection constitutes the main distinction between existing triangulation algorithms.

In our system for structure and motion estimation, triangulation is the only approach for the initial estimation of the scene structure, but we strongly encourage the use of bundle adjustment for a subsequent optimization of the reconstruction. As a consequence, in our case, the efficiency of a potential triangulation algorithm is more important than its accuracy. A number of different triangulation algorithms are presented and evaluated in [Har97b]. In the case of Euclidean reconstruction, simple algorithms like the linear methods or the midpoint method perform quite well, despite their theoretical shortcomings. There are also more sophisticated algorithms for optimal two-view triangulation [Oli02], for optimal three-view triangulation [Ste05], and for non-iterative multi-view triangulation [Liu05]. As we are mainly interested in an efficient solution, we use the midpoint method in our estimation system.

Given the extrinsic camera parameters $(\boldsymbol{R}_1, \boldsymbol{t}_1)$ and $(\boldsymbol{R}_2, \boldsymbol{t}_2)$ of two cameras and the projections $\boldsymbol{q}_1 \in \mathbb{R}^2$ and $\boldsymbol{q}_2 \in \mathbb{R}^2$ of a feature point in image coordinates, we have to compute the position $\boldsymbol{p} \in \mathbb{R}^3$ of the feature point in world coordinates. This configuration is illustrated in Figure 4.2. The basic approach of the midpoint method is to estimate the 3-D position of the feature point as the point of minimum distance from both view rays $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. In particular, the midpoint method determines the line segment that is perpendicular to both rays and computes the midpoint of this line segment. When the view rays are defined as

$$\boldsymbol{v}_1 = \boldsymbol{o}_1 + a_1 \boldsymbol{R}_1^{\mathrm{T}} \underline{\boldsymbol{q}}_1 \,, \quad \boldsymbol{o}_1 = -\boldsymbol{R}_1^{\mathrm{T}} \boldsymbol{t}_1 \,, \quad a_1 \in \mathbb{R} \,, \tag{4.1}$$

$$\boldsymbol{v}_2 = \boldsymbol{o}_2 + a_2 \boldsymbol{R}_2^{\mathrm{T}} \underline{\boldsymbol{q}}_2 \,, \quad \boldsymbol{o}_2 = -\boldsymbol{R}_2^{\mathrm{T}} \boldsymbol{t}_2 \,, \quad a_2 \in \mathbb{R} \,, \tag{4.2}$$

the direction of the perpendicular line segment is given by

$$\boldsymbol{v}_3 = \left(\boldsymbol{R}_1^{\mathrm{T}} \underline{\boldsymbol{q}}_1 \times \boldsymbol{R}_2^{\mathrm{T}} \underline{\boldsymbol{q}}_2 \right) . \tag{4.3}$$

Building a closed vector path that leaves $\boldsymbol{o}_1$ in the direction of the view ray $\boldsymbol{v}_1$, goes through $\boldsymbol{p}$ on the perpendicular line segment $\boldsymbol{v}_3$, goes to $\boldsymbol{o}_2$ in the opposite direction of the view ray $\boldsymbol{v}_2$, and finally returns on a straight line to $\boldsymbol{o}_1$ yields

$$a_1 \boldsymbol{R}_1^{\mathrm{T}} \underline{\boldsymbol{q}}_1 + a_3 \left(\boldsymbol{R}_1^{\mathrm{T}} \underline{\boldsymbol{q}}_1 \times \boldsymbol{R}_2^{\mathrm{T}} \underline{\boldsymbol{q}}_2 \right) - a_2 \boldsymbol{R}_2^{\mathrm{T}} \underline{\boldsymbol{q}}_2 - \boldsymbol{o}_2 + \boldsymbol{o}_1 = \boldsymbol{0} \,. \tag{4.4}$$

This equation can be written as a linear system with three unknowns

$$\left(\boldsymbol{R}_1^{\mathrm{T}} \underline{\boldsymbol{q}}_1 \;\middle|\; -\boldsymbol{R}_2^{\mathrm{T}} \underline{\boldsymbol{q}}_2 \;\middle|\; \boldsymbol{R}_1^{\mathrm{T}} \underline{\boldsymbol{q}}_1 \times \boldsymbol{R}_2^{\mathrm{T}} \underline{\boldsymbol{q}}_2 \right) \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \boldsymbol{o}_2 - \boldsymbol{o}_1 \,. \tag{4.5}$$

After the solution of the linear system, the optimum values $\hat{a}_1$ and $\hat{a}_2$ are known, and the feature point $\boldsymbol{p}$ is given by

$$\boldsymbol{p} = \frac{1}{2} \left(\boldsymbol{o}_1 + \hat{a}_1 \boldsymbol{R}_1^{\mathrm{T}} \underline{\boldsymbol{q}}_1 + \boldsymbol{o}_2 + \hat{a}_2 \boldsymbol{R}_2^{\mathrm{T}} \underline{\boldsymbol{q}}_2 \right) . \tag{4.6}$$

It is obvious that the triangulation cannot succeed when the two view rays are parallel. This configuration possibly occurs when the feature point lies close to the baseline or when the feature point is very distant from both cameras.

A slightly different triangulation problem arises in the five-point algorithm discussed in Subsection 4.2.3. The camera coordinate system of the first camera coincides with the world coordinate system $(\boldsymbol{R}_1, \boldsymbol{t}_1) = (\boldsymbol{I}, \boldsymbol{0})$ and the essential matrix $\boldsymbol{E}$ of the stereo system is known. The main difference, however, is due to the properties of the five-point algorithm, which estimates the camera motion for five feature position pairs in such a way that the view rays of these pairs actually intersect in space. Consequently, it is not necessary to approximate the point of intersection,

which makes the triangulation algorithm given in [Nis04b] even more efficient than the midpoint method. The algorithm consists of the computation of the vector

$$\boldsymbol{h} = \underline{\boldsymbol{q}}_2 \times \left(\operatorname{diag}(1,1,0)\ \boldsymbol{E}\underline{\boldsymbol{q}}_1 \right) , \tag{4.7}$$

which is used to express the feature point $\underline{\boldsymbol{p}}$ in homogeneous coordinates as

$$\underline{\boldsymbol{p}} = \left(\underline{\boldsymbol{q}}_1^{\mathrm{T}} \left(\boldsymbol{t}_2^{\mathrm{T}} \boldsymbol{h} \right) \ \mid \ -\underline{\boldsymbol{q}}_1^{\mathrm{T}} \boldsymbol{R}_2^{\mathrm{T}} \boldsymbol{h} \right)^{\mathrm{T}} . \tag{4.8}$$

4.2.2 Absolute Camera Pose Estimation

Given at least three feature points, whose positions are known both in the world coordinate system and in one image coordinate system, absolute camera pose estimation is concerned with computing the extrinsic camera parameters of the corresponding camera. In the minimal case of three feature points, this problem is known to have at most four possible solutions [Har94a]. When more than three feature points are available, the solution is usually unique, unless the optical center of the camera and all feature points lie on the same twisted cubic curve [DeM95]. In this case, there are two different sets of extrinsic camera parameters that explain the feature positions equally well, and it is not possible to determine the correct camera pose based on the feature data alone.

Six similar direct solutions for the estimation of the camera pose from three feature points are analyzed in [Har94a]. A generalized algorithm for three arbitrary view rays is presented in [Nis04a]. For increased accuracy, it is beneficial to use more than three feature points in the estimation. Both iterative algorithms [DeM95, Lu02] and non-iterative algorithms [Qua99, Fio01, Ans03, MN07] were proposed to solve this problem. They differ in the number of required feature points, the accuracy of the solution, and their computational efficiency. In our structure and motion estimation system, we employ one of the three-point algorithms detailed in [Har94a] for the disambiguation of the relative camera pose estimation discussed in Subsection 4.2.3. We additionally use the POSIT algorithm of [DeM95] to estimate the camera pose when a high number of features points is available, because this algorithm is proven to be very efficient despite its iterative nature.

4.2.2.1 The Three-Point Algorithm

For our work, we have selected the algorithm of Fischler and Bolles from the three-point algorithms in [Har94a]. The geometric setup of the three-point algorithm is illustrated in Figure 4.3. Given the positions of three feature points in world coordinates and image coordinates, the algorithm has to recover the extrinsic camera parameters $(\boldsymbol{R}, \boldsymbol{t})$. To this end, we define the side lengths of the triangle of feature points as

$$a = \| \boldsymbol{p}_2 - \boldsymbol{p}_3 \| , \quad b = \| \boldsymbol{p}_1 - \boldsymbol{p}_3 \| , \quad c = \| \boldsymbol{p}_1 - \boldsymbol{p}_2 \| . \tag{4.9}$$

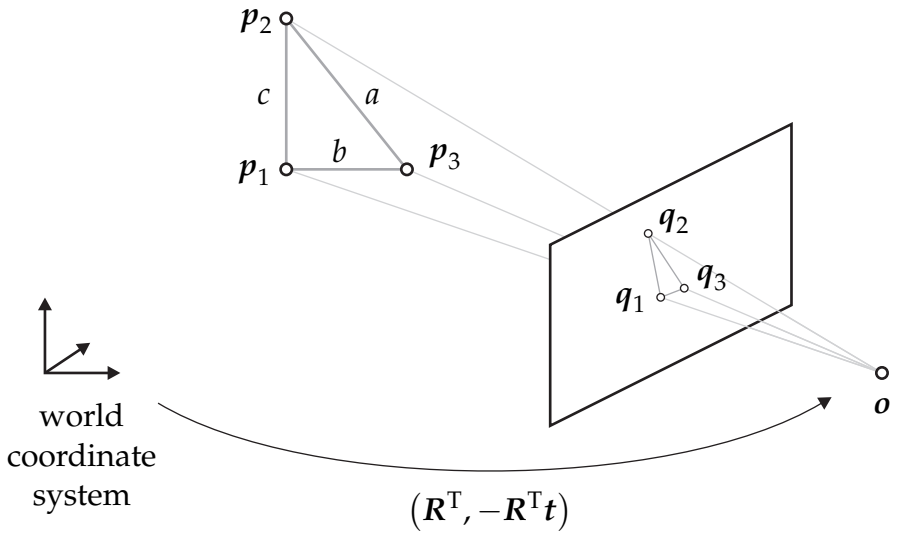


Figure 4.3: The three-point algorithm computes the pose of the camera from the positions of three feature points in world coordinates and image coordinates.

We also define the unit vectors in the direction of the view rays from the optical center $\boldsymbol{o}$ to the feature points in camera coordinates as

$$\breve{\underline{\boldsymbol{q}}}_i = \frac{\underline{\boldsymbol{q}}_i}{\|\underline{\boldsymbol{q}}_i\|} , \quad \breve{\underline{\boldsymbol{q}}}_i \in \mathbb{R}^3 , \quad i \in \{1,2,3\} \tag{4.10}$$

and the angles between the view rays as

$$\cos\alpha = \breve{\underline{\boldsymbol{q}}}_2^{\mathrm{T}} \breve{\underline{\boldsymbol{q}}}_3 , \quad \cos\beta = \breve{\underline{\boldsymbol{q}}}_1^{\mathrm{T}} \breve{\underline{\boldsymbol{q}}}_3 , \quad \cos\gamma = \breve{\underline{\boldsymbol{q}}}_1^{\mathrm{T}} \breve{\underline{\boldsymbol{q}}}_2 . \tag{4.11}$$

With these definitions, the first step of the solution is to determine the position of the feature points in the camera coordinate system, which can be expressed with the unknown distance s_i of the feature points from the optical center as

$$\boldsymbol{c}_i = s_i \breve{\underline{\boldsymbol{q}}}_i , \quad \boldsymbol{c}_i \in \mathbb{R}^3 , \quad i \in \{1,2,3\} . \tag{4.12}$$

Thus, the vectors $\boldsymbol{p}_i$ and $\boldsymbol{c}_i$ represent the same feature points in different coordinate systems. Applying the law of cosines to the three triangles formed by the optical center and each subset of two feature points yields

$$\begin{aligned} a^2 &= s_2^2 + s_3^2 - 2 s_2 s_3 \cos\alpha , \\ b^2 &= s_1^2 + s_3^2 - 2 s_1 s_3 \cos\beta , \\ c^2 &= s_1^2 + s_2^2 - 2 s_1 s_2 \cos\gamma . \end{aligned} \tag{4.13}$$

After replacing $s_2 = u s_1$ and $s_3 = v s_1$, we get

$$s_1^2 = \frac{a^2}{u^2 + v^2 - 2uv\cos\alpha} = \frac{b^2}{1 + v^2 - 2v\cos\beta} = \frac{c^2}{1 + u^2 - 2u\cos\gamma} . \tag{4.14}$$

By combining these expressions, Fischler and Bolles eliminate v and obtain the fourth order polynomial equation

$$d_4 u^4 + d_3 u^3 + d_2 u^2 + d_1 u + d_0 = 0\,, \tag{4.15}$$

$$\begin{aligned}
d_4 &= 4b^2c^2\cos^2\alpha - (a^2 - b^2 - c^2)^2\,,\\
d_3 &= -8b^2c^2\cos^2\alpha\cos\gamma - 4c^2(a^2 + b^2 - c^2)\cos\alpha\cos\beta\\
&\quad +4(a^2 - b^2 - c^2)(a^2 - b^2)\cos\gamma\,,\\
d_2 &= 8c^2(a^2 + b^2)\cos\alpha\cos\beta\cos\gamma\\
&\quad +4c^2(a^2 - c^2)\cos^2\beta + 4c^2(b^2 - c^2)\cos^2\alpha\\
&\quad -4(a^2 - b^2)^2\cos^2\gamma - 2(a^2 - b^2 - c^2)(a^2 - b^2 + c^2)\,,\\
d_1 &= -8a^2c^2\cos^2\beta\cos\gamma - 4c^2(b^2 - c^2)\cos\alpha\cos\beta\\
&\quad -4a^2c^2\cos\alpha\cos\beta + 4(a^2 - b^2 + c^2)(a^2 - b^2)\cos\gamma\,,\\
d_0 &= 4a^2c^2\cos^2\beta - (a^2 - b^2 + c^2)^2\,.
\end{aligned}$$

For every root of the fourth order polynomial, the corresponding value of v can be determined from

$$v = \frac{(a^2 - b^2 - c^2)u^2 + 2(b^2 - a^2)(cos\gamma)u + a^2 - b^2 + c^2}{2c^2(\cos\beta - (\cos\alpha)u)}\,. \tag{4.16}$$

When u and v are known, it is easy to compute s_1, s_2, and s_3, and consequently the position of the feature points in camera coordinates.

The roots of the polynomial can be computed as the eigenvalues of the companion matrix

$$\begin{pmatrix} d_3/d_4 & d_2/d_4 & d_1/d_4 & d_0/d_4 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix}\,. \tag{4.17}$$

With the help of any standard linear algebra package, the implementation of this approach is trivial. According to [Nis04b], however, it is more efficient to bracket the roots with a Sturm chain and to determine their accurate value with a root polishing scheme. The appendix of [Nis04b] contains a concise introduction to recursive Sturm chains, which also form the basis of our root finding algorithm. Instead of the standard bisection approach used in [Nis04b], we polish the bracketed roots with a modified regula falsi approach, which is more efficient on average. To this end, our implementation uses a variant of the modified regula falsi approach also known as the "Illinois" method, which is described and evaluated in [For95].

The second step of the three-point algorithm consists of computing the camera motion by aligning the world coordinate system and the camera coordinate system with the help of the three input feature points. Four different algorithms for this problem are reviewed in [Egg97], where the algorithm based on the singular value

decomposition exhibits the best overall performance. We start our description of this algorithm by expressing the problem as a least-squares optimization problem for N feature points

$$(\hat{\boldsymbol{R}}, \hat{\boldsymbol{t}}) = \underset{(\boldsymbol{R},\boldsymbol{t})}{\operatorname{argmin}} \sum_{i=1}^{N} \|\boldsymbol{c}_i - \boldsymbol{R}\boldsymbol{p}_i - \boldsymbol{t}\|^2 \,. \tag{4.18}$$

At first, the translation is eliminated by moving the center of mass of both sets of feature points to the origin of their respective coordinate system

$$\bar{\boldsymbol{c}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{c}_i \,, \quad \bar{\boldsymbol{p}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{p}_i \,, \tag{4.19}$$

$$\hat{\boldsymbol{R}} = \underset{\boldsymbol{R}}{\operatorname{argmin}} \sum_{i=1}^{N} \|(\boldsymbol{c}_i - \bar{\boldsymbol{c}}) - \boldsymbol{R}\,(\boldsymbol{p}_i - \bar{\boldsymbol{p}})\|^2 \,. \tag{4.20}$$

Then, as explained in [Egg97], a singular value decomposition of the correlation matrix

$$\sum_{i=1}^{N} (\boldsymbol{c}_i - \bar{\boldsymbol{c}})\,(\boldsymbol{p}_i - \bar{\boldsymbol{p}})^{\mathrm{T}} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{U}^{\mathrm{T}} \tag{4.21}$$

allows the computation of rotation and translation as

$$\hat{\boldsymbol{R}} = \boldsymbol{V}\boldsymbol{U}^{\mathrm{T}} \,, \quad \hat{\boldsymbol{t}} = \bar{\boldsymbol{c}} - \hat{\boldsymbol{R}}\bar{\boldsymbol{p}} \,. \tag{4.22}$$

In the case of planar sets of feature points, this algorithm is prone to estimating a reflection instead of a rotation. This case can be detected by checking the determinant of the rotation matrix. When the determinant is negative, the third column of matrix $\boldsymbol{V}$ has to be multiplied by -1 before computing $\hat{\boldsymbol{R}}$ [Egg97].

4.2.2.2 The POSIT Algorithm

As its name implies, the three-point algorithm efficiently computes the extrinsic camera parameters from exactly three feature points. However, when the feature positions are perturbed by noise, algorithms for a larger number of feature points can provide more accurate results. As a consequence, we resort to the POSIT algorithm of [DeM95] whenever more feature points and more computation time are available. The POSIT algorithm simplifies the absolute camera pose problem by assuming a weak-perspective projection model. In order to compute a solution for the perspective projection model, the feature points in the image coordinate system are iteratively moved to updated positions that conform to the weak-perspective projection model. A very similar approach is used in the factorization algorithm of Christy and Horaud in [Chr96].

The input of the POSIT algorithm consists of N feature points in both world coordinates and image coordinates, and one additional reference feature point

$$\boldsymbol{p}_i \in \mathbb{R}^3\,,\;\; \boldsymbol{q}_i \in \mathbb{R}^2\,,\;\; \forall i \in \{1,\ldots,N\}\,,\;\; N \geq 3\,,\;\; \boldsymbol{p}_r \in \mathbb{R}^3\,,\;\; \boldsymbol{q}_r \in \mathbb{R}^2\,. \tag{4.23}$$

The extrinsic camera parameters, which form the output of the algorithm, can be written as

$$\boldsymbol{R} = \begin{pmatrix} \boldsymbol{r}_1^{\mathrm{T}} \\ \boldsymbol{r}_2^{\mathrm{T}} \\ \boldsymbol{r}_3^{\mathrm{T}} \end{pmatrix} \in \mathbb{R}^{3\times 3}\,, \quad \boldsymbol{t} \in \mathbb{R}^3\,. \tag{4.24}$$

In the initial phase of the algorithm, the matrix $\boldsymbol{A}$ and its pseudoinverse $\boldsymbol{A}^+$ are computed from the positions of the feature points in world coordinates

$$\boldsymbol{A} = \begin{pmatrix} (\boldsymbol{p}_1 - \boldsymbol{p}_r)^{\mathrm{T}} \\ \ldots \\ (\boldsymbol{p}_N - \boldsymbol{p}_r)^{\mathrm{T}} \end{pmatrix} \in \mathbb{R}^{N\times 3}\,, \quad \boldsymbol{A}^+ = (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathrm{T}}\,. \tag{4.25}$$

Assuming a weak-perspective projection model for the input data, we arrange the image coordinates of the feature points as

$$\tilde{\boldsymbol{Q}} = ((\boldsymbol{q}_1 - \boldsymbol{q}_r) \;\; \ldots \;\; (\boldsymbol{q}_N - \boldsymbol{q}_r))^{\mathrm{T}} = \begin{pmatrix} \tilde{\boldsymbol{q}}_1 & \tilde{\boldsymbol{q}}_2 \end{pmatrix} \in \mathbb{R}^{N\times 2}\,. \tag{4.26}$$

This allows us to estimate two scaled row vectors of the rotation matrix

$$\tilde{\boldsymbol{r}}_1 = \boldsymbol{A}^+\tilde{\boldsymbol{q}}_1\,, \quad \tilde{\boldsymbol{r}}_2 = \boldsymbol{A}^+\tilde{\boldsymbol{q}}_2\,. \tag{4.27}$$

With the scale factors defined by the row vectors

$$s_1 = \|\tilde{\boldsymbol{r}}_1\|\,, \quad s_2 = \|\tilde{\boldsymbol{r}}_2\|\,, \quad s = \frac{s_1 + s_2}{2}\,, \tag{4.28}$$

we can finally compute the extrinsic camera parameters

$$\boldsymbol{r}_1 = \frac{1}{s_1}\tilde{\boldsymbol{r}}_1\,, \quad \boldsymbol{r}_2 = \frac{1}{s_2}\tilde{\boldsymbol{r}}_2\,, \quad \boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2\,, \quad \boldsymbol{t} = s\underline{\boldsymbol{q}}_r - \boldsymbol{R}\boldsymbol{p}_r\,. \tag{4.29}$$

As the row vectors $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are computed independently from each other, they are not necessarily orthogonal. Therefore, it is possible that the matrix $\boldsymbol{R}$ is not a perfectly orthonormal rotation matrix. As a consequence, we enforce its orthonormality with the help of a singular value decomposition.

In subsequent iterations of the algorithm, the estimated camera parameters are used to adjust the positions of the projected feature points to the weak-perspective projection model. To this end, the relative distance of every feature point from the reference feature point along the optical axis is computed as

$$d_i = s\,(\boldsymbol{p}_i - \boldsymbol{p}_r)^{\mathrm{T}}\boldsymbol{r}_3\,, \quad i \in \{1,\ldots,N\}\,. \tag{4.30}$$

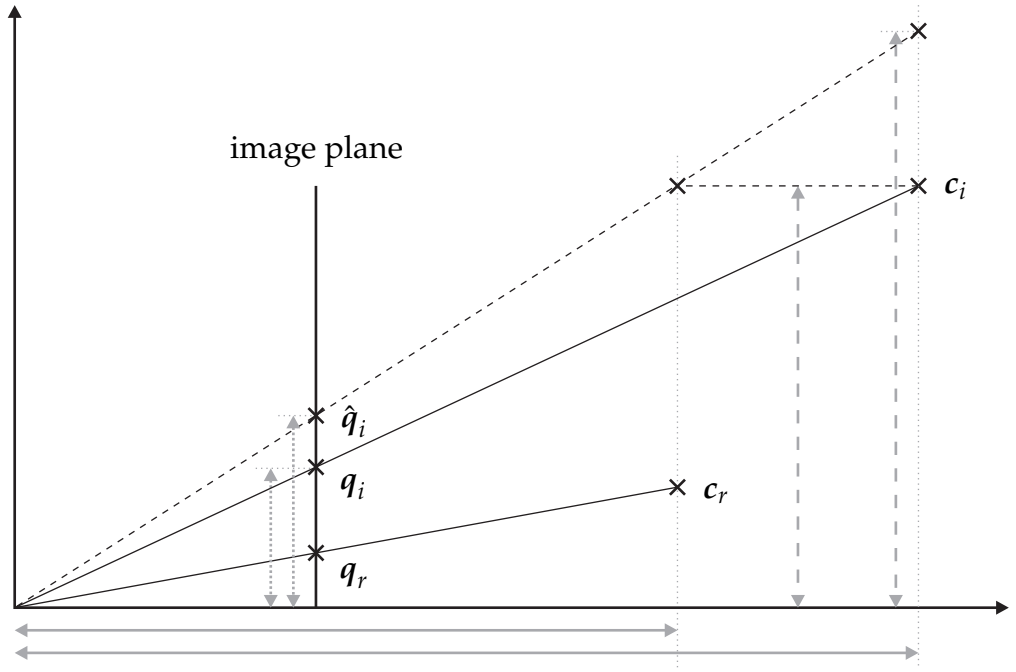


Figure 4.4: The POSIT algorithm adjusts the positions of the feature points $\boldsymbol{q}_i$ to simulate a weak-perspective projection. In this illustration, we show the feature points $\boldsymbol{p}_i$ and $\boldsymbol{p}_r$ as $\boldsymbol{c}_i = \boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{t}$ and $\boldsymbol{c}_r = \boldsymbol{R}\boldsymbol{p}_r + \boldsymbol{t}$ in camera coordinates. The values d_i are computed in (4.30) as the distance of $\boldsymbol{c}_i$ and $\boldsymbol{c}_r$ along the optical axis, relative to the distance of $\boldsymbol{c}_r$ and the optical center along the optical axis. Therefore, the distances marked with solid lines at the bottom of this illustration have a ratio of $1 : (1 + d_i)$. Simple geometric considerations confirm that the other marked distance pairs exhibit the same ratio, which explains the feature position adjustment for $\hat{\boldsymbol{q}}_i$ given in (4.31).

With this information, the positions of the feature points in image coordinates are updated according to

$$\hat{\boldsymbol{q}}_i = (1 + d_i)\, \boldsymbol{q}_i\,, \quad i \in \{1, \ldots, N\}\,. \tag{4.31}$$

The updated feature positions replace the standard positions in (4.26), which is also the starting point for new iterations. As the position of the reference feature point is identical for perspective and weak-perspective projection, its position does not have to be updated. The geometry of the feature position update is illustrated in Figure 4.4. The stopping criterion of the original POSIT algorithm is based on the change of the relative distances d_i. In contrast to this, we monitor the change of the vectors $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$. Our approach has the benefit of directly checking a part of the desired output. It is also slightly more efficient when the number of processed feature points is large.

Unlike the three-point algorithm, the POSIT algorithm does not limit the number of considered feature points. Therefore, we employ it to process all available feature points, which potentially increases the accuracy of the absolute camera pose

estimation. However, even when the POSIT algorithm processes a large number of feature points, its accuracy strongly depends on the accuracy of the position of the reference feature point. This shortcoming can be verified in the equations of (4.25) - (4.29). As a consequence, we propose an enhanced version of the POSIT algorithm, which uses the center of mass of the feature points as its virtual reference point.

The position of the virtual reference point $\boldsymbol{p}_v$ in world coordinates is easily computed as the average of all feature points

$$\boldsymbol{p}_v = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{p}_i \, . \tag{4.32}$$

In contrast to this, the position of the virtual reference point $\boldsymbol{q}_v$ in image coordinates is more difficult to determine, because the projection of the center of mass is not equal to the center of mass of the projections for the perspective projection model. However, the above relation holds true for the weak-perspective projection model. As the projection of the virtual reference point is identical for the perspective and the weak-perspective projection model by construction, we can update the position of the virtual reference point $\hat{\boldsymbol{q}}_v$ in image coordinates as the center of mass of the feature points $\hat{\boldsymbol{q}}_i$ in every iteration of the algorithm

$$\boldsymbol{q}_v = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{q}_i \, , \quad \hat{\boldsymbol{q}}_v = \frac{1}{N} \sum_{i=1}^{N} \hat{\boldsymbol{q}}_i = \frac{1}{N} \sum_{i=1}^{N} (1 + d_i) \, \boldsymbol{q}_i \, . \tag{4.33}$$

The proposed enhancement of the POSIT algorithm has two beneficial effects. On the one hand, the accuracy of the algorithm is considerably improved, because the noise in the positions of the feature points is partially canceled by the averaging in the computation of the virtual reference point. Therefore, from a statistical point of view, the virtual reference point is more accurate than any single feature point selected as reference point in the standard algorithm. On the other hand, the position of the virtual reference point at the center of mass of the feature points is the optimum configuration for the approximation of the perspective projection model with the weak-perspective projection model. This configuration increases the probability for successful convergence and decreases the number of required iterations. The only disadvantage of our enhancement is that the iterative correction of the position of the virtual reference point is prone to slightly increase the number of required iterations. Our experiments in Subsection 6.2.2 prove that the proposed enhancement of the POSIT algorithm is considerably more accurate than the standard algorithm.

4.2.3 Relative Camera Pose Estimation

In this subsection, we consider the problem of estimating the motion of the camera between two views from the corresponding projections of a number of feature

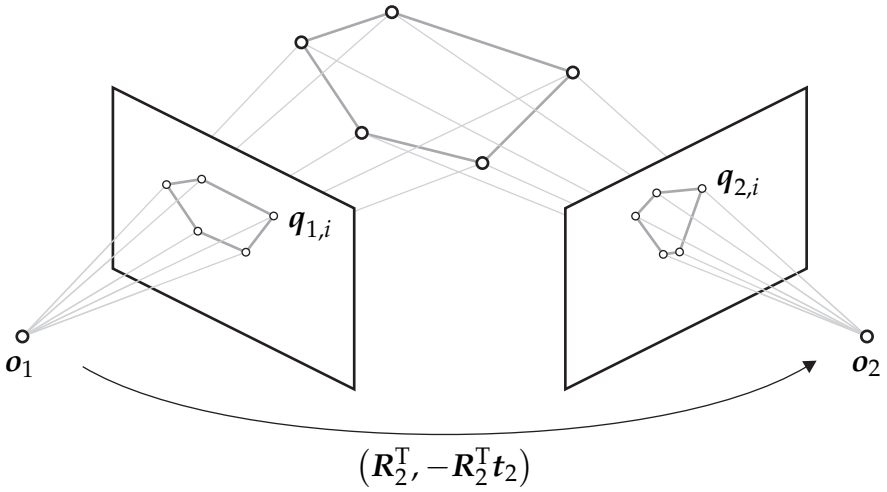


Figure 4.5: The five-point algorithm estimates the camera motion between two views from the positions of five or more feature points in image coordinates.

points. This problem requires the positions of at least five feature points in the calibrated framework, which yield up to ten possible solutions in general. An efficient algorithm suited for a numerical implementation is the five-point algorithm presented by Nister in [Nis04b]. Its basic idea is to compute a tenth degree polynomial, whose roots directly correspond to the possible solutions. The disambiguation of the solutions can then be performed with the help of an additional feature point or an additional view. Like all structure and motion estimation algorithms whose input consists solely of feature positions in image coordinates, the five-point algorithm is generally unable to recover the global scale of the scene.

The five-point algorithm of Nister is currently the state-of-the-art algorithm for calibrated two-view motion estimation. It is compared to the established seven-point [Har94b] and eight-point [Har97a] algorithms in the original work, where it is found to deliver the most consistent results [Nis04b]. In addition to that, as a member of the calibrated framework, the five-point algorithm can also cope with planar scenes, whereas the algorithms of the uncalibrated framework cannot determine a unique reconstruction from two views and a planar scene due to the planar structure degeneracy. Recently, three modifications for the five-point algorithm have been proposed. [Li06] uses the hidden variable technique to simplify the computation of the tenth degree polynomial. In [Ste06], the framework of algebraic geometry is employed to derive an alternate solution to the five-point relative pose problem. Finally, [Bat07] reformulates this problem as a constrained optimization problem that yields nine quadratic equations in six variables. Although all proposed modifications slightly improve the accuracy of the five-point algorithm, we rely on the original algorithm, whose efficiency is still unsurpassed.

The five-point relative pose problem is illustrated in Figure 4.5. As it is only possible to recover the relative motion, we are free to let the camera coordinate

system of the first view coincide with the world coordinate system. Therefore, the extrinsic camera parameters of the first view are $(\boldsymbol{R}_1, \boldsymbol{t}_1) = (\boldsymbol{I}, \boldsymbol{0})$ and the relative motion between the two views can be expressed in terms of the extrinsic camera parameters of the second view as $\left(\boldsymbol{R}_2^{\mathrm{T}}, -\boldsymbol{R}_2^{\mathrm{T}}\boldsymbol{t}_2\right)$. The positions of the feature points are given in homogeneous coordinates as

$$\underline{\boldsymbol{q}}_{1,i} = \begin{pmatrix} q_{1,i,1} \\ q_{1,i,2} \\ q_{1,i,3} \end{pmatrix}, \quad \underline{\boldsymbol{q}}_{2,i} = \begin{pmatrix} q_{2,i,1} \\ q_{2,i,2} \\ q_{2,i,3} \end{pmatrix}, \quad i \in \{1, \ldots, 5\} \tag{4.34}$$

for the left and right view, respectively.

The first step of the five-point algorithm is to recover the essential matrix $\boldsymbol{E}$. As described in Subsection 2.4, every pair of corresponding feature points has to satisfy the epipolar constraint

$$\underline{\boldsymbol{q}}_{2,i}^{\mathrm{T}} \, \boldsymbol{E} \, \underline{\boldsymbol{q}}_{1,i} = 0 \,. \tag{4.35}$$

After the definition of the matrix $\boldsymbol{Q}_i = \underline{\boldsymbol{q}}_{2,i}\underline{\boldsymbol{q}}_{1,i}^{\mathrm{T}}$ and the two vectors $\check{\boldsymbol{q}}_i$ and $\check{\boldsymbol{e}}$

$$\check{\boldsymbol{q}}_i = (Q_{i,11} \;\; Q_{i,12} \;\; Q_{i,13} \;\; Q_{i,21} \;\; Q_{i,22} \;\; Q_{i,23} \;\; Q_{i,31} \;\; Q_{i,32} \;\; Q_{i,33})^{\mathrm{T}} ,$$

$$\check{\boldsymbol{e}} = (E_{11} \;\; E_{12} \;\; E_{13} \;\; E_{21} \;\; E_{22} \;\; E_{23} \;\; E_{31} \;\; E_{32} \;\; E_{33})^{\mathrm{T}} ,$$

this constraint can be reformulated as the inner product

$$\check{\boldsymbol{q}}_i^{\mathrm{T}} \, \check{\boldsymbol{e}} = 0 \,. \tag{4.36}$$

In order to compute the vector representation $\check{\boldsymbol{e}}$ of the essential matrix $\boldsymbol{E}$, the vectors $\check{\boldsymbol{q}}_i$ are arranged as the rows of the matrix $\check{\boldsymbol{Q}}$

$$\check{\boldsymbol{Q}} = (\check{\boldsymbol{q}}_1 \;\; \check{\boldsymbol{q}}_2 \;\; \check{\boldsymbol{q}}_3 \;\; \check{\boldsymbol{q}}_4 \;\; \check{\boldsymbol{q}}_5)^{\mathrm{T}} \in \mathbb{R}^{5\times 9} \tag{4.37}$$

and the four vectors spanning the right nullspace of this matrix are estimated

$$\mathrm{kernel}(\check{\boldsymbol{Q}}) = \{\check{\boldsymbol{x}}, \check{\boldsymbol{y}}, \check{\boldsymbol{z}}, \check{\boldsymbol{w}}\} \,. \tag{4.38}$$

The vector $\check{\boldsymbol{e}}$ can then be expressed as a linear combination of these vectors

$$\check{\boldsymbol{e}} = x\,\check{\boldsymbol{x}} + y\,\check{\boldsymbol{y}} + z\,\check{\boldsymbol{z}} + w\,\check{\boldsymbol{w}} \,, \quad x, y, z, w \in \mathbb{R} \,. \tag{4.39}$$

As the essential matrix is only defined up to a scale factor, we assume $w = 1$ and insert the resulting equation into the constraints on the essential matrix in (2.12) and (2.14). Thus, we obtain an equation system with ten cubic polynomials in the variables x, y, and z. Performing an incomplete Gauss-Jordan elimination with partial pivoting yields the equation system shown in Table 4.1. We combine selected equations and arrange the results into the equation system

$$\begin{pmatrix} \langle e \rangle - z\langle f \rangle \\ \langle g \rangle - z\langle h \rangle \\ \langle i \rangle - z\langle j \rangle \end{pmatrix} = \begin{pmatrix} Z_{11}(z) & Z_{12}(z) & Z_{13}(z) \\ Z_{21}(z) & Z_{22}(z) & Z_{23}(z) \\ Z_{31}(z) & Z_{32}(z) & Z_{33}(z) \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \boldsymbol{Z}\,\boldsymbol{v} = \boldsymbol{0} \,. \tag{4.40}$$

	x^3	y^3	x^2y	xy^2	x^2z	x^2	y^2z	y^2	xyz	xy	x	y	1
$\langle a \rangle$	1	.	.	.	.	.	.	.	.	.	[2]	[2]	[3]
$\langle b \rangle$	0	1	.	.	.	.	.	.	.	.	[2]	[2]	[3]
$\langle c \rangle$	0	0	1	.	.	.	.	.	.	.	[2]	[2]	[3]
$\langle d \rangle$	0	0	0	1	.	.	.	.	.	.	[2]	[2]	[3]
$\langle e \rangle$	0	0	0	0	1	0	0	0	0	0	[2]	[2]	[3]
$\langle f \rangle$	0	0	0	0	0	1	0	0	0	0	[2]	[2]	[3]
$\langle g \rangle$	0	0	0	0	0	0	1	0	0	0	[2]	[2]	[3]
$\langle h \rangle$	0	0	0	0	0	0	0	1	0	0	[2]	[2]	[3]
$\langle i \rangle$	0	0	0	0	0	0	0	0	1	0	[2]	[2]	[3]
$\langle j \rangle$	0	0	0	0	0	0	0	0	0	1	[2]	[2]	[3]

Table 4.1: The main equation system of the five-point algorithm consists of ten polynomials in three variables. The illustrated state of the system is obtained by performing an incomplete Gauss-Jordan elimination with partial pivoting. A dot represents a real number and [n] denotes a polynomial of degree n in z.

The matrix $\boldsymbol{Z}$ contains univariate polynomials of degree 3 (first two columns) and degree 4 (third column) in z. For every solution of this equation system, the vector $\boldsymbol{v}$ has to be a nullvector of the matrix $\boldsymbol{Z}$. Consequently, the determinant of the matrix $\boldsymbol{Z}$, which is a tenth degree polynomial in z, must be zero. Therefore, the values of z can be computed as the real roots of a tenth degree polynomial. For each solution for z, we evaluate the polynomials in the first two rows of the matrix $\boldsymbol{Z}$, and obtain the values of x and y by solving the equation system

$$\begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -Z_{13} \\ -Z_{23} \end{pmatrix}. \tag{4.41}$$

Finally, the essential matrix can be determined for every solution of the equation system in (4.40) by applying the corresponding values of x, y, and z to (4.39).

In the following, we will complement the mathematical description of the first step of the five-point algorithm with a brief discussion of further technical details. A valuable property of the five-point algorithm is its ability to process more than five points. In the overdetermined case, the matrix $\check{\boldsymbol{Q}}$ has as many rows as there are points. Its nullvectors are computed with a singular value decomposition as the four singular vectors that correspond to the smallest singular values. The other steps of the five-point algorithm remain unaffected. In the minimal case, however, the nullspace of the matrix $\check{\boldsymbol{Q}}$ can be determined much more efficiently with a QR factorization. Another important computation step for the efficiency of the algorithm is the extraction of the roots of the tenth degree polynomial in z. This problem has already been discussed in the context of the three-point algorithm in the previous subsection and is solved with the same root bracketing and polishing scheme.

The second step of the five-point algorithm consists of recovering the relative camera motion from the essential matrices corresponding to the roots of the tenth degree polynomial. A singular value decomposition of the essential matrix yields

$$\boldsymbol{E} = \boldsymbol{U}\ \mathrm{diag}(d, d, 0)\ \boldsymbol{V}^{\mathrm{T}}, \quad d \in \mathbb{R}^{+}. \tag{4.42}$$

After the definition of the matrix

$$\boldsymbol{D} = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{4.43}$$

the extrinsic camera parameters of the second view are given by

$$\begin{aligned} \boldsymbol{R}_2 = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{\mathrm{T}} \quad \text{or} \quad \boldsymbol{R}_2 = \boldsymbol{U}\boldsymbol{D}^{\mathrm{T}}\boldsymbol{V}^{\mathrm{T}}, \\ \boldsymbol{t}_2 = (U_{13}\ \ U_{23}\ \ U_{33})^{\mathrm{T}} \quad \text{or} \quad \boldsymbol{t}_2 = -(U_{13}\ \ U_{23}\ \ U_{33})^{\mathrm{T}}. \end{aligned} \tag{4.44}$$

All four combinations of $\boldsymbol{R}_2$ and $\boldsymbol{t}_2$ satisfy the epipolar constraint. In order to determine the correct configuration, it is necessary to enforce the cheirality constraint, which states that the feature points must lie in front of the cameras. This can be achieved by triangulating a single feature point, because every feature point lies in front of both cameras only for the correct configuration.

In the appendix of [Nis04b], Nister describes a custom-made singular value decomposition of the essential matrix. The specialized algorithm requires only a small number of mathematical operations and is much more efficient than the standard singular value decomposition. In the minimal case of five feature points, the triangulation of these feature points can be performed with the efficient algorithm described at the end of Subsection 4.2.1. We have already stated that the five-point algorithm is unable to recover the global scale of the scene. It is interesting to note that the equations for the recovery of the camera motion fix the length of the recovered translation vector at $\|\boldsymbol{t}_2\| = 1$.

The final step of the five-point algorithm is to determine the true solution from the up to ten putative solutions computed as the roots of the tenth degree polynomial. This is possible with an additional feature point, which has to be triangulated and back-projected. For the back-projection, the extrinsic camera parameters and the position of the feature point in world coordinates are combined according to (2.2) and (2.3). When the position of the observed feature point $\boldsymbol{q}$ and the position of the back-projected feature point $\vec{\boldsymbol{q}}$ are known, the squared back-projection error is defined as

$$\epsilon_{\mathrm{bp}} = \|\boldsymbol{q} - \vec{\boldsymbol{q}}\|^2. \tag{4.45}$$

For the disambiguation of the putative solutions, the solution with the smallest sum of squared back-projection errors for the additional point and the two views is chosen as the final solution. However, when all feature points lie on a single plane, this approach suffers from the existence of special scene configurations, which lead

to an ambiguity with two distinct solutions, one of which is obviously incorrect [Nis04b].

In order to robustify the five-point algorithm with respect to the described scene configurations, we employ the alternative approach of disambiguating the computed solutions with an additional view. To this end, the five feature points are triangulated and the corresponding extrinsic camera parameters of the third view are estimated with the three-point algorithm. Then, the estimated camera parameters are used to back-project the two remaining feature points into the additional view. The final solution is obtained by finding the smallest sum of squared back-projection errors. As the absolute camera pose problem with three feature points has up to four possible solutions, the total number of putative solutions is at most forty. The only remaining practical restriction of the described approach is that the centers of projection of the different views must not coincide, which is an inherent limitation of all structure and motion algorithms.

4.2.4 Bundle Adjustment

The algorithms of the preceding subsections efficiently solve specific aspects of the structure and motion estimation problem. Although all presented algorithms exhibit perfect theoretical accuracy when ideal input data is processed with high-precision arithmetics, they are not optimized for generating accurate results with noisy input data. In addition to that, these algorithms are generally unable to compute a globally optimal reconstruction due to their restricted scope. Consequently, we complement the presented algorithms with a bundle adjustment approach, which refines the computed reconstruction by simultaneously optimizing both the camera motion and the scene structure with respect to a geometrically meaningful cost function. Its name derives from the bundles of light rays that pass through both the feature points and the optical centers of the cameras. An elaborate overview of historical, theoretical, and practical aspects of bundle adjustment is given in [Tri99].

4.2.4.1 Problem Statement

Bundle adjustment approaches can be applied to problems of different sizes, which range from the optimization of the position of a single feature point to the optimization of the reconstruction of a complete image sequence. In order to provide a general description of bundle adjustment, we assume that the optimization problem consists of M views and N feature points. Furthermore, we assume that every feature point has been observed in every view, which is by no means a requirement of bundle adjustment itself, but considerably simplifies the mathematical notation. Thus, the input of bundle adjustment consists of MN 2-D feature positions $\boldsymbol{q}_{ij}$ with $i \in \{1, \ldots, M\}$ and $j \in \{1, \ldots, N\}$. Like other approaches for non-linear optimization, bundle adjustment depends on initial estimates of the parameters to

be optimized. In our case, these estimates comprise the extrinsic camera parameters $(\tilde{\boldsymbol{R}}_i, \tilde{\boldsymbol{t}}_i)$ with $i \in \{1, \ldots, M\}$ and the 3-D feature positions $\tilde{\boldsymbol{p}}_j$ with $j \in \{1, \ldots, N\}$.

The standard cost function used for bundle adjustment is the sum of the squared back-projection errors of all feature points in all views. It can be shown that this cost function yields a maximum likelihood estimator when the errors in the observed 2-D feature positions conform to a zero-mean Gaussian distribution [Tru98]. However, even one single strong outlier in the feature positions can perturb the final solution by an arbitrarily large amount. This problem will be solved by the robust cost functions presented in Subsection 4.3.4. We now turn to the definition of the standard cost function, which requires a number of preliminary explanations. Let the optimization parameters, i. e., the extrinsic camera parameters and the 3-D feature positions, be encoded in the parameter vector $\boldsymbol{\phi}$ in a yet unspecified way. Although the 2-D back-projection $\vec{\boldsymbol{q}}_{ij}$ of feature point j into view i implicitly depends on the parameter vector, we do not indicate this relationship in the following equations to simplify our mathematical notation. With the definition of the vector function

$$\boldsymbol{\epsilon}(\boldsymbol{\phi}) = \left((\boldsymbol{q}_{11} - \vec{\boldsymbol{q}}_{11})^{\mathrm{T}} \;\; (\boldsymbol{q}_{12} - \vec{\boldsymbol{q}}_{12})^{\mathrm{T}} \;\; \ldots \;\; (\boldsymbol{q}_{MN} - \vec{\boldsymbol{q}}_{MN})^{\mathrm{T}} \right)^{\mathrm{T}}, \tag{4.46}$$

the standard bundle adjustment cost function is given by

$$\mathcal{E}(\boldsymbol{\phi}) = \frac{1}{2}\, \boldsymbol{\epsilon}(\boldsymbol{\phi})^{\mathrm{T}} \boldsymbol{\epsilon}(\boldsymbol{\phi}) \tag{4.47}$$

and the bundle adjustment problem can be formulated as

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\; \mathcal{E}(\boldsymbol{\phi}). \tag{4.48}$$

4.2.4.2 Parameterization

The definition of the parameter vector $\boldsymbol{\phi}$, also known as the parameterization of the optimization problem, is an important detail of every parameter estimation approach. The following properties indicate a suitable parameterization:

- The parameterization is finite near the current parameter estimate. Large or infinite parameter values require an overly large number of finite update steps of the iterative optimization algorithm.

- According to [Tri99], the parameterization should be locally close to linear with respect to its effect on the vector function $\boldsymbol{\epsilon}(\boldsymbol{\phi})$. This property improves the convergence rate of the optimization algorithm. In contrast to this, parameterization singularities are prone to cause slow convergence and numerical instabilities.

- The parameterization is minimal, i. e., the number of parameters equals the degrees of freedom of the modeled geometric entity. A parameterization that violates this property increases the size of the parameter space, which usually decreases the computational efficiency of the optimization algorithm, unless this parameterization also has a positive effect on its convergence rate.
- The parameterization is fair, i. e., any rigid transformation of space results in an orthogonal transformation of the parameter values. This property ensures that the sensitivity of the optimization parameters on small errors in the input data is not governed by the choice of the world coordinate system [Hor99].

It is possible that two parameterization properties exclude each other for some geometric entities. For example, the 3-D feature positions in standard coordinates can become quite large when a feature point lies far from the center of the world coordinate system. This problem does not arise for homogeneous coordinates, which allow the representation of any feature point, even including feature points at infinity, with a normalized vector of unit length. However, homogeneous coordinates are not a minimal parameterization of the point positions. As a consequence, selecting a parameterization for a given geometric entity often involves judging the respective trade-offs by experience.

We are especially interested in the computational efficiency provided by a minimal parameterization. Thus, we represent both the camera translations and the feature positions in standard coordinates. The camera rotation, which is usually specified as a 3×3 rotation matrix, has only three degrees of freedom. Alternative representations include Euler angles, which are not a fair parameterization, quaternions, which are not a minimal parameterization, and the angle-axis representation, which has all desired properties. The angle-axis representation consists of a vector $\check{\boldsymbol{r}} \in \mathbb{R}^3$ that points in the direction of the rotation axis. The rotation angle is encoded in the Euclidean norm of the vector. A rotation in angle-axis representation can be converted into a rotation matrix with Rodrigues' rotation formula:

$$\boldsymbol{R} = \boldsymbol{I} + \frac{\sin\|\check{\boldsymbol{r}}\|}{\|\check{\boldsymbol{r}}\|} \begin{pmatrix} 0 & -\check{r}_3 & \check{r}_2 \\ \check{r}_3 & 0 & -\check{r}_1 \\ -\check{r}_2 & \check{r}_1 & 0 \end{pmatrix} + \frac{1-\cos\|\check{\boldsymbol{r}}\|}{\|\check{\boldsymbol{r}}\|^2} \begin{pmatrix} 0 & -\check{r}_3 & \check{r}_2 \\ \check{r}_3 & 0 & -\check{r}_1 \\ -\check{r}_2 & \check{r}_1 & 0 \end{pmatrix}^2 .$$

As it is not possible to apply a rotation directly in the angle-axis representation, Rodrigues' rotation formula is used very often in the course of the optimization process, e. g., for the computation of the current reprojection error. However, directly using the rotation matrix representation also requires additional operations. As the elements of the parameter vector are optimized without considering the orthonormality of the encoded rotation matrices, the matrices have to be normalized after every parameter update. All in all, the computation time saved by the angle-axis representation easily outweighs the time required for the additional conversion operations.

We are finally able to specify our parameter vector $\boldsymbol{\phi}$. It can be partitioned into the vectors $\boldsymbol{\phi}_{c,i}$, which describe the extrinsic camera parameters, and the vectors $\boldsymbol{\phi}_{p,j}$, which describe the feature positions, as follows

$$\boldsymbol{\phi} = \left(\boldsymbol{\phi}_C^{\mathrm{T}}\ ,\ \boldsymbol{\phi}_P^{\mathrm{T}}\right)^{\mathrm{T}} = \left(\boldsymbol{\phi}_{c,1}^{\mathrm{T}}\ ,\ldots,\ \boldsymbol{\phi}_{c,M}^{\mathrm{T}}\ ,\ \boldsymbol{\phi}_{p,1}^{\mathrm{T}}\ ,\ldots,\ \boldsymbol{\phi}_{p,N}^{\mathrm{T}}\right)^{\mathrm{T}}. \tag{4.49}$$

The camera parameter vectors $\boldsymbol{\phi}_{c,i}$ consist of six components forming a minimal parameterization of the rotation and the translation of a single camera:

$$\boldsymbol{\phi}_{c,i} = \left(\check{\boldsymbol{r}}_i^{\mathrm{T}}\ ,\ \boldsymbol{t}_i^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^6. \tag{4.50}$$

The feature positions are represented in standard coordinates as

$$\boldsymbol{\phi}_{p,j} = \boldsymbol{p}_j \in \mathbb{R}^3. \tag{4.51}$$

All in all, the parameter vector $\boldsymbol{\phi}$ has $6M + 3N$ components.

4.2.4.3 Optimization Algorithms

The non-linear least-squares optimization problem arising in bundle adjustment is traditionally solved with one of the many variants of the Levenberg-Marquardt optimization method. For the derivation of this method, we introduce a linear model $\boldsymbol{l}(\boldsymbol{h})$ of the vector function $\boldsymbol{\epsilon}(\boldsymbol{\phi})$ by performing a Taylor expansion around the current parameter value

$$\boldsymbol{\epsilon}(\boldsymbol{\phi}+\boldsymbol{h})\ \approx\ \boldsymbol{l}(\boldsymbol{h}) = \boldsymbol{\epsilon}(\boldsymbol{\phi}) + \boldsymbol{J}(\boldsymbol{\phi})\boldsymbol{h}\ ,\quad (\boldsymbol{J}(\boldsymbol{\phi}))_{ij} = \frac{\partial \epsilon_i}{\partial \phi_j}(\boldsymbol{\phi})\ . \tag{4.52}$$

The partial derivatives of $\boldsymbol{\epsilon}(\boldsymbol{\phi})$ are stored in the Jacobian matrix $\boldsymbol{J}(\boldsymbol{\phi})$. As the symbolic computation of the partial derivatives of $\boldsymbol{\epsilon}(\boldsymbol{\phi})$ is quite complex, we compute the entries of the Jacobian matrix by numerical differentiation. By combining (4.47) and (4.52), we can define the linear model $L(\boldsymbol{h})$ of the cost function $\mathcal{E}(\boldsymbol{\phi})$ as

$$\mathcal{E}(\boldsymbol{\phi}+\boldsymbol{h}) \approx L(\boldsymbol{h}) = \frac{1}{2}\,\boldsymbol{l}(\boldsymbol{h})^{\mathrm{T}}\boldsymbol{l}(\boldsymbol{h}) = \mathcal{E}(\boldsymbol{\phi}) + \boldsymbol{h}^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{\phi})^{\mathrm{T}}\boldsymbol{\epsilon}(\boldsymbol{\phi}) + \frac{1}{2}\,\boldsymbol{h}^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{\phi})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{\phi})\boldsymbol{h}\ .$$

The vector minimizing the linearized cost function $L(\boldsymbol{h})$ is known as the Gauss-Newton step $\boldsymbol{h}_{\mathrm{gn}}$

$$\boldsymbol{h}_{\mathrm{gn}} = \operatorname*{argmin}_{\boldsymbol{h}}\ L(\boldsymbol{h})\ . \tag{4.53}$$

After setting the first derivative of $L(\boldsymbol{h})$ to zero

$$L'(\boldsymbol{h}) = \boldsymbol{J}(\boldsymbol{\phi})^{\mathrm{T}}\boldsymbol{\epsilon}(\boldsymbol{\phi}) + \boldsymbol{J}(\boldsymbol{\phi})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{\phi})\boldsymbol{h} = 0\ , \tag{4.54}$$

the Gauss-Newton step $\boldsymbol{h}_{\text{gn}}$ can be computed as the solution of the so-called normal equations

$$\left(\boldsymbol{J}(\boldsymbol{\phi})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{\phi})\right)\boldsymbol{h}_{\text{gn}} = -\boldsymbol{J}(\boldsymbol{\phi})^{\mathrm{T}}\boldsymbol{\epsilon}(\boldsymbol{\phi}) \; . \tag{4.55}$$

Although $\boldsymbol{h}_{\text{gn}}$ is a descent direction for $\mathcal{E}(\boldsymbol{\phi})$, the actual length of the update step has to be determined with an additional line search to obtain guaranteed convergence. The Levenberg-Marquardt method avoids the additional effort by using a minor variation of (4.55) called the augmented normal equations

$$\left(\boldsymbol{J}(\boldsymbol{\phi})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{\phi}) + \lambda\,\boldsymbol{I}\right)\boldsymbol{h}_{\text{lm}} = -\boldsymbol{J}(\boldsymbol{\phi})^{\mathrm{T}}\boldsymbol{\epsilon}(\boldsymbol{\phi}) \; , \quad \lambda > 0 \; . \tag{4.56}$$

The introduction of the damping parameter λ has several effects:

- As $\lambda > 0$, the coefficient matrix $(\boldsymbol{J}(\boldsymbol{\phi})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{\phi}) + \lambda\,\boldsymbol{I})$ is positive definite. Thus, the augmented normal equations always have a unique solution.
- For small values of λ, $\boldsymbol{h}_{\text{lm}}$ almost equals $\boldsymbol{h}_{\text{gn}}$, which is ideal when the cost function $\mathcal{E}(\boldsymbol{\phi})$ and its linear model $L(\boldsymbol{h})$ closely resemble each other.
- For large values of λ, $\boldsymbol{h}_{\text{lm}}$ is a small step in the steepest descent direction. Therefore, guaranteed convergence can be achieved by choosing values of λ that are large enough.

The general approach of the Levenberg-Marquardt method can be summarized as follows. At first, the parameter vector $\boldsymbol{\phi}$ is set to the initial parameter estimates provided as input and the damping parameter λ is initialized. Then, the main loop of the method is entered. In every iteration, the parameter update $\boldsymbol{h}_{\text{lm}}$ is computed by solving the augmented normal equations for the current parameters. When the cost function decreases for the new parameters $(\boldsymbol{\phi} + \boldsymbol{h}_{\text{lm}})$, the parameter update is accepted, otherwise it is rejected. There are a number of different strategies for updating the damping parameter λ. However, all strategies have in common that the damping parameter λ has to be increased in the case of a rejected parameter update. Finally, different stopping criteria are used for the termination of the main optimization loop.

A description of the traditional Levenberg-Marquardt method can be found in the appendix of [Har00]. In this thesis, we use the modern variant described in [Lou05b]. Apart from the fact that [Har00] does not specify any stopping criteria, the differences between the two variants are related to the damping parameter λ. The traditional variant initializes λ with a user-defined value. In contrast to this, the initialization of λ in the modern variant is

$$\tilde{\lambda} = \lambda_{\text{init}} \; * \; \max_i \left(\boldsymbol{J}(\tilde{\boldsymbol{\phi}})^{\mathrm{T}}\boldsymbol{J}(\tilde{\boldsymbol{\phi}})\right)_{ii} \; , \quad \lambda_{\text{init}} = 0.001 \; , \tag{4.57}$$

which takes the input data into account and facilitates a faster convergence in the early iteration steps. In a similar way, the update strategy for the damping parameter λ is more sophisticated in the modern variant. The traditional approach is

to increase or decrease λ by a factor of ten after every iteration, depending on the change of the cost function. This strategy can cause the damping parameter to alternate between the same two values, which is prone to slow down the convergence. The update strategy of the modern variant is based on the gain ratio γ

$$\gamma = \frac{\mathcal{E}(\boldsymbol{\phi}) - \mathcal{E}(\boldsymbol{\phi} + \boldsymbol{h}_{\text{lm}})}{L(\mathbf{0}) - L(\boldsymbol{h}_{\text{lm}})} \,, \tag{4.58}$$

$$L(\mathbf{0}) - L(\boldsymbol{h}_{\text{lm}}) = \frac{1}{2}\, \boldsymbol{h}_{\text{lm}}^{\text{T}} \left(\lambda\, \boldsymbol{h}_{\text{lm}} - \boldsymbol{J}(\boldsymbol{\phi})^{\text{T}} \boldsymbol{\epsilon}(\boldsymbol{\phi}) \right) , \tag{4.59}$$

which measures how well the linear model approximates the true cost function. When the gain ratio γ is greater than zero, i. e., the cost function has decreased, the damping parameter λ is multiplied by the maximum of $\frac{1}{3}$ and $1 - (2\gamma - 1)^3$. Otherwise, the damping parameter is multiplied by 2^i for the i-th consecutive failure to decrease the cost function. This strategy carefully tunes the damping parameter λ when the gain ratio is positive, but also takes decisive corrective action when the cost function has not been decreased for several iterations.

Our implementation of the Levenberg-Marquardt optimization method supports three stopping criteria. In order to protect the method against an infinite loop, we limit the maximum number of iterations with the help of the user-specified parameter $\delta_{\text{bun\_iter}}$. The standard stopping criterion checks if the gradient of the cost function is sufficiently close to zero

$$\| \boldsymbol{J}(\boldsymbol{\phi})^{\text{T}} \boldsymbol{\epsilon}(\boldsymbol{\phi}) \|_\infty \leq \theta_{\text{bun\_gra}} \,, \tag{4.60}$$

which is a necessary condition for a local minimum. Finally, we check the actual change in the parameter vector $\boldsymbol{\phi}$ with the third stopping criterion

$$\|\boldsymbol{h}_{\text{lm}}\| \leq \theta_{\text{bun\_upd}} \left(\|\boldsymbol{\phi}\| + \theta_{\text{bun\_upd}} \right) . \tag{4.61}$$

This criterion complements the second criterion, which is prone to the effects of rounding errors when $\theta_{\text{bun\_gra}}$ is very small. In our implementation, we use

$$\theta_{\text{bun\_gra}} = 10^{-6} \quad \text{and} \quad \theta_{\text{bun\_upd}} = 10^{-6} . \tag{4.62}$$

Lourakis and Argyros proposed to replace the Levenberg-Marquardt method with Powell's dog leg method in [Lou05b]. Both methods compute their parameter updates as a combination of the Gauss-Newton and the steepest descent directions. Unlike the Levenberg-Marquardt method, the dog leg method solves the normal equations without adding a damping parameter. Instead, it controls the maximum step size with the help of the radius of a trust region. The better efficiency of the dog-leg algorithm mainly derives from the fact that the normal equations have to be solved at most once for every successful iteration, instead of once for every iteration, including unsuccessful ones. However, while the augmented normal equations usually yield a positive definite matrix, which can be inverted with an efficient

Cholesky factorization, the standard normal equations are prone to produce positive semi-definite and indefinite matrices, which can only be inverted with more complicated and less efficient algorithms. All in all, the marginal increase in computation speed that we have been able to achieve in our experiments does not warrant a replacement of the reliable Levenberg-Marquardt method in our case.

4.2.4.4 Efficient Bundle Adjustment

The most time-consuming operation during the solution of the augmented normal equations is the inversion of a matrix of size $(6M + 3N) \times (6M + 3N)$, which has a run-time complexity of $O((6M + 3N)^3)$. Consequently, the efficient solution of large bundle adjustment problems requires additional techniques. There are several possible techniques for accelerating bundle adjustment:

- The most obvious technique is the reduction of the amount of data, which can be achieved by removing either cameras or feature points from the input data. Examples include the use of virtual key frames in [Shu99], the restriction of the optimization to the most recent frames in the real-time approach of [Eng06], and the independent processing of the segments of a video sequence described in Subsection 4.3.5. The main drawback of this technique is that the discarding of relevant data possibly reduces the accuracy of the bundle adjustment approach.

- Another technique for avoiding the inversion of the large matrix is called interleaving. Its basic idea is to alternate the optimization of the camera parameters and the point positions. As each camera is independent of the other cameras when the feature points are fixed, and vice versa, the largest matrix that has to be inverted has a size of 6×6. However, the theoretical advantage of a much lower computational complexity is mostly offset by the much slower convergence. Especially when the average trail length is small compared to the length of the video sequence, parameter changes take very long to propagate from one end of the sequence to the other.

- The most promising technique for accelerating bundle adjustment is to exploit the sparsity of the problem structure, which manifests itself in the Jacobian matrix $\boldsymbol{J}(\boldsymbol{\phi})$. All elements of a row of the Jacobian matrix are zero except for the parameters of one camera and one feature point. Detailed descriptions of the algorithm for exploiting the resulting structure of the matrix $\boldsymbol{J}(\boldsymbol{\phi})^{\mathrm{T}}\boldsymbol{J}(\boldsymbol{\phi})$ are given in [Har00] and [Eng06]. The new algorithm has to invert one matrix of size $6M \times 6M$ and N matrices of size 3×3. This reduction in computational complexity, which comes without any significant disadvantages, is especially impressive when the number of feature points is large. Nevertheless, the additional omission of cameras can be advisable for long video sequences, when ultimate computational efficiency is required.

As we accelerate the bundle adjustment approach in our structure and motion estimation system by exploiting the sparsity of the Jacobian matrix, we provide a concise explanation of this technique. Compared to the standard Levenberg-Marquardt method, only the normal equations are solved differently, whereas all other parts remain unaffected. When the Jacobian matrix is split into two submatrices $\boldsymbol{J}(\boldsymbol{\phi}) = (\, \boldsymbol{J}_C(\boldsymbol{\phi}) \;\; \boldsymbol{J}_P(\boldsymbol{\phi}) \,)$ and the vector $\boldsymbol{\phi}$ is omitted for better readability, the augmented normal equations can be written as

$$\begin{pmatrix} \boldsymbol{J}_C^{\mathrm{T}} \boldsymbol{J}_C + \lambda \boldsymbol{I} & \boldsymbol{J}_C^{\mathrm{T}} \boldsymbol{J}_P \\ \boldsymbol{J}_P^{\mathrm{T}} \boldsymbol{J}_C & \boldsymbol{J}_P^{\mathrm{T}} \boldsymbol{J}_P + \lambda \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{h}_C \\ \boldsymbol{h}_P \end{pmatrix} = \begin{pmatrix} -\boldsymbol{J}_C^{\mathrm{T}} \boldsymbol{\epsilon} \\ -\boldsymbol{J}_P^{\mathrm{T}} \boldsymbol{\epsilon} \end{pmatrix}. \tag{4.63}$$

The matrices $\boldsymbol{H}_{CC} = \boldsymbol{J}_C^{\mathrm{T}} \boldsymbol{J}_C + \lambda \boldsymbol{I}$ and $\boldsymbol{H}_{PP} = \boldsymbol{J}_P^{\mathrm{T}} \boldsymbol{J}_P + \lambda \boldsymbol{I}$ are both block-diagonal, whereas the structure of the matrix $\boldsymbol{H}_{CP} = \boldsymbol{J}_C^{\mathrm{T}} \boldsymbol{J}_P = (\boldsymbol{J}_P^{\mathrm{T}} \boldsymbol{J}_C)^{\mathrm{T}}$ depends on the input data. Multiplying the augmented normal equations by

$$\begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{H}_{PP}^{-1} \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \boldsymbol{I} & -\boldsymbol{H}_{CP} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \tag{4.64}$$

from the left yields the equation

$$\begin{pmatrix} \boldsymbol{H}_{CC} - \boldsymbol{H}_{CP} \boldsymbol{H}_{PP}^{-1} \boldsymbol{H}_{CP}^{\mathrm{T}} & \boldsymbol{0} \\ \boldsymbol{H}_{PP}^{-1} \boldsymbol{H}_{CP}^{\mathrm{T}} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{h}_C \\ \boldsymbol{h}_P \end{pmatrix} = \begin{pmatrix} -\boldsymbol{J}_C^{\mathrm{T}} \boldsymbol{\epsilon} + \boldsymbol{H}_{CP} \boldsymbol{H}_{PP}^{-1} \boldsymbol{J}_P^{\mathrm{T}} \boldsymbol{\epsilon} \\ -\boldsymbol{H}_{PP}^{-1} \boldsymbol{J}_P^{\mathrm{T}} \boldsymbol{\epsilon} \end{pmatrix}. \tag{4.65}$$

The update vector $\boldsymbol{h}_C$ for the extrinsic camera parameters can be computed from the first row of this equation. Afterwards, it is possible to determine the update vector $\boldsymbol{h}_P$ for the feature positions from the second row of the equation. The most important aspect of an efficient implementation is to capitalize on the block structure of the matrix $\boldsymbol{H}_{PP}$.

4.2.4.5 Gauge Freedom

As we have already discussed in the first section of this chapter, there are properties of the reconstructed scene that cannot be estimated from the images of a video sequence alone. When the video sequence has been captured with a calibrated camera, the reconstruction has seven degrees of freedom, which correspond to the position and orientation of the world coordinate system and the global scale of the reconstruction. The freedom to arbitrarily choose these properties is called gauge freedom. Although failing to constrain the gauge of the reconstruction can lead to numerical problems in the solution of the normal equations, we did not encounter any such problems with our bundle adjustment approach. A more elaborate discussion of many aspects of gauge freedom can be found in [Tri99]. Furthermore, Bartoli presents an approach for gauge invariant bundle adjustment in [Bar03].

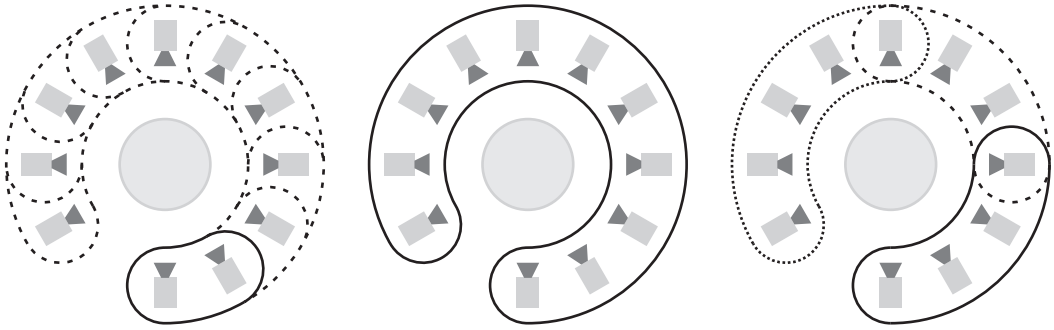


Figure 4.6: The three examples of this figure illustrate different strategies for the segmentation of an input image sequence.

4.3 The Structure and Motion Estimation System

This section contains a detailed description of our proposed structure and motion estimation system. In the first subsection, we give a short overview of our strategy for processing the input image sequence. In addition to the algorithms discussed in Section 4.2, our structure and motion estimation system employs two new algorithms. The key frame selection algorithm is described in Subsection 4.3.2, and the algorithm for estimating the relative scale factor of two independent reconstructions is detailed in Subsection 4.3.3. In Subsection 4.3.4, we present different approaches for increasing the robustness of the algorithms in our structure and motion estimation system. Finally, we closely examine the structure of our estimation system in Subsection 4.3.5. To this end, we show how the discussed algorithms interoperate with each other and review our design decisions with respect to the design goals stated in Subsection 4.1.1.

4.3.1 Overview

Choosing a strategy for the segmentation of the input image sequence is one of the most important design decisions for every structure and motion estimation system. We will substantiate our choice by analyzing the examples shown in Figure 4.6. The left example is typical of real-time systems that have a strong focus on low latency operation like [McL00, Lou05a, Eng06]. At first, an initial estimate of the scene structure and the camera motion is obtained from a small number of views. Then, the remaining views are processed one after the other as they are captured by the camera. Although this strategy is unrivaled when low latency is the most important requirement, it has several drawbacks for less specialized applications. First of all, the quality of the initial reconstruction often suffers from the limited amount of available input data. The resulting inaccuracy also affects the later processing stages, where the remaining views are merged with the initial reconstruction. In addition to that, many systems have to improve their accuracy by recomputing a

considerable part of the scene structure and the camera motion for every added view, which has a negative impact on their general computational efficiency.

The strategy in the middle of Figure 4.6 is the direct opposite of the first strategy. Its main goal is to process as many views as possible in one step. Prominent exponents of this strategy are the factorization algorithms discussed in Subsection 4.1.2. Using all or most of the input data simultaneously increases the potential accuracy of the reconstruction, which is one of the inherent strengths of this strategy. However, long image sequences often include a high percentage of incomplete feature trails. In the worst case, the first and the last view do not have any feature points in common. This poses severe problems to factorization algorithms as well as invariant-based algorithms. Consequently, some practical implementations pursue a hybrid strategy, which consists of reconstructing the largest possible subsequence first and adding the remaining views one by one afterwards [Sch04]. Unfortunately, this approach vitiates the advantages of both original strategies.

The segmentation strategy used in our structure and motion estimation system is illustrated in the right example of Figure 4.6. It partitions the image sequence into segments that are reconstructed independently. Although the performance of this strategy is mostly determined by the rules for computing the segmentation, it tends to provide a balanced combination of efficiency and accuracy. By allowing for the fact that it is usually not possible to reconstruct a complete image sequence in one step, it also gains a lot of versatility over the strategy in the middle of Figure 4.6. Furthermore, when highest accuracy is required, it is always possible to optimize the complete reconstruction with a bundle adjustment algorithm. Other structure and motion estimation systems employing this segmentation strategy are presented in [Shu99] and [Gib02].

The segmentation strategy of our estimation system strongly influences its basic approach for processing the input image sequence. As illustrated in Figure 4.7, our estimation system uses a three-step process. The first step is the segmentation of the input image sequence, which is controlled by the key frame selection algorithm described in Subsection 4.3.2. The second step consists of the independent reconstruction of the individual segments, which is discussed in Subsection 4.3.5. This subsection also contains a detailed explanation of the system structure and the technical specifications of our structure and motion estimation system. Finally, the reconstructed segments are merged into one reconstruction in the third step. We provide further information on this step in Subsection 4.3.3.

4.3.2 Key Frame Selection

The key frame selection algorithm described in this subsection computes the start frame and the end frame of every segment. In order to facilitate the merging of the reconstructed segments, adjacent segments have one key frame in common. Furthermore, the segmentation of the input image sequence has to possess two important properties. On the one hand, every segment must contain a certain num-

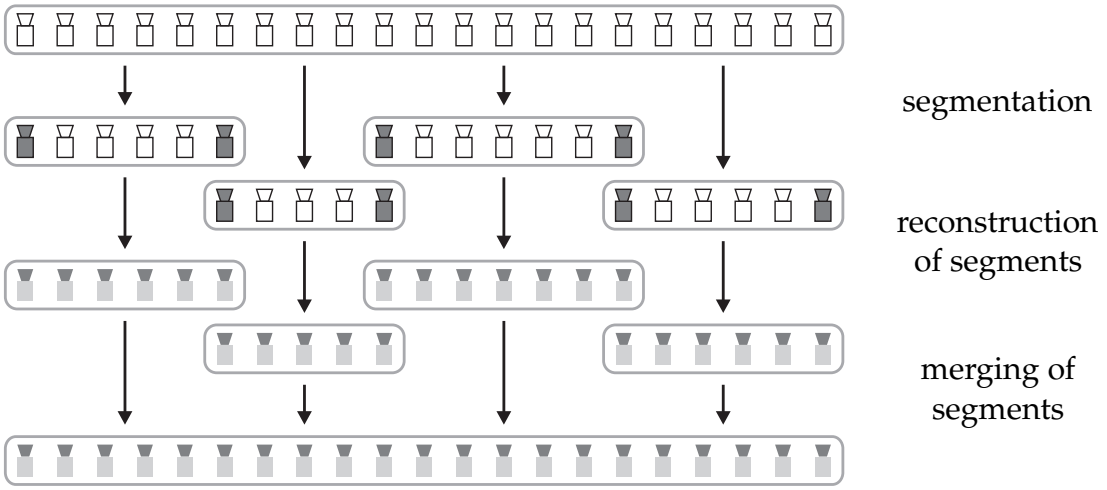


Figure 4.7: The basic approach of our structure and motion estimation system for processing the input image sequence consists of three distinct steps. In order to facilitate the merging of the segments in the third step, the segmentation ensures that adjacent segments have one view in common.

ber of feature points that are available in all frames of the segment. This property allows our reconstruction algorithms to process the segments as described in Subsection 4.3.5. On the other hand, every segment has to contain a sufficiently large baseline, which is the amount of camera translation perpendicular to the view rays from the optical center to the feature points. This is a vital requirement for the estimation of the camera motion with the five-point algorithm as well as the subsequent triangulation of the feature points.

Although the key frame selection algorithms in other structure and motion estimation systems often have to meet differing requirements, it is instructive to examine their quality criteria. The most basic quality criterion is given by the number of frames in the resulting segment. It is used in [Nis00] to induce a bias towards long segments, which are preferable to a larger number of small segments. Another quality criterion considers the number of feature points that are available both in the previous key frame and the current frame. This criterion is present in many key frame selection algorithms [Whi01, Gib02, Sai03, Rep05], where it is often expressed as the ratio between the number of lost features and the total number of features in the previous key frame.

Most key frame selection algorithms try to obtain a rough estimate of the scene configuration, which is then used to find key frames that facilitate the subsequent reconstruction of the segments. A simple measure for the camera motion is provided by the translation of the feature points [Whi01]. However, this criterion cannot discern camera rotation, which does not affect the reconstruction, and camera translation, which has to be sufficiently large for a successful reconstruction. A more reliable way of detecting the presence of camera translation is to estimate a

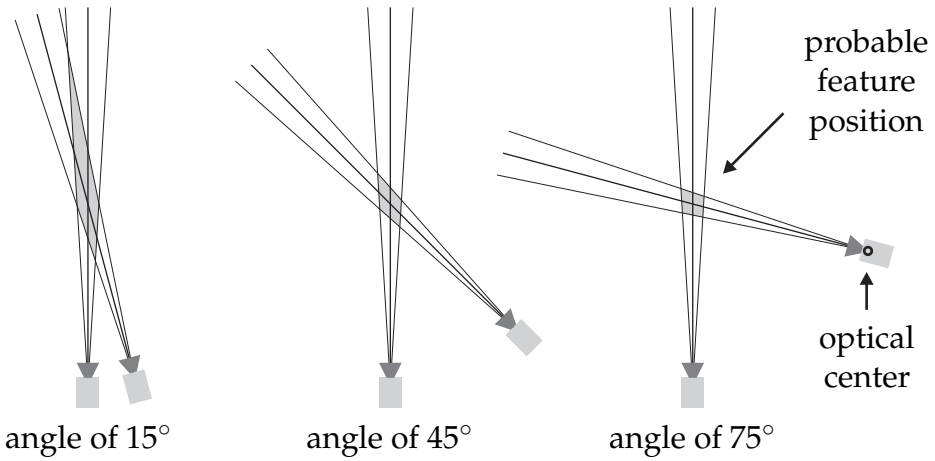


Figure 4.8: The accuracy of the triangulation of a feature point strongly depends on the angle between the view rays from the optical centers. The probable position of a triangulated feature point is defined by the intersection of two cones, which represent the uncertainty of the positions of the tracked feature points in the image plane.

planar projective homography for the corresponding feature points of two frames [Sai03]. Feature data that accurately conforms to the homographic model can either be explained by a dominantly planar scene structure or by the absence of camera translation. In both cases, the concerned frame is not well suited as a key frame. An additional criterion for double-checking the scene configuration is the epipolar model represented by the fundamental matrix, which should be able to fit all correctly tracked feature points for all scene configurations [Gib02]. The Geometric Robust Information Criterion (GRIC) model selection approach is used to perform a quantitative comparison of the homographic model and the epipolar model in [Rep05]. While the presented criteria are primarily used to avoid degenerate scene configurations, the approach in [Tho04] expressly selects the best key frames with respect to the estimation error of the reconstruction. Although the publication contains no information about the efficiency of the proposed approach, its description suggests that it is very time-consuming.

All of the quality criteria discussed in the preceding paragraph are generally applied to uncalibrated image sequences. In contrast to this, our structure and motion estimation system only works with calibrated image sequences. This fact strongly affects the requirements of our key frame selection algorithm. First, we do not have to detect a possible planarity of the scene structure, because our algorithms are able to cope with this configuration. Furthermore, it allows us to efficiently compute a Euclidean reconstruction of two views with the five-point algorithm. The general inability to recover the global scale factor makes it impossible to compare the camera translations of different view pairs directly. Thus, we use the angle of the view rays from the optical centers to the available feature points instead. As the angles are faithfully recovered in the Euclidean framework, they can easily be compared

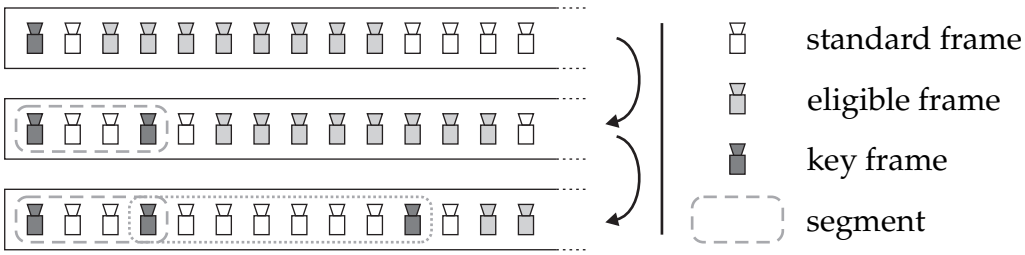


Figure 4.9: The exemplary computation of the first two segments of an image sequence illustrates the basic approach of our key frame selection algorithm.

for different view pairs. Under the assumption that the distance of the camera from the scene is roughly constant, the median angle of the view rays is a good way to judge the potential accuracy of a reconstruction. This proposition is further substantiated in Figure 4.8.

Figure 4.9 illustrates the basic approach of our key frame selection algorithm. The first frame of the input image sequence is automatically selected as the first key frame, because the resulting segmentation has to cover the whole image sequence. The second key frame is both the last frame of the first segment and the first frame of the second segment. Its position is confined by two user-defined parameters, which specify the minimum number $\delta_{\text{frm\_min}}$ and the maximum number $\delta_{\text{frm\_max}}$ of frames in a segment. For the purpose of illustration, these parameters are set to values of three and ten in Figure 4.9. The default values of these two parameters in our structure and motion estimation system are discussed in Subsection 4.3.5.

Our key frame selection algorithm computes the last frame $\hat{m}_2$ of every new segment by maximizing a quality measure q_o for the segment defined by the previous key frame $\hat{m}_1$ and one frame m_2 of the set $\mathcal{F}$ of eligible frames

$$\hat{m}_2 = \underset{m_2 \in \mathcal{F}}{\operatorname{argmax}}\, q_o(\hat{m}_1, m_2)\,. \tag{4.66}$$

The overall quality measure is the product of three separate quality measures

$$q_o(m_1, m_2) = q_{\text{angles}}(m_1, m_2)\; q_{\text{frames}}(m_1, m_2)\; q_{\text{trails}}(m_1, m_2)\,. \tag{4.67}$$

The first quality measure $q_{\text{angles}}(m_1, m_2)$ is defined as the median of the sines of the view ray angles of the feature points available in the frames m_1 and m_2. It is computed with a robust version of the five-point algorithm, which is described in Subsection 4.3.4. The sine of the angle is used to obtain a value between zero and one. Furthermore, it introduces a non-linear mapping that decreases the probability that key frames resulting in very small view ray angles are selected.

The second quality measure $q_{\text{frames}}(m_1, m_2)$ is based on the ratio between the number of frames in the segment and the maximum number of frames

$$q_{\text{frames}}(m_1, m_2) = 1 - \left(\frac{m_2 - m_1 + 1}{\delta_{\text{frm\_max}}} - \frac{1}{2} \right)^2 . \tag{4.68}$$

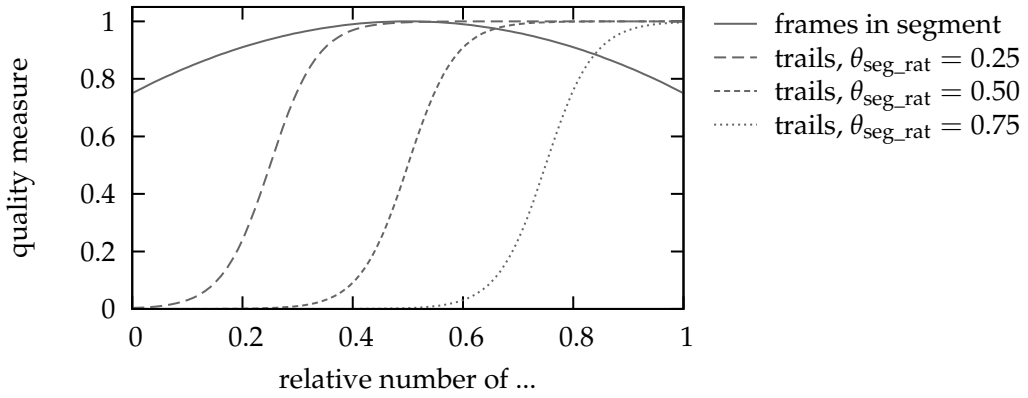


Figure 4.10: This graph visualizes the shape of the quality measures $q_{\text{frames}}(\cdot)$ and $q_{\text{trails}}(\cdot)$ with respect to the corresponding ratios. The minimum relative number of complete feature trails in a segment can be adjusted with the parameter $\theta_{\text{seg\_rat}}$.

As illustrated in Figure 4.10, this quality measure introduces a small bias towards segments of medium length. The third quality measure

$$q_{\text{trails}}(m_1, m_2) = \left(1 + 10^{-10\left(N_{\text{ratio}}(m_1,m_2) - \theta_{\text{seg\_rat}}\right)}\right)^{-1} \tag{4.69}$$

is based on the ratio

$$N_{\text{ratio}}(m_1, m_2) = \frac{N_{\text{fp}}(m_1, m_2)}{N_{\text{fp}}(m_1, m_1)} \tag{4.70}$$

between the number $N_{\text{fp}}(m_1, m_2)$ of feature points available in all frames from m_1 to m_2 and the number $N_{\text{fp}}(m_1, m_1)$ of feature points in frame m_1. This ratio is monotonically decreasing with the distance from m_2 to m_1. Figure 4.10 illustrates that the quality measure $q_{\text{trails}}(m_1, m_2)$ rapidly diminishes when the ratio $N_{\text{ratio}}(m_1, m_2)$ approaches the user-defined parameter $\theta_{\text{seg\_rat}}$ from above. The purpose of the third quality measure $q_{\text{trails}}(m_1, m_2)$ is to ensure that the number of complete feature trails in every segment is sufficiently large for a robust and accurate reconstruction. At the same time, it has almost no impact on the key frame selection when the ratio $N_{\text{ratio}}(m_1, m_2)$ is much larger than $\theta_{\text{seg\_rat}}$. The default value of $\theta_{\text{seg\_rat}}$ in our structure and motion estimation system is 0.25.

In summary, our key frame selection algorithm sequentially processes the input image sequence by alternating two operations. Given the last computed key frame, the algorithm determines the set of eligible key frames by limiting the size of the resulting new segment. From this set, it selects the frame with the highest overall quality measure as the next key frame. When our key frame selection algorithm reaches the end of the input image sequence, the following two distinct problems can occur:

- The last key frame does not coincide with the last frame of the image sequence, but there are not enough remaining frames for another segment.

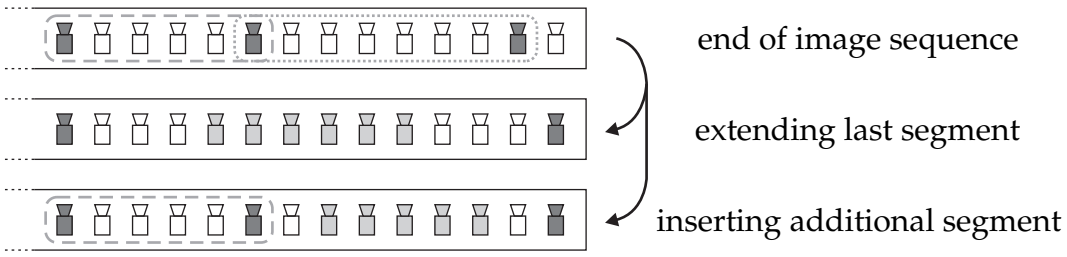


Figure 4.11: When the end of the input image sequence has been reached, a post-processing step in our key frame selection algorithm ensures that the segmentation covers the whole sequence. To this end, our algorithm either extends and resizes the last segment or inserts an additional segment. The meaning of the different camera symbols is explained in Figure 4.9.

- The last key frame coincides with the last frame of the image sequence, but the value of the quality measure of the final segment is very small.

Our key frame selection algorithm tackles these problems in a post-processing step that explores the two alternative solutions illustrated in Figure 4.11. First, the last segment is extended to the last frame of the sequence, and the second to last key frame is repositioned by maximizing the minimum of the overall quality measures of the last two segments. Second, an additional key frame is used to split the frames after the second to last segment into two segments. In this case, the position of the additional key frame is determined by maximizing the minimum of the overall quality measures of the last three segments. Finally, the alternative solution yielding the better one of the two maximized quality measures is selected.

During the reconstruction of the segments, which is described in Subsection 4.3.5, the solutions of the five-point algorithm for the relative camera pose of the two outer key frames of a segment are disambiguated by applying the three-point algorithm to a third key frame. For a segment with the outer key frames $\hat{m}_1$ and $\hat{m}_2$, the intermediate key frame $\hat{m}_3$ is computed according to

$$\hat{m}_3 = \underset{m_3 \in \mathcal{F}_{\text{mid}}}{\operatorname{argmax}} \; q_{\text{mid}}(\hat{m}_1, \hat{m}_2, m_3)\,, \quad \mathcal{F}_{\text{mid}} = \{\hat{m}_1 + 1,\; \ldots,\; \hat{m}_2 - 1\}\,, \tag{4.71}$$

where $q_{\text{mid}}(\cdot)$ is defined as

$$q_{\text{mid}}(m_1.m_2, m_3) = \; \min(q_{\text{angles}}(m_1, m_3),\, q_{\text{angles}}(m_3, m_2))\,. \tag{4.72}$$

The quality measure $q_{\text{mid}}(\cdot)$ prevents that the camera position in the intermediate key frame lies too close to the camera position of either of the outer key frames. For a large number of frame pairs m_1 and m_2, the quality measure $q_{\text{angles}}(\cdot)$ is considered more than once during the key frame selection process. Consequently, we store the computed values of $q_{\text{angles}}(\cdot)$ in a temporary data cache to avoid unnecessary recomputations.

For applications that require very high efficiency, the computation of the median view ray angles required for the quality measure $q_{\text{angles}}(\cdot)$ can be deactivated with the user-defined parameter $\delta_{\text{seg\_vra}}$. The modified overall quality measure is then given by

$$\tilde{q}_o(m_1, m_2) = q_{\text{frames}}(m_1, m_2)\, q_{\text{trails}}(m_1, m_2) \tag{4.73}$$

and the intermediate key frames are determined as

$$\hat{m}_3 = (\hat{m}_1 + \hat{m}_2)/2\,. \tag{4.74}$$

However, this approach is only recommended when the configuration of the scene structure and the camera motion is known to be unproblematic. We evaluate the performance of our key frame selection algorithm both with and without the computation of the median view ray angles in Section 6.2.

4.3.3 Merging of Segments

Our key frame selection algorithm partitions the image sequence into segments, which are reconstructed independently as described in Subsection 4.3.5. Afterwards, the reconstructed segments have to be merged into a complete reconstruction. One important aspect of our approach for the merging of the segments is the organization of the merging process. Furthermore, the merging itself can be divided into two distinct steps. In the first step, the two considered segments are placed into one common coordinate system. In the second step, the final feature positions are determined from the possibly inconsistent estimates in the reconstructed segments.

Our approach for the merging of the segments is sequential. At first, the complete reconstruction is initialized with the first segment. Then, the remaining segments are appended consecutively to this reconstruction. This approach makes it possible to efficiently optimize the data of every merged segment as described at the end of Subsection 4.3.5. As the segments are fully reconstructed before the merging takes place, processing them in a different order does not provide any significant advantage. More sophisticated approaches, like the hierarchical approach proposed in [Nis00], are only beneficial when the reconstruction and the merging of the segments are closely coupled.

The first step of the merging process is to place the two reconstructions into a common coordinate system. For Euclidean reconstructions, this problem requires the estimation of a rotation, a translation, and a relative scale factor. Our solution to this problem is detailed in Figure 4.12. It estimates the rotation and the translation by aligning the common view of the two reconstructions. This reduces the problem to the estimation of the relative scale of the reconstructions. We estimate the relative scale factor by considering the feature points that are available in both reconstructions. Incidentally, it is also possible to compute the complete alignment with these feature points alone. However, in this case the two sets of extrinsic camera parameters of the shared view are not guaranteed to coincide and the obtained final result is not necessarily better.

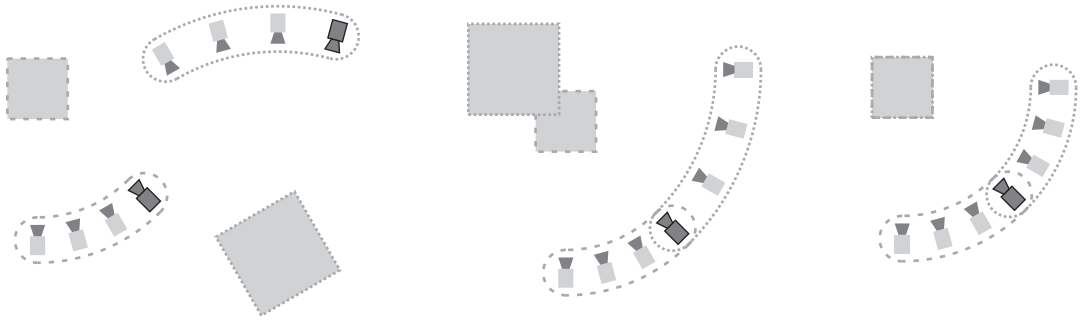


Figure 4.12: This figure illustrates our basic approach for placing two independently reconstructed segments into a common coordinate system. The relative motion between the two segments is defined by a rotation, a translation, and a relative scale factor. As our key frame selection algorithm ensures that adjacent segments have one view in common, we can estimate the rotation and the translation between the two segments by aligning this view. After that, we estimate the relative scale factor with the help of the feature points available in both segments.

We prepare the estimation of the relative scale factor by transforming all feature points n available in both the reconstruction r and the reconstructed segment t into the camera coordinate system of the shared view m

$$\boldsymbol{c}_{r,mn} = \boldsymbol{R}_{r,m}\,\boldsymbol{p}_{r,n} + \boldsymbol{t}_{r,m}\;, \quad \boldsymbol{c}_{t,mn} = \boldsymbol{R}_{t,m}\,\boldsymbol{p}_{t,n} + \boldsymbol{t}_{t,m}\;. \tag{4.75}$$

The relative scale factor $\hat{s}_m$ of the reconstructions with the shared key frame m is then computed by minimizing the sum of the squared distances of the common feature points n

$$\hat{s}_m = \underset{s_m}{\operatorname{argmin}} \sum_n \|\boldsymbol{c}_{r,mn} - s_m\,\boldsymbol{c}_{t,mn}\|^2 = \frac{\sum_n \boldsymbol{c}_{r,mn}^{\mathrm{T}}\boldsymbol{c}_{t,mn}}{\sum_n \boldsymbol{c}_{t,mn}^{\mathrm{T}}\boldsymbol{c}_{t,mn}}\;. \tag{4.76}$$

In order to decrease the influence of the outliers among the feature points, our system uses a robust version of this algorithm, which is further described in Subsection 4.3.4. After the estimation of the relative scale factor, the triangulated feature points of the reconstructed segment t can be transformed into the world coordinate system of the reconstruction r according to

$$\hat{\boldsymbol{p}}_{t,n} = \boldsymbol{R}_{r,m}^{\mathrm{T}}\,\boldsymbol{R}_{t,m}\,\hat{s}_m\,\boldsymbol{p}_{t,n} + \boldsymbol{R}_{r,m}^{\mathrm{T}}\,(\hat{s}_m\,\boldsymbol{t}_{t,m} - \boldsymbol{t}_{r,m})\,. \tag{4.77}$$

Finally, the extrinsic camera parameters of the views l in the segment t have to be transformed into the coordinate system of the reconstruction r. The new rotation matrices $\boldsymbol{R}_{r,l} = \hat{\boldsymbol{R}}_{t,l}$ are given by

$$\hat{\boldsymbol{R}}_{t,l} = \boldsymbol{R}_{t,l}\,\boldsymbol{R}_{t,m}^{\mathrm{T}}\,\boldsymbol{R}_{r,m}\;, \quad l > m\;, \tag{4.78}$$

and the new translation vectors $\boldsymbol{t}_{r,l} = \hat{\boldsymbol{t}}_{t,l}$ are given by

$$\begin{aligned} \hat{\boldsymbol{t}}_{t,l} &= \hat{s}_m \, \boldsymbol{t}_{t,l} - \hat{\boldsymbol{R}}_{t,l} \, \boldsymbol{R}_{r,m}^{\mathrm{T}} \left(\hat{s}_m \, \boldsymbol{t}_{t,m} - \boldsymbol{t}_{r,m} \right) \\ &= \hat{s}_m \, \boldsymbol{t}_{t,l} - \boldsymbol{R}_{t,l} \, \boldsymbol{R}_{t,m}^{\mathrm{T}} \left(\hat{s}_m \, \boldsymbol{t}_{t,m} - \boldsymbol{t}_{r,m} \right) , \quad l > m \, . \end{aligned} \tag{4.79}$$

As the transformed extrinsic camera parameters are unique, they can simply be copied into the merged reconstruction.

The second step of the merging is concerned with the processing of the possibly inconsistent feature positions. Feature points that are only available in one segment have exactly one triangulated position, which can be copied into the complete reconstruction after its transformation into the correct world coordinate system. In contrast to this, feature points that are available both in the reconstruction r and the reconstructed segment t can have two inconsistent positions $\boldsymbol{p}_{r,n}$ and $\hat{\boldsymbol{p}}_{t,n}$. Our solution to this problem is to select the position yielding the smaller median squared back-projection error with respect to all available views in the merged reconstruction. Thus, we back-project the reconstructed feature point positions into image coordinates according to (2.2) and (2.3). For one feature point n and one view m, the squared back-projection error is defined as

$$\epsilon_{\mathrm{bp},mn} = \| \boldsymbol{q}_{mn} - \vec{\boldsymbol{q}}_{mn} \|^2 , \tag{4.80}$$

where $\boldsymbol{q}_{mn}$ is the observed position of the feature point and $\vec{\boldsymbol{q}}_{mn}$ is the back-projected position of the feature point. The most promising alternative solution is to retriangulate the feature points with all available views in the merged reconstruction, which is less efficient but more accurate. In our view, the first solution is more versatile, because the accuracy of the feature positions can easily be improved by activating one of the optional bundle adjustment approaches described in Subsection 4.3.5, whereas the higher computational cost of the alternative solution cannot be recovered.

4.3.4 Robust Estimation

Although robustness to outliers is one of the most important design goals for our structure and motion estimation system, the algorithms presented in Section 4.2 are not specifically prepared to cope with outliers in their input data. In order to remedy this obvious shortcoming, we employ two common techniques for robust estimation, which are known as least median of squares (LMedS) and M-estimators. In addition to an overview of our outlier handling strategy, this subsection contains a short description of both techniques, which is based on both [Zha97] and [Ste99]. We also explain how these techniques are applied to the algorithms in our structure and motion estimation system.

4.3.4.1 Basic Approach

In the description of our feature point tracking system in Chapter 3, the term outlier is exclusively used for erroneous feature positions that are part of the output of the tracking system. In contrast to this, outliers can occur in the input data, the intermediate results, and the output data of our structure and motion estimation system. Our outlier handling strategy is based on these rules:

- All algorithms in our system have to be sufficiently robust against outliers in their respective input data.
- Outliers detected by one algorithm are not deleted from the input data of subsequent algorithms.
- Outliers in the output data are neither detected nor deleted.

Both the second and third rule seem counterproductive at first glance. In order to justify these rules, we take a closer look at the conditions in our structure and motion estimation system.

There are several different configurations for incorrect feature positions in a feature trail. Single outliers can be caused by a single blurred image, whereas consecutive outliers occur when the feature tracker completely loses the correct feature point. When the feature tracker flip-flops between two similar feature points, even alternating patterns of inliers and outliers are possible. During the segmentation of the input image sequence, a feature trail is possibly divided into two or more parts. Consequently, the algorithms performing the reconstruction of the segments do not have enough information to decide between inliers and outliers, as a large part of the feature trail might lie outside the current segment. In addition to that, removing feature positions as outliers leads to non-contiguous feature trails, which considerably complicate the implementation of the reconstruction framework. Finally, leaving potential outliers in the input data gives all algorithms the chance to consider all the available data.

In contrast to our feature point tracking system, our structure and motion estimation system does not screen the output data for outliers. In our opinion, leaving this task to the algorithms processing the results of the structure and motion estimation provides a higher versatility, because the requirements of these algorithms for the accuracy and the completeness of the provided data can differ substantially. This approach is possible because the camera parameters, the feature positions, and the feature trails contain all the information necessary for a successful outlier detection. This is also the main difference to our feature point tracking system, where the actual image sequence is usually deleted after the tracking, which renders a later outlier detection impossible.

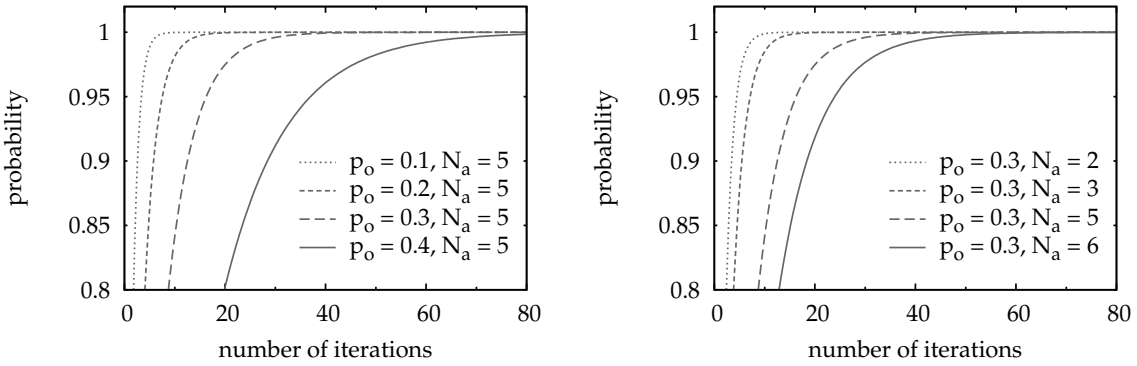


Figure 4.13: The probability of finding at least one set of good samples with a random sampling technique depends on the fraction p_o of outliers, the size N_a of the sample set, and the number of iterations N_i.

4.3.4.2 Least Median of Squares

We use the least median of squares technique to robustify several algorithms in our structure and motion estimation system. Its basic idea is to minimize the median of the squared residuals

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \underset{i \in \{1,\ldots,N_s\}}{\operatorname{med}} \epsilon_i(\boldsymbol{\phi})^2. \tag{4.81}$$

The definition of the parameter vector $\boldsymbol{\phi}$ and the cost function $\epsilon_i(\boldsymbol{\phi})$ is specific to the robustified algorithm, whereas the size N_s of the sample set depends on the input data. We complement the following introduction of the least median of squares technique with a discussion of its application to the algorithms in our estimation system further below.

Robustness is often measured with the breakdown point, which specifies the minimum fraction of outliers that can cause the estimate to diverge arbitrarily far from the correct estimate. The least median of squares technique has a breakdown point of 0.5, because its objective function ignores the values of the worse half of the residuals. Unfortunately, the use of the median in the objective function prevents a closed form solution for most problems. As a consequence, practical implementations rely on a random sampling technique for searching the parameter space. When the fraction of outliers in the input data is p_o and the number of random sampling iterations is N_i, the probability of finding at least one set of N_a good samples is given by

$$p_{\mathrm{rs}}(p_o, N_i, N_a) = 1 - (1 - (1 - p_o)^{N_a})^{N_i}. \tag{4.82}$$

The number of random sampling iterations is commonly selected by asserting a maximum value for the fraction of outliers and specifying a minimum value for the probability p_{rs}. As random sampling techniques are generally unable to guarantee that at least one set of good samples is found, the specified minimum probability has to be smaller than 1.0. Consequently, typical values lie between 0.99 and 1.0.

input: sample set $\mathcal{S}_s$ with $\lvert\mathcal{S}_s\rvert = N_s$
FOR $j \in \{1, \ldots, N_i\}$
randomly select N_a samples from $\mathcal{S}_s$ into sample set $\mathcal{S}_j$
compute parameter vector $\boldsymbol{\phi}_j$ for sample set $\mathcal{S}_j$
compute squared residuals $\epsilon_i(\boldsymbol{\phi}_j)^2$ for all samples in $\mathcal{S}_s$
compute median of squared residuals
select best parameter vector $\tilde{\boldsymbol{\phi}}$ according to (4.83)
estimate standard deviation $\sigma(\tilde{\boldsymbol{\phi}})$ of residuals according to (4.85)
compute sample set of inliers $\mathcal{S}^*$ according to (4.86)
compute final parameter vector $\hat{\boldsymbol{\phi}}$ for sample set $\mathcal{S}^*$

Figure 4.14: The basic approach of the least median of squares technique

Figure 4.13 shows the values of p_{rs} as a function of the number N_i of iterations for several combinations of the fraction p_o of outliers and the size N_a of the sample set. It can clearly be seen that the number of iterations required to obtain a specified minimum probability increases both with the fraction of outliers and the size of the sample set.

The basic approach of our implementation of the least median of squares technique is illustrated in Figure 4.14. During the random sampling, the minimum number N_a of samples required for a successful estimation of the parameter vector $\boldsymbol{\phi}$ is randomly drawn from the input data. After the estimation of the parameter vector $\boldsymbol{\phi}_j$ for the sample set $\mathcal{S}_j$, the median of the squared residuals is computed for all samples in the input sample set $\mathcal{S}_s$. Finally, the parameter vector $\tilde{\boldsymbol{\phi}}$ yielding the smallest median error is selected according to

$$\tilde{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}_j}{\operatorname{argmin}} \underset{i \in \{1,\ldots,N_s\}}{\operatorname{med}} \epsilon_i(\boldsymbol{\phi}_j)^2. \tag{4.83}$$

As the least median of squares technique uses a minimal subset of samples for estimating the parameter vector, its accuracy in the presence of noise in the input data is rather low. Consequently, the estimated parameter vector $\tilde{\boldsymbol{\phi}}$ is commonly refined in an additional processing step. To this end, the standard deviation of the residuals can be estimated as

$$\tilde{\sigma}(\tilde{\boldsymbol{\phi}}) = 1.4826 \left(1 + \frac{5}{N_s - N_a}\right) \sqrt{\underset{i \in \{1,\ldots,N_s\}}{\operatorname{med}} \epsilon_i(\tilde{\boldsymbol{\phi}})^2}. \tag{4.84}$$

The derivation of this equation is explained in [Rou87, page 202]. In order to simplify the computation, we omit the finite sample correction factor in our implemen-

	N_a	N_i	sample	algorithm	cf.	refine	alt. ref.
segmentation	5	16	$3 \times \boldsymbol{q}$	five-point	4.2.3	-	-
outer motion	5	64	$3 \times \boldsymbol{q}$	five-point	4.2.3	five-point	bun. adj.
inner motion	3	32	$\boldsymbol{q}, \boldsymbol{p}$	three-point	4.2.2	POSIT	bun. adj.
triangulation	2	16	$2 \times \boldsymbol{q}$	midpoint	4.2.1	-	bun. adj.
relative scale	3	32	$2 \times \boldsymbol{p}$	(4.76)	4.3.3	(4.76)	-

Table 4.2: This table provides an overview of all applications of the least median of squares technique in our structure and motion estimation system. It details the number N_a of samples used for the computation of the parameter vector in the random sampling phase, as well as the default number N_i of random sampling iterations. For all applications in this table, one sample is defined by a varying number of positions of one feature point in world coordinates $\boldsymbol{p} \in \mathbb{R}^3$ and image coordinates $\boldsymbol{q} \in \mathbb{R}^2$. The last two columns list the supported algorithms for refining the parameter vector estimated in the random sampling phase. We give further details on the robustified algorithms in the text below.

tation and estimate the standard deviation as

$$\sigma(\tilde{\boldsymbol{\phi}}) = 1.4826 \sqrt{\underset{i \in \{1,\ldots,N_s\}}{\text{med}} \epsilon_i(\tilde{\boldsymbol{\phi}})^2}. \tag{4.85}$$

Input samples whose residuals are larger than the estimated standard deviation multiplied by a factor $\theta_{\text{out\_lms}}$ are treated as outliers and excluded from the final sample set

$$\mathcal{S}^* = \left\{ \mathcal{S}_s[i] \;\middle|\; 1 \leq i \leq |\mathcal{S}_s| \text{ and } \epsilon_i(\tilde{\boldsymbol{\phi}})^2 < (\theta_{\text{out\_lms}}\, \sigma(\tilde{\boldsymbol{\phi}}))^2 \right\}. \tag{4.86}$$

In accordance with the publications [Zha97] and [Ste99], our implementation uses the factor $\theta_{\text{out\_lms}} = 2.5$. The final solution of the least median of squares technique is given by the parameter vector $\hat{\boldsymbol{\phi}}$ computed for the sample set $\mathcal{S}^*$ of inliers. Interestingly, the computation of the final solution can also be expressed as a weighted least-squares problem

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\text{argmin}} \sum_i w_i\, \epsilon_i(\boldsymbol{\phi})^2, \quad w_i = \begin{cases} 1: & \epsilon_i(\tilde{\boldsymbol{\phi}})^2 \leq (\theta_{\text{out\_lms}}\, \sigma(\tilde{\boldsymbol{\phi}}))^2 \\ 0: & \text{otherwise} \end{cases}. \tag{4.87}$$

Thus, the practical implementation of the least median of squares technique is basically a least-squares estimation with integrated outlier rejection.

Table 4.2 gives an overview of all applications of the least median of squares technique in our structure and motion estimation system. In the following, we additionally provide a detailed explanation of the specific characteristics of each application. As described in Subsection 4.3.2, our key frame selection algorithm, which controls the segmentation of the image sequence, uses a robust version of the five-point algorithm to compute the quality measure $q_{\text{angles}}(m_1, m_2)$. In this context, one sample

is defined as the position of one feature point in three different frames in image coordinates. The three frames are the first frame, the last frame, and the middle frame of the tentative segment. The squared residual corresponding to one sample is given by the sum of the squared back-projection errors for all three views. The definition of a single squared back-projection error can be found in (4.80). In the random sampling phase, the relative pose of the camera for the first frame and the last frame is computed with the five-point algorithm. As described in Subsection 4.2.3, the five-point algorithm yields up to ten solutions for the relative pose. In order to disambiguate the solutions, we triangulate all available feature points with the midpoint algorithm as detailed in Subsection 4.2.1. We also use the triangulated positions of three of the five feature points of the minimal sample set to compute the absolute pose of the camera for the middle frame with the help of the three-point algorithm, which is presented in Subsection 4.2.2. Finally, we use the median of the sum of the squared back-projection errors of all three views both for the disambiguation of the multiple solutions and the selection of the best parameter vector of the random sampling phase. For our key frame selection algorithm, the efficiency of the described computation is more important than its accuracy. Therefore, we completely skip the refinement described in the last three lines of Figure 4.14 in this application of the least median of squares technique.

The task of the second application is to compute the relative pose of the camera for the outer key frames of every segment. Like the first application, it uses the three-point algorithm in conjunction with an additional key frame to disambiguate the solutions computed with the five-point algorithm. The selection of the three key frames is described in Subsection 4.3.2. In contrast to the first application, the accuracy of the relative pose estimation has a strong influence on the accuracy of the reconstruction. As a consequence, we refine the estimation by applying the five-point algorithm to all feature points in the set of inliers $\mathcal{S}^*$. As the five-point algorithm yields up to ten solutions, the disambiguation described above also has to be performed for the final solution.

For the computation of the inner motion, i. e., the absolute pose of the camera for all inner frames of a segment, we use the three-point algorithm in the random sampling phase of the least median of squares technique. The three-point algorithm is detailed in Subsection 4.2.2. The squared residual corresponding to one sample is given by the squared back-projection error of the respective feature point as defined in (4.80). The three-point algorithm cannot process more than three feature points. Thus, we have to use another algorithm for estimating the final solution. To this end, we use the POSIT algorithm with virtual reference point, which has been proposed in Subsection 4.2.2. However, according to [DeM95], certain configurations of the input data result in divergent behavior of the POSIT algorithm. What is more, the POSIT algorithm is not suitable for planar scenes. As a consequence, we only use the solution of the POSIT algorithm when this solution actually reduces the median of the squared back-projection errors of the available feature points.

The triangulation of the feature points is the fourth application of the least me-

dian of squares technique in Table 4.2. In the random sampling phase, the sample set consists of two feature positions in image coordinates. One squared residual is given by the squared back-projection error of the feature point in the respective view. In our structure and motion estimation system, the feature points are triangulated with the midpoint algorithm, which is described in Subsection 4.2.1. In the default case, we do not refine the triangulated feature positions by considering more than two views for the triangulation. However, Table 4.2 shows that our system supports an alternate approach for refining the final solution of the estimation of the outer motion, the inner motion, and the triangulation. When the alternate approach is activated with the user-specified parameter $\delta_{\text{bun\_type}}$, the standard algorithms for refining the final solution are replaced with a bundle adjustment approach. The basic principles of bundle adjustment are explained in Subsection 4.2.4. Further details on the application of bundle adjustment approaches in our structure and motion estimation system are presented in Subsection 4.3.5. Our experimental evaluation in Section 6.2 shows that the alternate approach increases both the accuracy of the reconstruction and the required computation time.

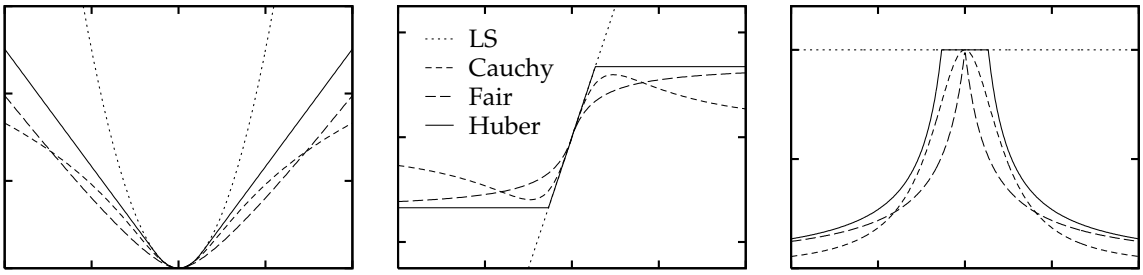
Our algorithm for the estimation of the relative scale of two reconstructions is defined by (4.75) and (4.76) in Subsection 4.3.3. It is the final application of the least median of squares technique in our structure and motion estimation system. One sample comprises the positions of one feature point in two different world coordinate system. Likewise, one squared residual is given by the squared distance of the two positions of one feature point after applying the scale factor s_m. Although it is possible to estimate the scale factor with a single feature point, we use three feature points in the random sampling phase, in order to increase the robustness of the algorithm against noise in the feature point coordinates. As shown in Table 4.2, the refinement of the scale factor is also performed with the algorithm used in the random sampling phase.

4.3.4.3 M-Estimators

The second robust estimation technique used in this work consists of a class of algorithms known as M-estimators. We use them exclusively to increase the robustness of the bundle adjustment approach described in Subsection 4.2.4. The basic idea of an M-estimator is to replace the square of the residuals by a different function $\rho(.)$ of the residuals as follows

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \sum_i \rho(\epsilon_i(\boldsymbol{\phi}))\,. \tag{4.88}$$

In order to improve the robustness of the original least-squares approach, the function $\rho(.)$ has to grow subquadratically. We will omit the parameter of the residual function in some of the following equations for better readability. For the solution of the optimization problem, we define the influence function $\psi(.)$ and the weight



residual function ρ influence function ψ weight function w

Figure 4.15: M-estimators are defined by their residual function ρ, which also determines the corresponding influence function and the weight function. In addition to the three implemented M-estimators, the graphs also illustrate the respective functions of the standard least-squares approach.

function $w(.)$ as

$$\psi(\epsilon_i) = \frac{\partial \rho(\epsilon_i)}{\partial \epsilon_i}\,, \quad w(\epsilon_i) = \frac{\psi(\epsilon_i)}{\epsilon_i}\,. \tag{4.89}$$

Using these definitions, we are able to reformulate the optimization problem in (4.88) as an iteratively reweighted least-squares problem

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \sum_i w(\check{\epsilon}_i(\check{\boldsymbol{\phi}}))\, \epsilon_i(\boldsymbol{\phi})^2\,, \tag{4.90}$$

where $\check{\epsilon}_i(\check{\boldsymbol{\phi}})$ represents the residuals computed in the previous iteration. This alternative formulation is derived in [Zha97].

A large number of different M-estimators have been suggested in the literature. We focus on three commonly used M-estimators with complementary properties. These three M-estimators are illustrated in Figure 4.15. Our first estimator is the Cauchy estimator, which is defined as

$$\rho_c(\epsilon_i) = \frac{1}{2}\, c_c^2\, \log\left(1 + \left(\frac{\epsilon_i}{c_c}\right)^2\right), \quad w_c(\epsilon_i) = \frac{1}{1 + (\epsilon_i / c_c)^2}\,. \tag{4.91}$$

It is the maximum likelihood estimator for the Cauchy distribution, which is a very heavy-tailed distribution. As the Cauchy estimator has a redescending influence function, it is especially robust against gross outliers. However, this type of influence function also tends to create false local minima, which makes an accurate initialization of the parameter vector very important. The other two estimators are the Fair estimator

$$\rho_f(\epsilon_i) = c_f^2 \left(\frac{|\epsilon_i|}{c_f} - \log\left(1 + \frac{|\epsilon_i|}{c_f}\right)\right), \quad w_f(\epsilon_i) = \frac{1}{1 + |\epsilon_i| / c_f} \tag{4.92}$$

and the Huber estimator

$$\rho_h(\epsilon_i) = \begin{cases} \epsilon_i^2/2 & : \ |\epsilon_i| \leq c_h \\ c_h\,(|\epsilon_i| - c_h/2) & : \ |\epsilon_i| > c_h \end{cases}, \quad w_h(\epsilon_i) = \begin{cases} 1 & : \ |\epsilon_i| \leq c_h \\ c_h/|\epsilon_i| & : \ |\epsilon_i| > c_h \end{cases}. \tag{4.93}$$

Both estimators are less robust to outliers than the Cauchy estimator, but they are also less prone to generating false local minima. The residual function of the Fair estimator has continuous derivatives of the first three orders, whereas the residual function of the Huber estimator has a discontinuous second derivative. However, this theoretical shortcoming rarely affects the convergence properties of this widely used estimator.

The integration of the M-estimators into our bundle adjustment approach makes a small number of adaptations necessary. As the Levenberg-Marquardt method is iterative, we apply the weights of the M-estimator in every Levenberg-Marquardt iteration, instead of solving the iteratively reweighted least-squares problem directly. All three M-estimators use one additional parameter, which is computed as the product of a tuning constant and the robust estimate of the standard deviation of the residuals

$$c_c = 2.3849\,\sigma(\check{\boldsymbol{\phi}})\,, \quad c_f = 1.3998\,\sigma(\check{\boldsymbol{\phi}})\,, \quad c_h = 1.345\,\sigma(\check{\boldsymbol{\phi}})\,. \tag{4.94}$$

The estimate of the standard deviation is defined in (4.85), whereas the tuning constants are explained in [Zha97]. As it is more difficult for the Levenberg-Marquardt algorithm to converge when the estimate of the standard deviation is adjusted in every iteration, the estimate of the standard deviation is fixed after $\delta_{\text{rob\_iter}}$ iterations. Our default value for this user-specified parameter is eight.

The standard cost function of our bundle adjustment approach considers the back-projection errors of all available feature positions. According to equation (4.46), the residual for one feature point n and one view m is defined as the back-projection error

$$\boldsymbol{\epsilon}_{mn}(\boldsymbol{\phi}) = (\boldsymbol{q}_{mn} - \vec{\boldsymbol{q}}_{mn}(\boldsymbol{\phi})) \in \mathbb{R}^2\,. \tag{4.95}$$

However, the two components of the back-projection error are strongly dependent. Consequently, it does not make sense to compute separate weights for them. In accordance with [Eng06], we use the norm of the back-projection errors for the computation of the weights required in (4.90)

$$\epsilon_i(\boldsymbol{\phi}) = \|\boldsymbol{q}_{mn} - \vec{\boldsymbol{q}}_{mn}(\boldsymbol{\phi})\|_2\,, \tag{4.96}$$

where the index i corresponds to feature point n in view m. Yet, the Levenberg-Marquardt optimization method requires the components of the back-projection errors for its processing. Thus, the weight function of the M-estimator has to be computed for the norm of the back-projection error, but must then be applied to the components of this error. This can be achieved by multiplying the components of the back-projection error with the square root of the weights computed for its norm

$$w(\check{\epsilon}_i(\check{\boldsymbol{\phi}}))\ \epsilon_i(\boldsymbol{\phi})^2 = \hat{\boldsymbol{\epsilon}}_{mn}(\boldsymbol{\phi})^{\mathrm{T}}\hat{\boldsymbol{\epsilon}}_{mn}(\boldsymbol{\phi})\,, \tag{4.97}$$

import feature trails (cf. text below)
select key frames and create segments (cf. Subs. 4.3.2)
FOR every segment $t \in \{1, \ldots, T\}$
compute relative pose for outer key frames $\hat{m}_1$ and $\hat{m}_2$ (cf. Subs. 4.2.3)
triangulate feature points using outer key frames (cf. text below)
FOR every inner frame $m \in \{\hat{m}_1 + 1, \ldots, \hat{m}_2 - 1\}$
compute absolute pose for frame m (cf. Subs. 4.2.2)
FOR every feature point n in segment t
triangulate feature point n (cf. Subs. 4.2.1)
(optionally) optimize segment t (cf. Subs. 4.2.4)
initialize current reconstruction with segment $t = 1$
FOR every segment $t \in \{2, \ldots, T\}$
merge segment t with current reconstruction (cf. Subs. 4.3.3)
(optionally) optimize merged segment t (cf. Subs. 4.2.4)
(optionally) optimize complete reconstruction (cf. Subs. 4.2.4)

Figure 4.16: This figure illustrates the structure of our structure and motion estimation system. The given references lead to the description of the standard algorithms for the respective problem. In addition to that, we provide a detailed explanation of the techniques for robust estimation used in our system in Subsection 4.3.4. Furthermore, an overview of our system can be found Subsection 4.3.1.

where

$$\hat{\boldsymbol{\epsilon}}_{mn}(\boldsymbol{\phi}) = \sqrt{w(\check{\epsilon}_i(\check{\boldsymbol{\phi}}))}\boldsymbol{\epsilon}_{mn}(\boldsymbol{\phi})\,. \tag{4.98}$$

The user-specified parameter $\delta_{\text{rob\_type}}$ allows the activation of one of the three implemented M-estimators for all applications of bundle adjustment in our structure and motion estimation system. While the additional computational cost per iteration is negligible, it is possible that this robust estimation technique moderately increases the number of required iterations. In contrast to this, the robustness to outliers is improved considerably. As a consequence, we recommend to activate one of the M-estimators whenever the input data is not completely free of outliers.

4.3.5 System Structure

In this subsection, we provide a detailed description of our structure and motion estimation system. To this end, its structure is illustrated in Figure 4.16. In addition to that, all user-defined parameters of our structure and motion estimation system are presented in Table 4.3. The input data of our estimation system consist of the

name	type	description	default
$\delta_{\text{len\_min}}$	trail import	minimum trail length	3
$\delta_{\text{frm\_min}}$	segmentation	minimum number of frames	5
$\delta_{\text{frm\_max}}$	segmentation	maximum number of frames	64
$\theta_{\text{seg\_rat}}$	segmentation	ratio of tracked / total trails	0.25
$\delta_{\text{seg\_vra}}$	segmentation	compute view ray angle	true
$\delta_{\text{seg\_iter}}$	segmentation	number of LMedS iterations	16
$\delta_{\text{out\_iter}}$	outer motion	number of LMedS iterations	128
$\delta_{\text{inn\_iter}}$	inner motion	number of LMedS iterations	64
$\delta_{\text{tri\_iter}}$	triangulation	number of LMedS iterations	32
$\delta_{\text{sca\_iter}}$	relative scale	number of LMedS iterations	64
$\delta_{\text{bun\_type}}$	bundle adjustment	bundle adjustment type	$*$
$\delta_{\text{rob\_type}}$	bundle adjustment	M-estimator type	$*$
$\delta_{\text{bun\_iter}}$	bundle adjustment	maximum number of iterations	32
$\delta_{\text{rob\_iter}}$	bundle adjustment	iterations with update of σ	8

Table 4.3: This table lists the parameters of our structure and motion estimation system. Its fourth column contains the default values of the parameters. Default values shown as $*$ are discussed in the text below.

2-D positions of tracked feature points in sensor coordinates and the intrinsic camera parameters of the camera that captured the image sequence. During the import of the feature trails, the intrinsic camera parameters are used to transform the coordinates of the feature positions from the sensor coordinate system to the image coordinate system. Furthermore, feature trails that are shorter than the minimum trail length $\delta_{\text{len\_min}}$ are deleted from the input data, because such feature trails often give rise to spurious 3-D feature points in the reconstruction. The preprocessed feature trails are the only input of the subsequent processing steps.

In the next step, the key frame selection and the creation of the segments are performed. Our key frame selection algorithm is detailed in Subsection 4.3.2. The application of the least median of squares technique to the estimation of the quality measure representing the median view ray angle between two tentative key frames is discussed in Subsection 4.3.4. Table 4.3 details the user-specified parameters for setting the number of random sampling iterations for every application of the least median of squares technique. After the key frame selection, the actual segmentation is performed by creating one data storage object with all relevant feature positions for every segment. This approach facilitates the independent reconstruction of the created segments.

The reconstruction of a segment t is split into several processing steps. First, the relative pose of the two outer frames of the segment is computed with the robust version of the five-point algorithm described in Subsection 4.3.4. As our approach for the disambiguation of the solutions of the five-point algorithm has to triangu-

late the available feature points anyway, we are able to omit the second processing step by storing the feature positions computed in the first processing step. As the quality of the result of this processing step determines the success of the whole reconstruction of the segment, we use a large default value for the number $\delta_{\text{out\_iter}}$ of the random sampling iterations of the least median of squares technique. The robust computation of the absolute pose for the inner frames is detailed in Subsection 4.3.4. Finally, all feature points are triangulated with a robust version of the midpoint algorithm as explained in Subsection 4.3.4. This subsection also contains detailed references to the descriptions of the standard algorithms in Section 4.2.

As discussed in Subsection 4.3.4, the user-specified parameter $\delta_{\text{bun\_type}}$ allows the activation of our bundle adjustment approach for refining the estimation of the relative pose, the absolute pose, and the triangulation of the feature points described above. For the estimation of the relative pose, the extrinsic camera parameters for the three key frames and the feature positions of all feature points available in all frames of the segment are optimized at once. In this case, the iterations of the Levenberg-Marquardt algorithm are limited by $\delta_{\text{bun\_iter}} = 32$ and $\delta_{\text{rob\_iter}} = 16$. For the estimation of the absolute pose, the extrinsic camera parameters of the inner frames are optimized independently using the feature positions identified as inliers in the random sampling phase of the least median of squares technique. Similarly, the feature positions are optimized independently using all tracked feature positions identified as inliers. As the optimized parameter vector is much smaller in the last two cases, we set $\delta_{\text{bun\_iter}} = 16$ and $\delta_{\text{rob\_iter}} = 8$. In all three cases, the bundle adjustment approach can be robustified by using the parameter $\delta_{\text{rob\_type}}$ to activate one of the three M-estimators described in Subsection 4.3.4

The parameter $\delta_{\text{bun\_type}}$ allows the independent activation of three further applications of our bundle adjustment approach. The first application constitutes the last processing step of the reconstruction of the segments. It optimizes all extrinsic camera parameters and all feature positions simultaneously. However, it is mainly useful for improving the success rate of the subsequent merging of the segments, especially when the input data contains a large percentage of outliers or represents a difficult scene configuration. Otherwise, it is preferable to activate the second application, which is performed directly after merging a segment with the data of the current reconstruction. Like the first application, it optimizes the camera parameters and feature positions of one segment at a time. In addition to that, it considers the reprojection errors of the currently optimized feature points in the frames of the preceding segments. Our experimental evaluation in Section 6.2 proves that this approach yields more accurate results, while being only marginally less efficient. The third and final application of bundle adjustment, which is the last processing step in Figure 4.16, optimizes the complete reconstruction and provides the highest accuracy. As we have stated in Subsection 4.2.4, the run-time complexity of bundle adjustment is governed by the factor $(6M)^3$, where M is the number of optimized views. Therefore, we recommend to activate this optimization only for short image sequences or when accuracy is more important than efficiency.

4.4 Summary

This chapter is concerned with the problem of structure and motion estimation. Its first section contains an overview of both the problem and the related work. In contrast to the previous chapter, where we basically enhance an existing system by adding, improving, and replacing algorithms, our contribution in this chapter mainly consists of combining existing algorithms into a completely new system. As a consequence, we first describe the employed algorithms independently in the second section of this chapter. Afterwards, we explain the assembly of our structure and motion estimation system in the third section.

Our main design philosophy is to build a versatile structure and motion estimation system by combining an efficient initial reconstruction with an accurate optimization. Thus, it is possible for the user to emphasize either accuracy or efficiency by deciding between the activation or deactivation of the optional optimization steps. Under all circumstances, our system should be able to successfully cope with a certain amount of outliers and the most common scene configurations. Our main contribution in the area of structure and motion estimation lies in the assembly of a versatile and robust estimation system from a number of carefully selected state-of-the-art algorithms. The performance of this system will be examined in Chapter 6. In addition to that, we enhanced existing and developed new algorithms. In this context, the extension of the POSIT algorithm with a virtual reference point and our new key frame selection algorithm are especially noteworthy.

Chapter 5

Point Set Registration

The basic task of point set registration is to align two point sets that represent at least partially overlapping parts of the same scene or object. In our approach for sparse 3-D reconstruction, these point sets are obtained as the output of the system for structure and motion estimation described in Chapter 4. The impossibility to determine the global scale of the reconstruction with a structure and motion estimation algorithm has already been discussed in Subsection 4.1.1. As a consequence, two independently reconstructed point sets are bound to have a different scale. Thus, their registration requires the estimation of the rigid motion and the relative scale that bring the two point sets into alignment.

In the first section of this chapter, we thoroughly examine the problem of point set registration and provide an overview of the existing approaches in this problem area. The second section contains a description of the iterative closest point (ICP) algorithm, which is the prevalent algorithm for point set registration in the literature. In addition to that, we also provide a classification of the existing extensions to this algorithm. Several techniques for improving the robustness of the ICP algorithm to outliers are discussed in the third section. In the fourth section of this chapter, we present our approach for integrating the estimation of the relative scale between the two point sets into the ICP algorithm. Section 5.5 comprises an illustration of the structure of our variant of the ICP algorithm as well as a discussion of its user-specified parameters. The short summary in the final section completes this chapter.

5.1 Overview

5.1.1 Problem Statement

We start our overview with a description of the ideal motion problem, which is a simplification of the general problem of point set registration. The input data of the ideal motion problem consists of two point sets, the data point set $\mathcal{A}$ and the model point set $\mathcal{B}$. These point sets are required to have the same number of points. Furthermore, there has to be a rigid motion which aligns the two point sets, so that

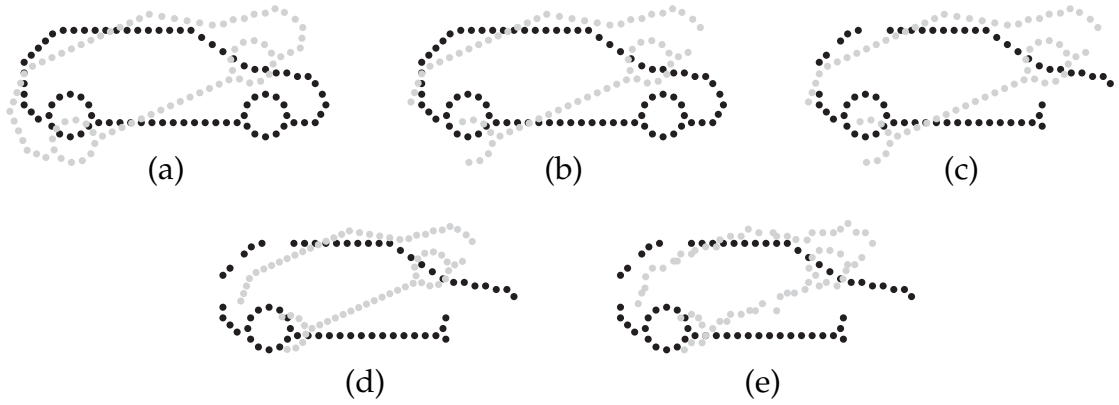


Figure 5.1: Two-dimensional examples illustrate the wide range of configurations for the input data of point set registration.

every point $\boldsymbol{a}_i \in \mathcal{A}$ is moved to a corresponding point $\boldsymbol{b}_j \in \mathcal{B}$:

$$\forall \boldsymbol{a}_i \in \mathcal{A} \quad \exists \boldsymbol{b}_j \in \mathcal{B} : \quad \boldsymbol{b}_j = \boldsymbol{R}\boldsymbol{a}_i + \boldsymbol{t} \,. \tag{5.1}$$

In this thesis, we only consider 3-D point sets, which means that

$$\begin{aligned} \boldsymbol{a}_i &\in \mathbb{R}^3, \quad i \in \{1, \ldots, |\mathcal{A}|\} \,, \\ \boldsymbol{b}_j &\in \mathbb{R}^3, \quad j \in \{1, \ldots, |\mathcal{B}|\} \,. \end{aligned} \tag{5.2}$$

Consequently, the rigid motion is defined by the rotation matrix $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ and the translation vector $\boldsymbol{t} \in \mathbb{R}^3$, which are invariably applied to the points of the data point set $\mathcal{A}$.

Figure 5.1 (a) represents an instance of the ideal motion problem. In practice, however, the assumptions of the ideal motion problem are rarely satisfied. When a partial reconstruction of an object is compared to a complete model of this object, one of the point sets is a subset of the other. This configuration is illustrated in Figure 5.1 (b). In this case, the number of points in the two point sets differs, and one point set contains points that have no corresponding point in the other point set. The registration of two partial reconstructions depicted in configuration (c) is conceptually similar, except that both point sets contain points that have no corresponding point in the other point set.

As we stated in the introduction of this chapter, two point sets generated by structure and motion estimation algorithms have an unknown relative scale, which has to be computed during their registration. This configuration is shown in Figure 5.1 (d). The two preceding chapters provide many reasons why the reconstruction of the 3-D points cannot be totally accurate. It is common knowledge that most other methods for obtaining 3-D point positions from real-world scenes are also affected by some level of noise. These issues are illustrated in Figure 5.1 (e). In addition to that, even a perfect reconstruction of the depth of a feature point does not guarantee

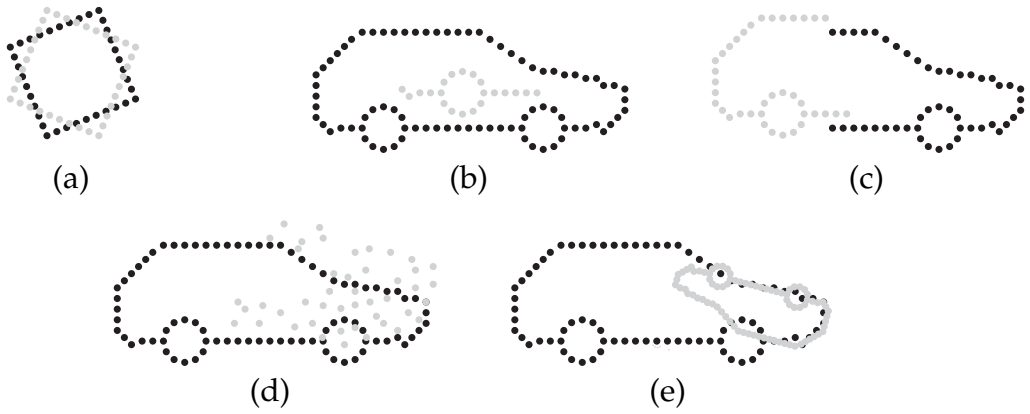


Figure 5.2: The two-dimensional examples in this figure demonstrate configurations that complicate or prevent successful point set registration.

that the same feature points were selected during the generation of the two point sets. The impact of this problem is reduced when the density of the feature points is increased.

In order to generalize the ideal motion problem as suggested by the configurations in Figure 5.1, we introduce the set of corresponding point pairs

$$\mathcal{C} = \{(i,j) \mid \boldsymbol{a}_i \in \mathcal{A} \text{ and } \boldsymbol{b}_j \in \mathcal{B} \text{ are corresponding points}\}\,. \tag{5.3}$$

Then, the task of point set registration is to compute the motion $(\boldsymbol{R}, \boldsymbol{t})$ that aligns the point pairs defined by $\mathcal{C}$. The additional computation of the relative scale s yields the motion $(\boldsymbol{R}, \boldsymbol{t}, s)$. It is discussed in Section 5.4. Due to the noise in the point positions, the computed motion cannot be expected to achieve a perfect alignment of the corresponding point pairs. Interestingly, there are straightforward solutions for computing the correspondences when the motion is known, and vice versa. Nevertheless, the simultaneous estimation of the correspondences and the motion is a difficult problem.

In the following, we discuss the basic assumptions for successful point set registration by examining the violations of these assumptions in Figure 5.2. The general shape of the point sets in example (a) does not allow the estimation of an unambiguous alignment. A similar case is shown in Figure 5.2 (b), where the shape of the smaller point set occurs in more than one location of the large point set. Example (c) illustrates that even two point sets with complex shapes can be insufficient when their overlap is too small. Finally, Figure 5.2 (d) demonstrates that a large overlap can also be insufficient for a successful alignment when high levels of noise distort the shape of the point sets. We subsume all four examples under the assumption that the shape of the overlapping region of the two point sets must be suitable for an unambiguous alignment.

The example in Figure 5.2 (e) illustrates that the registration of two point sets becomes difficult when the motion for aligning them is large. As the algorithms

discussed in this chapter are only locally convergent, they require a coarse pre-alignment of the input point sets. Although globally convergent algorithms are outside the scope of this work, a short overview of suitable pre-alignment techniques is given in the following subsection.

We employ the following performance criteria for the systematic comparison of different registration algorithms:

- The basin of convergence is an important criterion for locally convergent algorithms. In the context of point set registration, it measures the probability of finding the globally optimal registration subject to the magnitude of the initial motion. In principle, the basin of convergence determines if a pre-alignment of the point sets is necessary and how accurate this pre-alignment has to be. Unfortunately, for many registration algorithms, the basin of convergence strongly depends on the shape of the point sets and the particular direction of the motion.

- The computational efficiency of point set registration algorithms can easily be measured with the total computation time for a complete registration. For most algorithms, this time strongly depends on the number of points in the input point sets. Furthermore, the number of points in the input point sets is highly application-specific. Consequently, it is important to evaluate the computational efficiency of the algorithms with realistic point sets for every application.

- In the context of point set registration, outliers are points that do not have a corresponding point in the other point set. In contrast to the algorithms in the two preceding chapters, which encounter outliers that are either the result of a measurement error or an estimation error, point set registration algorithms additionally have to cope with outliers that are a direct consequence of partially overlapping point sets, even if the point sets themselves are completely devoid of erroneous data. As a consequence, the robustness to outliers is a very important performance criterion for point set registration.

- When a registration algorithm is sufficiently robust to successfully cope with the outliers in the input point sets, the accuracy of the estimated motion can be evaluated. This performance criterion is mainly used to measure how different levels of noise in the coordinates of the input points affect the estimated motion.

A small basin of convergence can be counteracted by using multiple starting positions, which increases the overall computation time. Furthermore, aggressive outlier elimination tends to increase robustness, while reducing the basin of convergence, at least for the algorithms discussed in [Phi07]. Our intended applications induce us to prioritize the robustness of our algorithms and to balance their performance with respect to the other criteria.

5.1.2 Related Work

There are several alternatives for categorizing the existing approaches for point set registration. One possible categorization distinguishes between globally and locally convergent algorithms. While globally convergent algorithms can estimate arbitrary motions, locally convergent algorithms require that the motion aligning the point sets lies inside their basin of convergence. A comprehensive selection of existing algorithms from both categories is presented and evaluated in [Sal07].

One common property of many globally convergent algorithms is their low accuracy. Consequently, these algorithms are often used to compute a rough pre-alignment for more accurate locally convergent algorithms. The framework proposed in [Li07] is a notable exception, because it promises both global convergence and high accuracy. As a downside, it is only applicable to small point sets and lacks robustness to outliers. Similarly, many globally convergent algorithms depend on special properties of the input data, which limits their general applicability. In the following, we shortly discuss some globally convergent algorithms to substantiate our allegations.

The pre-alignment algorithm proposed in [Chu98] is based on aligning the principal axes of the input point sets. Consequently, the accuracy of this algorithm strongly depends on the degree of overlap of the point sets. The RANSAC-based DARCES algorithm of [Che99a] employs a constrained exhaustive search. It is robust to partially overlapping point sets, but it is very susceptible to noisy input data. The genetic algorithm presented in [Lom06] estimates the overlap of the point sets during the registration. Its accuracy is sufficient for the intended task of performing a pre-alignment for a locally convergent algorithm. Other globally convergent algorithms find and match special features in the input data. For example, [Gel05] computes a feature descriptor for every input point. It requires the input data to contain densely sampled object surfaces. An extreme case of application-specific global registration is given by the pre-alignment algorithm in [Mur01], which extracts and matches the skeletons of underwater offshore structures.

As a consequence of our application requirements, we concentrate on locally convergent algorithms in the remainder of this chapter. The most popular algorithm in this category is the iterative closest point (ICP) algorithm, which was put forward by Besl and McKay in [Bes92]. At the same time, Chen and Medioni proposed a similar algorithm in [Che92]. The popularity of the ICP algorithm is demonstrated by the large number of extensions proposed for it in the literature. A comprehensive overview of existing extensions is given in [Rus01]. As our work is also based on the ICP algorithm, we provide a detailed discussion of this algorithm and its extensions in Section 5.2.

Recently, several locally convergent registration algorithms that are not based on the ICP algorithm have been proposed. Fitzgibbon employs the general-purpose Levenberg-Marquardt algorithm for point set registration in [Fit03]. Compared to the ICP algorithm, this approach is reported to have a larger basin of convergence

and higher efficiency when the input data is discretized in a preprocessing step. Other approaches are based on the combination of instantaneous kinematics and squared distance functions [Pot04], the optimization of a robust surface interpenetration measure with an enhanced genetic algorithm [Sil05], and the local optimization of a probabilistic cost function with a Nelder-Mead simplex search [Sha08]. However, it is difficult to judge the overall performance of these approaches from the application-specific experiments presented in the original papers.

Another categorization considers the different types of input data supported by the registration algorithms. The basic ICP algorithm works with unstructured point sets, which only contain the coordinates of the points. This is also the data representation used in our algorithms. Although range images are basically structured 3-D point sets, the implicitly provided neighborhood relations of the points can be used to approximate local surface patches and to compute local surface normals. Some variants of the ICP algorithm use this information to increase their computational efficiency [Che92] or their robustness to outliers [Pul99]. Sharp et al. compute invariant features from the range images and combine both positional and feature distances in [Sha02]. It is also possible to incorporate information about the intensity [Wei97] or the color [Dou06] of the points into the registration algorithm. Further data representations used in variants of the ICP algorithm include triangle meshes [Tur94] and parametric surfaces [Bis96].

Our final categorization is concerned with the number of input point sets. We focus on the registration of two point sets, and all algorithms discussed up to now belong to this category. The other category consists of algorithms for multiple input data sets. Although no dedicated overview of this category is known to us, [Pul99], [Hub03], and [Ben04] provide a thorough introduction to this problem area.

Point set registration has a surprisingly large number of applications. The intended applications for the algorithms presented in this subsection include the inspection of surfaces in manufacturing [Bis96], the computation of scene models for remotely operated underwater vehicles [Mur01], and the generation of digital 3-D models of complex real-life objects in the cultural heritage domain [Ben04]. Further interesting applications comprise the registration of human retinal images [Ste03], the comparison of scanned faces to a database of 3-D face models for face recognition [Lu04], and the registration of 3-D data reconstructed from endoscopic images to computer tomography data [Bur05].

5.2 The Iterative Closest Point Algorithm

5.2.1 Basic Principles

The ICP algorithm proposed by Besl and McKay in [Bes92] is a special-purpose nonlinear optimization algorithm for point set registration. Its input consists of a data point set $\mathcal{A}$ and a model point set $\mathcal{B}$, and the ICP algorithm estimates the motion

$(\boldsymbol{R}, \boldsymbol{t})$ aligning these point sets. As the ICP algorithm is only locally convergent, a successful registration requires that the motion to be estimated lies inside the basin of convergence of the algorithm.

The main loop of the iterative closest point algorithm consists of two steps. In the first step, the set of corresponding point pairs $\mathcal{C}$ is determined. To this end, the data points are transformed according to the current motion parameters $(\boldsymbol{R}, \boldsymbol{t})$, and the point pairs are created by searching for the closest model point for every transformed data point:

$$\forall (i,j) \in \mathcal{C} \;:\quad \boldsymbol{b}_j = \operatorname*{argmin}_{\boldsymbol{b}_h \in \mathcal{B}} \|\boldsymbol{b}_h - \boldsymbol{R}\boldsymbol{a}_i - \boldsymbol{t}\|^2. \tag{5.4}$$

This operation is performed with the help of a nearest neighbor algorithm. In the second step, the motion parameters that optimally align the current point pairs are estimated. The objective function of the respective optimization is defined as the sum of the squared intra-pair distances of the point pairs

$$\epsilon_{\text{icp}}(\boldsymbol{R}, \boldsymbol{t}) = \sum_{(i,j) \in \mathcal{C}} \|\boldsymbol{b}_j - \boldsymbol{R}\boldsymbol{a}_i - \boldsymbol{t}\|^2. \tag{5.5}$$

Consequently, the estimation of the motion parameters $(\boldsymbol{R}, \boldsymbol{t})$ can be expressed as the least-squares problem

$$(\hat{\boldsymbol{R}}, \hat{\boldsymbol{t}}) \;=\; \operatorname*{argmin}_{\boldsymbol{R},\boldsymbol{t}} \; \epsilon_{\text{icp}}(\boldsymbol{R}, \boldsymbol{t}) \;=\; \operatorname*{argmin}_{\boldsymbol{R},\boldsymbol{t}} \sum_{(i,j) \in \mathcal{C}} \|\boldsymbol{b}_j - \boldsymbol{R}\boldsymbol{a}_i - \boldsymbol{t}\|^2. \tag{5.6}$$

In the context of point set registration, the estimated minimum value of the objective function $\epsilon_{\text{icp}}(\hat{\boldsymbol{R}}, \hat{\boldsymbol{t}})$ is also called registration error. In order to avoid the accumulation of rounding errors, the motion parameters are not estimated incrementally, but for the original input point sets in every iteration. Finally, as an iterative algorithm, the ICP algorithm requires a stopping criterion. Besl and McKay propose to stop the algorithm when the reduction of the registration error falls below a specified threshold [Bes92].

One important property of the ICP algorithm is its guaranteed convergence to a local minimum. Besl and McKay argue in [Bes92] that this property can be proven by showing that the registration error ϵ_{icp} is both bounded below and monotonically decreasing. The first condition is easily satisfied, because ϵ_{icp} cannot fall below zero. For the second condition, it is instructive to examine the two operations that affect the registration error. When the point pairs are computed with the nearest neighbor algorithm, every data point either keeps its corresponding model point, or is assigned a new one, which has to be closer than the old one to be eligible. In any case, the registration error cannot increase during this operation. The second operation consists of the minimization of the registration error with respect to the motion parameters. This operation cannot increase the registration error either, because in this case the old motion parameters would represent a solution with a smaller registration error.

In the following, we describe further details of the ICP algorithm that are specific to our proposed variant. There are many different algorithms for the nearest neighbor search among the model points in the first step of the main loop. When the selection is limited to exact nearest neighbor algorithms, only the computational efficiency of the ICP algorithm is affected. However, this performance criterion can be affected very strongly, because nearest neighbor search with simple algorithms like exhaustive search is easily the most time-consuming operation of the ICP algorithm [Zin03b]. As a consequence, we employ an efficient k-D tree algorithm, whose search in the model point set has a computational complexity of $O(\log_2 |\mathcal{B}|)$ for every data point. Further details on our specific variant of the k-D tree algorithm lie outside the scope of this work, but can be found in [Zin03b].

Interestingly, the problem of motion estimation as defined in equation (5.6) is identical to the problem in the second step of the three-point algorithm for absolute camera pose estimation in Subsection 4.2.2. In order to prepare the discussion of integrated scale estimation in Section 5.4, we shortly reiterate the equations in (4.19) to (4.22) with updated notation. After computing the center of mass of the points

$$\bar{\boldsymbol{a}} = \frac{1}{|\mathcal{C}|} \sum_{(i,j) \in \mathcal{C}} \boldsymbol{a}_i \,, \quad \bar{\boldsymbol{b}} = \frac{1}{|\mathcal{C}|} \sum_{(i,j) \in \mathcal{C}} \boldsymbol{b}_j \,, \tag{5.7}$$

the point sets are centralized, which yields

$$\hat{\boldsymbol{R}} = \underset{\boldsymbol{R}}{\operatorname{argmin}} \sum_{(i,j) \in \mathcal{C}} \left\| \left(\boldsymbol{b}_j - \bar{\boldsymbol{b}}\right) - \boldsymbol{R} \left(\boldsymbol{a}_i - \bar{\boldsymbol{a}}\right) \right\|^2 . \tag{5.8}$$

This problem can be solved with a singular value decomposition of

$$\sum_{(i,j) \in \mathcal{C}} \left(\boldsymbol{b}_j - \bar{\boldsymbol{b}}\right) \left(\boldsymbol{a}_i - \bar{\boldsymbol{a}}\right)^{\mathrm{T}} = \boldsymbol{V} \boldsymbol{D} \boldsymbol{U}^{\mathrm{T}}, \tag{5.9}$$

which is used to compute the motion parameters as

$$\hat{\boldsymbol{R}} = \boldsymbol{V} \boldsymbol{U}^{\mathrm{T}} \,, \quad \hat{\boldsymbol{t}} = \bar{\boldsymbol{b}} - \hat{\boldsymbol{R}} \bar{\boldsymbol{a}} \,. \tag{5.10}$$

A negative determinant of the rotation matrix indicates that the true solution can be obtained by multiplying the third column of matrix $\boldsymbol{V}$ by -1. Further details on this algorithm and potential alternatives are presented in [Egg97].

Some techniques for robust estimation presented in Section 5.3 cancel the guaranteed convergence of the ICP algorithm, because they can cause the registration error to increase from one iteration to the next. With the standard stopping criterion of Besl and McKay, every occurrence of this behavior terminates the ICP algorithm. In order to avoid this problem in our variant of the ICP algorithm, we use two different stopping criteria. The first stopping criterion considers the norm of the differences between the current motion parameters and the motion parameters of the previous

iteration. If this norm falls below the user-specified threshold θ_{stop} for both the rotation matrix and the translation vector, our ICP algorithm is stopped. In addition to that, we use the maximum number δ_{iter} of iterations as a second stopping criterion, in order to prevent a divergent computation from causing an endless loop. Section 5.5 contains both an illustration of the complete structure of our variant of the ICP algorithm and further details on the respective user-specified parameters.

5.2.2 Existing Extensions

The persistent popularity of the ICP algorithm is responsible for the large number of existing extensions, which customize the algorithm for a wide range of different applications. As we focus on working with unstructured 3-D point sets, we limit our discussion in this subsection to extensions that are compatible with this type of input data. In order to provide a systematic overview, we distinguish between three categories of extensions:

- The first category is concerned with extensions that enhance the selection of control points, i. e., the selection of the first point of every point pair in the set of corresponding point pairs.
- Extensions of the second category alter the correspondence estimation, which determines the second point of every point pair. Additionally, we allocate extensions that remove unsuitable point pairs to this category.
- The third category comprises extensions of the motion estimation, which computes the motion parameters for the set of corresponding point pairs.

All extensions presented in this subsection can be integrated into our variant of the ICP algorithm. However, in order to concentrate our efforts on our main work areas, we decided to implement only the extensions described in the two subsequent sections.

The standard ICP algorithm does not perform an explicit selection of control points, but uses all data points as control points. For this approach, the amount of outliers in the set of corresponding point pairs can be reduced by using the point set with the higher proportion of overlap as data point set. When no information about the actual overlap is available, it is advisable to use the smaller point set as data point set. Especially for very large point sets, the computation time of the ICP algorithm can be reduced by using only a subset of the data points as control points. Furthermore, it is possible to improve the computational efficiency without impairing the accuracy by employing a hierarchy with different levels of subsampling as proposed in [Neu97]. Finally, another extension is to select control points from both input point sets. Small improvements of the basin of convergence are reported for this approach in [Rus01]. As a downside, searching for nearest neighbors in both point sets doubles the initialization cost of the nearest neighbor algorithm.

Although there are many extensions for improving the search for corresponding points, most of these extensions cannot be applied to unstructured 3-D point sets [Rus01]. The lack of additional information makes it difficult to outperform standard nearest neighbor algorithms for this kind of input data. Fitzgibbon proposes to precompute the nearest neighbors on a discretized grid around the model points, which is closely related to the approach of the distance transform [Fit03]. Although this extension increases the computational efficiency of the ICP algorithm, the involved discretization is also bound to impair its accuracy. In contrast to this, extensions that remove unsuitable point pairs from the set of corresponding point pairs are an important tool for increasing the robustness of the ICP algorithm. We thoroughly discuss these extensions in Section 5.3.

There are several starting points for extending the motion estimation of the ICP algorithm. It is possible to implicitly alter the cost function in equation (5.6) by introducing an additional weighting of the point pairs. Incidentally, the removal of point pairs described in Section 5.3 can also be interpreted as a weighting of the point pairs with a factor of either one or zero. Rusinkiewicz and Levoy report in [Rus01] that more elaborate weighting approaches do not significantly improve the performance of the ICP algorithm in their experiments.

Another starting point for extending the motion estimation of the ICP algorithm is to replace the direct solution for minimizing the cost function presented in Subsection 5.2.1. The use of an alternate cost function makes this approach inevitable when no direct solution of the resulting optimization problem exists. For instance, Blais and Levine employ a simulated annealing algorithm for minimizing their cost function in [Bla95]. However, this iterative optimization algorithm requires considerably more computation time than the direct solution [Rus01]. Besl and McKay propose to increase the convergence speed of the standard motion estimation algorithm by extrapolating the motion parameters from the estimation results of consecutive iterations [Bes92].

Finally, the motion estimation of the ICP algorithm can be extended by changing the underlying motion model. In view of our application requirements, we are especially interested in the additional estimation of the relative scale of the two input point sets. In contrast to this, more elaborate motion models, for example for non-rigid motions, lie outside the scope of our work. As a consequence, we dedicate Section 5.4 to the presentation of our extension for integrated scale estimation.

5.3 Robust Correspondence Estimation

As we have already motivated in Subsection 5.1.1, the robustness to outliers is a very important aspect of point set registration. For the standard ICP algorithm, however, even a single outlier can cause an arbitrarily large deviation from the correct registration. Consequently, several robust variants of the ICP algorithm have been proposed since its initial publication. Early exponents include the robust al-

gorithm presented by Masuda and Yokoya in [Mas95] and the RICP algorithm put forward in [Tru99]. For the sake of brevity, we refer to the former robust algorithm as the MICP algorithm in the following discussion.

The robustness of both the MICP algorithm and the RICP algorithm is increased with the least median of squares (LMedS) technique, which has already been discussed in Subsection 4.3.4. Its basic principle is to minimize the median of the squared residuals. In the context of the ICP algorithm, one residual represents the intra-pair distance of one point pair. Thus, the objective function defined in (5.5) is replaced with

$$\epsilon_{\text{lms}}(\boldsymbol{R}, \boldsymbol{t}) = \underset{(i,j) \in \mathcal{C}}{\text{med}} \|\boldsymbol{b}_j - \boldsymbol{R}\boldsymbol{a}_i - \boldsymbol{t}\|^2. \tag{5.11}$$

As no closed form solution is known for this problem, both robust algorithms apply the standard approximation procedure, which consists of sampling the parameter space by repeatedly solving the original least-squares problem for a minimum number of randomly selected samples of the input data and choosing the solution that yields the best median error for all samples.

Although both the MICP algorithm and the RICP algorithm share the same theoretical background, their practical approaches are very dissimilar. The main loop of the MICP algorithm selects a random sample of the data point set, which is aligned to the model point set with the help of a standard ICP algorithm. The estimated motion parameters are retained when the median error for all data points has been reduced, and discarded otherwise. Thus, the MICP algorithm is designed as an LMedS technique wrapped around the ICP algorithm. In contrast to this, the RICP algorithm uses the LMedS technique for the motion estimation inside the main loop of the ICP algorithm. To this end, motion parameters are estimated for several random samples consisting of three point pairs and rated with the help of the median error for all point pairs. The best motion parameters are refined according to the approach illustrated in Figure 4.14. In short, a maximum threshold for the residuals is determined by multiplying the robustly estimated standard deviation of the residuals by a user-defined factor. Then, point pairs with residuals beyond the threshold are removed and the motion parameters are re-estimated with the standard least-squares approach of (5.6) for the remaining point pairs. Both the MICP algorithm and the RICP algorithm contain two nested loops, one of which is the main loop of the standard ICP algorithm. Consequently, it is not surprising that both robust algorithms are considerably less efficient than the standard ICP algorithm for many configurations of the input data [Zin03a].

In the remainder of this section, we describe the three extensions for robust correspondence estimation that are implemented in our variant of the ICP algorithm. The basic approach of all three extensions is to identify and remove outliers among the corresponding point pairs at the end of the correspondence estimation step. The resulting structure of our variant of the ICP algorithm is illustrated in Figure 5.3. As no additional information is provided by the input data in our work, the intra-pair distance of a point pair, which is equivalent to the residual of this point pair, is the

only available criterion for identifying outliers. Furthermore, inliers are expected to have a smaller residual than outliers. Consequently, the basic principle of the implemented extensions is to determine a threshold for the residuals that separates inliers and outliers.

Our first extension is based on the integrated outlier rejection of the least median of squares technique described in Subsection 4.3.4. According to (4.85), the extension robustly estimates the standard deviation of the residuals as

$$\sigma = 1.4826 \sqrt{\underset{(i,j)\in\mathcal{C}}{\text{med}} \; \|\boldsymbol{b}_j - \boldsymbol{R}\boldsymbol{a}_i - \boldsymbol{t}\|^2} \,. \tag{5.12}$$

Point pairs with a residual of more than $\theta_{\text{out\_lms}}\,\sigma$ are treated as outliers and deleted from the set of corresponding point pairs $\mathcal{C}$. As proposed in [Zha97] and [Ste99], we use the factor $\theta_{\text{out\_lms}} = 2.5$.

Alternatively, we can derive this extension from the RICP algorithm by omitting its random sampling step. In the standard approach of the LMedS technique, the random sampling step is required to obtain a solution that is not perturbed by outliers. In the ICP algorithm, however, an approximate solution for the motion parameters is available from the preceding iteration. In addition to that, the initial alignment of the two point sets can be seen as an approximate solution for the first iteration. Consequently, it is possible to perform an outlier removal without the random sampling step by relying on the motion parameters robustly estimated in the preceding iteration. As a result, our first extension exhibits a much better computational efficiency than the RICP algorithm, while its robustness to outliers is similar [Zin03a].

The breakdown point of the LMedS technique is 0.5. Thus, in theory, our first extension can cope with configurations where up to 50% of the computed point pairs are outliers. However, in practice, the fraction of outliers also influences the basin of convergence of the estimation. Consequently, the higher the fraction of outliers, the more accurate the initial alignment of the two input point sets has to be for a successful registration. Compared to the standard ICP algorithm, our LMedS extension shows a much improved robustness to outliers [Zin03a], but also two minor drawbacks. Firstly, the guaranteed local convergence of the ICP algorithm can no longer be proven when the extension is integrated, which requires the adaptation of the stopping criteria described in Subsection 5.2.1. Secondly, the additional operations of the LMedS extension slightly reduce the computational efficiency of the ICP algorithm. We evaluate this aspect of all implemented extensions in Section 6.3.

In the same way that our first extension is derived from the LMedS technique, the second extension for robust correspondence estimation is derived from the least trimmed squares (LTS) technique. In principle, the LTS technique minimizes the smallest possible sum of a specified fraction θ_{lts} of the squared residuals. A comprehensive discussion of this technique, also regarding its application to the ICP algorithm, is given in [Che05]. The objective function of the LTS extension is given

by

$$\epsilon_{\text{lts}}(\boldsymbol{R}, \boldsymbol{t}, \theta_{\text{lts}}) = \sum_{(i,j) \in \mathcal{D}(\theta_{\text{lts}})} \|\boldsymbol{b}_j - \boldsymbol{R}\boldsymbol{a}_i - \boldsymbol{t}\|^2, \tag{5.13}$$

where $\mathcal{D}(\theta_{\text{lts}})$ is defined as the subset of $\mathcal{C}$ with $\lfloor \theta_{\text{lts}} \, |\mathcal{C}| \rfloor$ elements that yields the smallest value for $\epsilon_{\text{lts}}(\boldsymbol{R}, \boldsymbol{t}, \theta_{\text{lts}})$. In practice, the LTS extension sorts the point pairs according to their intra-pair distance and retains the first $\lfloor \theta_{\text{lts}} \, |\mathcal{C}| \rfloor$ point pairs. In other words, the threshold for the residuals is set to the residual of the point pair at position $\lfloor \theta_{\text{lts}} \, |\mathcal{C}| \rfloor$ in the sorted sequence of all point pairs of $\mathcal{C}$. This approach has already been proposed, albeit without the theoretical background, in [Pul99].

In contrast to the LMedS extension, the LTS extension preserves the guaranteed convergence of the ICP algorithm, because the number of inliers remains constant in all iterations of the algorithm. A more elaborate proof of the local convergence of the LTS extension is given in [Che05]. However, the necessity to directly specify the fraction of inliers θ_{lts} before the registration is a severe drawback of this approach, because the actual fraction of inliers is an unknown quantity in many applications. When the specified value of θ_{lts} is higher than the actual value, the remaining outliers will impair the accuracy of the estimation. On the other hand, specifying a lower fraction of inliers reduces the basin of convergence of the algorithm. As a consequence, a practical implementation of the LTS extension for arbitrary input point sets requires an automatic approach for roughly estimating the fraction of inliers θ_{lts}.

Our implementation of the LTS extension is based on the approach proposed in [Che05]. The basic concept of this approach is to balance the minimization of the registration error and the maximization of the fraction of inliers θ_{lts}. This goal is achieved with the objective function

$$\epsilon_{\text{ots}}(\theta_{\text{lts}}) = \left(\frac{1}{\theta_{\text{lts}}}\right)^{\theta_{\text{elts}}} \hat{\epsilon}_{\text{lts}}(\theta_{\text{lts}}) \, , \tag{5.14}$$

where $\hat{\epsilon}_{\text{lts}}(\theta_{\text{lts}})$ is the final value of the objective function in (5.13) for a specified fraction of inliers θ_{lts} after a complete registration with the ICP algorithm. The default value of θ_{elts} is 4 in [Che05]. Increasing the value of this parameter leads to a higher fraction of included point pairs, and vice versa.

The objective function $\epsilon_{\text{ots}}(\theta_{\text{lts}})$ is minimized with a golden section search in a specified search interval for the fraction of inliers θ_{lts}. Our implementation uses the search interval $[0.2, 1.0]$. In every iteration, the golden section search analyzes the objective function at two positions v_2 and v_3 in the current interval $[v_1, v_4]$, where

$$\begin{aligned} v_2 &= v_1 + 0.5(3 - \sqrt{5})(v_4 - v_1) \\ v_3 &= v_4 - 0.5(3 - \sqrt{5})(v_4 - v_1) \, . \end{aligned} \tag{5.15}$$

The search interval is then reduced to

$$\begin{aligned} &\epsilon_{\text{ots}}(v_2) < \epsilon_{\text{ots}}(v_3) \Rightarrow [v_1, v_3] \\ &\epsilon_{\text{ots}}(v_2) \geq \epsilon_{\text{ots}}(v_3) \Rightarrow [v_2, v_4] \, . \end{aligned} \tag{5.16}$$

As the size of the search interval is decreased by approximately 38% in every iteration, it is smaller than 1% of its original size after ten iterations of the golden section search. This accuracy is sufficient for the fraction of inliers θ_{lts} in practice. Due to the special placement of the positions v_2 and v_3, it is possible to reuse one function value in the next iteration. Nevertheless, the LTS extension with golden section search is far less efficient than the LMedS extension, because every evaluation of the objective function $\epsilon_{\text{ots}}(\theta_{\text{lts}})$ entails a registration of the two input point sets with the ICP algorithm.

The third and final extension for robust correspondence estimation in our work was proposed in [Phi07]. Following the designation of the algorithm in the original work, we denominate this approach as the least fractional squares (LFS) extension. Its basic idea is to optimize an objective function that is equivalent to (5.14) inside the main loop of the ICP algorithm. The objective function of the LFS extension is given by

$$\epsilon_{\text{lfs}}(\boldsymbol{R}, \boldsymbol{t}, \theta_{\text{lts}}) = \left(\frac{1}{\theta_{\text{lts}}}\right)^{\theta_{\text{elfs}}} \epsilon_{\text{lts}}(\boldsymbol{R}, \boldsymbol{t}, \theta_{\text{lts}}) \; . \tag{5.17}$$

We have unified the notation in (5.14) and (5.17) with respect to the exponents θ_{elts} and θ_{elfs}. Consequently, the respective parameter values given in [Che05] and [Phi07] have to be adjusted to our notation. Interestingly, the theoretical analysis in [Phi07] suggests that optimal performance is achieved for $\theta_{\text{elfs}} \approx 2.9$, whereas the experimental evaluations in the same work show better performance for $\theta_{\text{elfs}} = 7$. We present and analyze the results of our evaluations regarding the parameters θ_{elts} and θ_{elfs} in Subsection 6.3.3.

Our implementation of the LFS extension computes the value of $\epsilon_{\text{lfs}}(\boldsymbol{R}, \boldsymbol{t}, \theta_{\text{lts}})$ for every possible fraction of inliers θ_{lts} at the end of the correspondence estimation step. After the sorting of the residuals, the extension only requires a constant number of operations for every one of the $|\mathcal{C}|$ possible fractions, as long as the current value of $\epsilon_{\text{lts}}(\boldsymbol{R}, \boldsymbol{t}, \theta_{\text{lts}})$ is reused in the computation for the next fraction. The fraction of inliers θ_{lts} that results in the lowest value of $\epsilon_{\text{lfs}}(\boldsymbol{R}, \boldsymbol{t}, \theta_{\text{lts}})$ is used to determine the point pairs for the subsequent motion estimation. Although the estimated fraction of inliers θ_{lts} possibly changes from one iteration to the next, the LFS extension of the ICP algorithm retains the guaranteed convergence of the original ICP algorithm [Phi07].

5.4 Integrated Scale Estimation

In this section, we present an extension of the ICP algorithm that allows the registration of two point sets with an unknown relative scale. In [Bur05], the relative scale of the two input point sets is determined with a preprocessing algorithm, while the subsequent ICP algorithm is limited to refining the rotation and the translation. Due to the integration of the scale estimation into the ICP algorithm, our extension allows a more accurate registration in this scenario. Recently, Du et al. have

proposed an extension of the ICP algorithm that estimates independent scale factors for every axis of the coordinate system of the data point set [Du07]. However, the experimental evaluation in their work suggests that the increased number of parameters strongly decreases the basin of convergence of the estimation for three-dimensional input data.

Our extension for integrated scale estimation replaces the objective function of the motion estimation defined in (5.5) with

$$\epsilon_{\text{ise}}(\boldsymbol{R}, \boldsymbol{t}, s) = \sum_{(i,j) \in \mathcal{C}} \|\boldsymbol{b}_j - s\boldsymbol{R}\boldsymbol{a}_i - \boldsymbol{t}\|^2. \tag{5.18}$$

The optimization of the respective least-squares problem

$$(\hat{\boldsymbol{R}}, \hat{\boldsymbol{t}}, \hat{s}) \;=\; \underset{\boldsymbol{R},\boldsymbol{t},s}{\operatorname{argmin}} \sum_{(i,j) \in \mathcal{C}} \|\boldsymbol{b}_j - s\boldsymbol{R}\boldsymbol{a}_i - \boldsymbol{t}\|^2 \tag{5.19}$$

is based on the observation that the estimation of the optimum rotation matrix $\hat{\boldsymbol{R}}$ presented in (5.7) to (5.10) is not affected by the scale factor s at all. Applying a scale factor s to the data points only affects the singular values in the matrix $\boldsymbol{D}$, whereas the matrices $\boldsymbol{U}$ and $\boldsymbol{V}^{\mathrm{T}}$ of the singular value decomposition in (5.9) remain unaltered. Consequently, the rotation matrix can still be computed as defined in (5.10). After the estimation of the optimal rotation matrix $\hat{\boldsymbol{R}}$, the scale factor s is the only remaining unknown in the minimization problem

$$\hat{s} = \underset{s}{\operatorname{argmin}} \sum_{(i,j) \in \mathcal{C}} \|(\boldsymbol{b}_j - \bar{\boldsymbol{b}}) - s\hat{\boldsymbol{R}}(\boldsymbol{a}_i - \bar{\boldsymbol{a}})\|^2 . \tag{5.20}$$

After setting the derivative of this sum to zero, the scale factor $\hat{s}$ is given by

$$\hat{s} = \left(\sum_{(i,j) \in \mathcal{C}} \tilde{\boldsymbol{b}}_j^T \tilde{\boldsymbol{a}}_i \right) \left(\sum_{(i,j) \in \mathcal{C}} \tilde{\boldsymbol{a}}_i^T \tilde{\boldsymbol{a}}_i \right)^{-1}, \tag{5.21}$$

where the vectors $\tilde{\boldsymbol{a}}_i$ and $\tilde{\boldsymbol{b}}_j$ are defined as

$$\tilde{\boldsymbol{a}}_i = \hat{\boldsymbol{R}} \left(\boldsymbol{a}_i - \bar{\boldsymbol{a}} \right) , \quad \tilde{\boldsymbol{b}}_j = \left(\boldsymbol{b}_j - \bar{\boldsymbol{b}} \right) . \tag{5.22}$$

Finally, the translation vector $\hat{\boldsymbol{t}}$ can be computed according to

$$\hat{\boldsymbol{t}} = \bar{\boldsymbol{b}} - \hat{s}\hat{\boldsymbol{R}}\bar{\boldsymbol{a}} . \tag{5.23}$$

Our extension for integrated scale estimation replaces the motion estimation step of the original ICP algorithm. In other parts of the algorithm, only the application of the motion parameters to the data point set has to be extended with the added scale factor. As a consequence, this extension can easily be applied to a wide range

initialize nearest neighbor search with model point set $\mathcal{B}$		
FOR $k = 1$ TO δ_{iter}		
	compute $\mathcal{A}_k$ by applying the current motion to the data point set $\mathcal{A}$	
	compute the set of corresponding point pairs $\mathcal{C}_k$ for $\mathcal{A}_k, \mathcal{B}$ (cf. Subs. 5.2.1)	
	compute $\mathcal{C}_k^*$ by optionally rejecting point pairs (cf. Sect. 5.3)	
	IF	scale estimation is enabled
	THEN	compute new motion $(\boldsymbol{R}_k, \boldsymbol{t}_k, s_k)$ for $\mathcal{C}_k^*, \mathcal{A}, \mathcal{B}$ (cf. Sect. 5.4)
	ELSE	compute new motion $(\boldsymbol{R}_k, \boldsymbol{t}_k)$ for $\mathcal{C}_k^*, \mathcal{A}, \mathcal{B}$ (cf. Subs. 5.2.1)
	IF	$\|\boldsymbol{R}_k - \boldsymbol{R}_{k-1}\| < \theta_{\text{stop}}$ AND $\|\boldsymbol{t}_k - \boldsymbol{t}_{k-1}\| < \theta_{\text{stop}}$ AND $\|s_k - s_{k-1}\| < \theta_{\text{stop}}$
	THEN	EXIT FOR

Figure 5.3: The structure of our variant of the ICP algorithm

of variants of the ICP algorithm. In particular, the extensions for robust correspondence estimation described in Section 5.3 are perfectly compatible with the extension for integrated scale estimation. The scale estimation results in a minor increase of the computation time per iteration. We investigate its effects on the basin of convergence of the ICP algorithm in Section 6.3. In addition to that, we also evaluate the basin of convergence of the ICP algorithm with integrated scale estimation with respect to the initial scale factor.

5.5 Technical Specifications

We illustrate the structure of our variant of the ICP algorithm in Figure 5.3. The respective approach for rejecting outliers from the set of corresponding point pairs for the LMedS extension, the LTS extension, and the LFS extension is described in Section 5.3. In contrast to the other two extensions, the LTS extension requires a golden section search to automatically determine the optimum value for the fraction of inliers θ_{lts}. As the additional search, which repeatedly calls the ICP algorithm with different values of θ_{lts}, is unique to the LTS extension, it has been omitted from the illustration of the structure of our variant of the ICP algorithm in Figure 5.3. In our implementation, only one extension for outlier rejection can be activated at any one time. Of course, it is also possible to completely deactivate the outlier rejection.

Table 5.1 presents all user-specified parameters of our variant of the ICP algorithm. The parameter δ_{iter} specifies the maximum number of iterations of the ICP algorithm. We use it exclusively to stop a possible infinite loop caused by divergent behavior of the ICP algorithm when the LMedS extension is enabled. As a

name	type	description	default
δ_{iter}	stopping criterion	maximum number of iterations	256
θ_{stop}	stopping criterion	minimum motion difference	10^{-9}
δ_{type}	outlier rejection	outlier rejection type	$*$
δ_{ilts}	outlier rejection	number of search iterations for LTS	8
θ_{elts}	outlier rejection	exponent for LTS	4.0
θ_{elfs}	outlier rejection	exponent for LFS	5.0
δ_{scale}	scale estimation	enable scale estimation	false

Table 5.1: This table enumerates the user-specified parameters of our variant of the ICP algorithm. Its fourth column contains the default values of the parameters. Default values shown as $*$ are discussed in the text below.

consequence, its default value of 256 iterations has been chosen to be large enough to allow the convergence of the ICP algorithm for most configurations of the input data. When the maximum number of iterations is reached although the ICP algorithm is still converging, e. g., for very large point sets with more than 100000 points, the value of δ_{iter} has to be increased. Commonly, our variant of the ICP algorithm is terminated by the stopping criterion controlling the minimum motion difference θ_{stop}. Its value can be adjusted to better reflect the accuracy required by the application.

The parameter δ_{type} allows the selection of one of the extensions for outlier rejection, which are presented in Section 5.3. In addition to the omission of any outlier rejection, the possible choices comprise the LMedS extension, the LTS extension with automatic search, and the LFS extension. The respective parameters δ_{ilts}, θ_{elts}, and θ_{elfs} are discussed in Section 5.3. Finally, the parameter δ_{scale} allows the activation of the integrated scale estimation, which is described in Section 5.4.

5.6 Summary

This chapter covers the problem of point set registration. In its first section, we derive the problem definition by generalizing the ideal motion problem. In addition to that, we illustrate the required assumptions for a successful registration of two point sets and discuss the performance criteria for evaluating point set registration algorithms. We also categorize the existing work on point set registration with respect to several different aspects. In accordance with our application requirements, we focus on a locally convergent algorithm that allows the registration of two 3-D point sets, the ICP algorithm. As a consequence, the second section of this chapter comprises an explanation of the ICP algorithm and its most important extensions.

The third section contains the description of three extensions for more robust correspondence estimation. Our detailed analysis correlates the three approaches and exposes interesting similarities between them. However, our main contribution in

this problem area is the thorough experimental evaluation of the implemented approaches in Subsection 6.3.3. In the fourth section, we present a new approach for adding the estimation of an unknown scale factor to the ICP algorithm. The systematic experimental evaluation of this approach can be found in Subsection 6.3.4.

Chapter 6

Experimental Evaluation

This chapter contains the experimental evaluation of the algorithms presented in the three preceding chapters. Its main task is to evaluate the performance of the proposed algorithms with respect to the criteria defined in Subsection 3.1.1, Subsection 4.1.1, and Subsection 5.1.1, respectively. In addition to that, we use the experimental evaluation of the algorithms to analyze the impact of important user-specified parameters on the performance of these algorithms.

For a meaningful interpretation of the computation times presented in this chapter, it is necessary to consider the configuration of the test environment. Unless explicitly noted otherwise, the experiments were performed with the following configuration. The test images were captured with the digital camera Sony DFW-VL 500 at a resolution of 640×480 pixels. For the implementation of the algorithms, we used the C++ programming language. The algorithms were compiled with g++ 4.3.3 in 64 bit mode. We evaluated the resulting programs on a computer equipped with one AMD Athlon X2 5000+ dual-core cpu and 4096 MB of main memory.

The first three sections of this chapter correspond to the three main work areas of this thesis. For every experiment described in these sections, we provide a detailed description of the experimental setup. Furthermore, we present and thoroughly analyze the results of the performed experiments. In the final section of this chapter, we demonstrate the joint operation of our proposed algorithms using three test sequences with known ground truth data.

6.1 Feature Point Tracking

This section is concerned with the experimental evaluation of the feature point tracking algorithms presented in Chapter 3. To this end, we analyze the properties of the individual algorithms by simulating the displacement of the tracked feature points on images of real-world test scenes. This approach strikes a fine balance between the accuracy of the quantitative evaluation and the transferability of the results to real-world applications. In addition to that, we evaluate the performance of our feature point tracking system in Subsection 6.1.3. Furthermore, our tracking system is evaluated in conjunction with our structure and motion estimation system in Section 6.4.

algorithm	type	motion	intensity	reference
gd tra	gradient descent	t	-	Subs. 3.3.3, (3.21)
gd tra iie	gradient descent	t	b + c	Subs. 3.3.4, (3.34)
gd aff	gradient descent	a + t	-	Subs. 3.3.3, (3.21)
gd aff die	gradient descent	a + t	b + c	Subs. 3.3.4, (3.31)
gd aff sie	gradient descent	a + t	b + c	Subs. 3.3.4, (3.33)
gd aff iie	gradient descent	a + t	b + c	Subs. 3.3.4, (3.34)
bm nssd	block matching	t	-	Subs. 3.4.2, (3.40)
bm corr	block matching	t	c	Subs. 3.4.2, (3.41)
bm coef	block matching	t	b + c	Subs. 3.4.2, (3.42)

Table 6.1: This table lists all motion estimation algorithms that are available in our feature point tracking system. The abbreviations in the first column are used throughout this section to reference the respective algorithms. Some algorithms estimate only the translation (t) of a feature point, whereas other algorithms also estimate its affine distortion (a). The robustness of the algorithms to intensity changes in the feature windows is specified for changes in brightness (b) and contrast (c).

6.1.1 Experimental Setup

Table 6.1 provides an overview of all motion estimation algorithms described in Chapter 3. While our block matching algorithms only estimate the translation of a feature point, we have implemented gradient descent algorithms that allow the additional estimation of the affine distortion of the feature window. There are variants of both types of algorithms that are robust to changes in brightness and contrast in the feature windows. As described in Subsection 3.3.4, the gradient descent algorithms achieve this robustness by explicitly estimating the parameters of an affine linear intensity equalization model. In contrast to this, the respective block matching algorithm resorts to a difference measure that is invariant to changes in both brightness and contrast. The default values of the user-defined parameters of all evaluated motion estimation algorithms are listed in Table 3.1 in Subsection 3.3.7.

For the evaluation of the individual motion estimation algorithms, we captured images of the four static scenes depicted in Figure 6.1. For each scene, the captured images comprise one base image, one image that is identical to the base image up to image noise, and two images that are slightly overexposed and slightly underexposed, respectively. The resulting images are illustrated for one of the four scenes in Figure 6.2. An important step in our evaluation of the individual algorithms is the selection of suitable feature points in the four base images. To this end, we select the 2000 features points with the highest values of the computed quality measure for every considered combination of the feature selection parameters. As default parameter values, we use the values specified in Table 3.1 with the Tomasi-Kanade detector, except for the minimum distance to the image border, which is set to 32. Furthermore, the user-specified parameters $\delta_{\text{ftr\_num}}$ and $\theta_{\text{ftr\_qual}}$ are not applicable

Figure 6.1: Four static test scenes are used for the evaluation of the individual motion estimation algorithms.

Figure 6.2: In addition to the images captured with standard exposure settings (middle image), every static test scene was slightly overexposed (left image) and underexposed (right image). The resulting images are used to evaluate the robustness of the individual motion estimation algorithms to intensity changes in the feature windows.

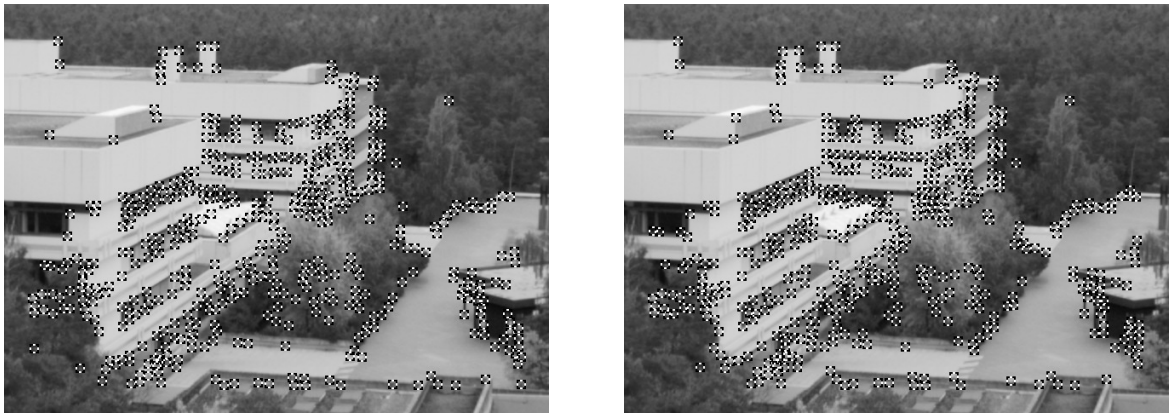


Figure 6.3: The feature points in both images were selected using the parameters described in the text below. The left image shows 461 features points obtained with the Tomasi-Kanade detector, whereas the right image shows 472 feature points obtained with the Harris detector. In both cases, the total number of feature points in all four base images is 2000.

to the described feature selection approach. Figure 6.3 depicts the feature points selected in one base image for two different feature quality measures.

As illustrated in Figure 3.3, the individual motion estimation algorithms require both a reference image and a current image as input data. In our experimental setup, the captured base images are always used as reference images. For standard experiments, we use the images with identical exposure settings as current images. For evaluating the robustness of the algorithms to intensity changes, we use both the overexposed and the underexposed images as current images. The individual motion estimation algorithms also require a starting position for their estimation of the feature position in the current image. As the four test scenes are static, the feature points lie at the same position in all input images of a test scene. Thus, we simulate the motion of the feature points by displacing the starting position in the current image. To this end, we generate starting positions by specifying the absolute value of the displacement and randomly choosing the direction of the displacement from a uniform distribution. In the subsequent subsection, all motion estimation algorithms are evaluated with a feature window size of 11×11 pixels to facilitate the comparison of the individual algorithms. Unless explicitly noted, all other parameters are set to their default values listed in Table 3.1.

Our proposed experimental setup has the following properties. As our test images were captured with a real camera, they contain realistic amounts of noise. In addition to that, the fixed feature positions result in highly accurate ground truth data for evaluating the accuracy and the basin of convergence of the individual motion estimation algorithms. In contrast to this, image sequences with moving objects make it difficult to obtain accurate ground truth data due to problems like sampling artifacts and distortions caused by the camera lens. As a downside, our experimental setup does not allow the evaluation of the motion estimation algo-

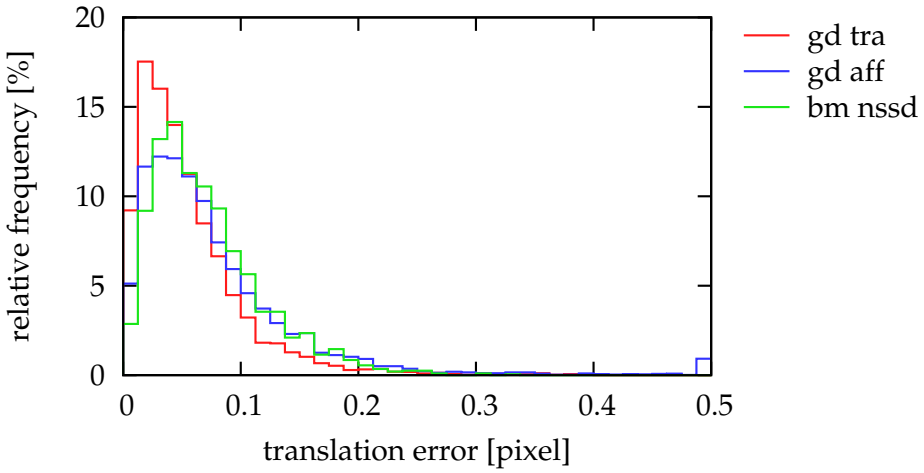


Figure 6.4: The accuracy of three motion estimation algorithms for images with identical exposure settings is illustrated with a histogram of the translation error. The rightmost value of the histogram also represents the translation errors that are outside the range of the graph. One million estimations were performed for every algorithm.

rithms with respect to phenomena that only occur when either the camera or the scene objects move, e. g., the perspective distortion of the feature windows. Consequently, we investigate these phenomena during the evaluation of the complete feature point tracking system with more realistic input data in Subsection 6.1.3.

6.1.2 Evaluation of Individual Algorithms

In our first experiment, we evaluate the accuracy of three motion estimation algorithms using the test images with identical exposure settings. In order to ensure a successful convergence of the algorithms, we use initial displacements of less than 2 pixels. Each of the 2000 feature points in our four test scenes contributes to 50 estimations for every distance between 0.0 pixels and 1.8 pixels that is a multiple of 0.2 pixels. The results of this experiment are illustrated in Figure 6.4. The accuracy of the three algorithms is very similar, but the gradient descent translation estimation algorithm is slightly more accurate than the other two algorithms. However, it is important to note that our experimental setup prevents sampling artifacts, which potentially decrease the accuracy of the algorithms in real image sequences.

Our second experiment evaluates the accuracy of three suitable motion estimation algorithms using the test images with differing exposure settings. As the evaluation considers both the overexposed and the underexposed images of the four test scenes, the total number of estimations per algorithm is doubled. The results of our second experiment are depicted in Figure 6.5. Again, the results of the three algorithms are very similar. In this experiment, the block matching algorithm is slightly more accurate than the two gradient descent algorithms. Comparing the

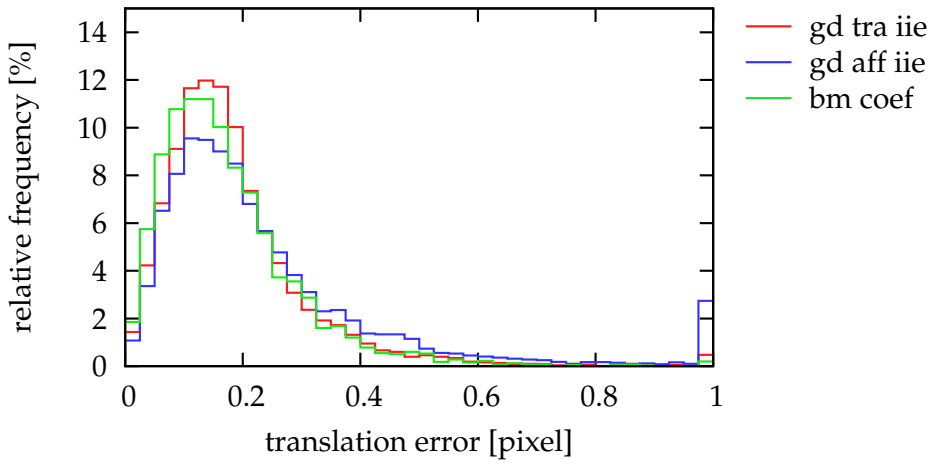


Figure 6.5: The accuracy of three motion estimation algorithms for the test images with differing exposure settings is illustrated with a histogram of the translation error. The rightmost value of the histogram also represents the translation errors that are outside the range of the graph. Two million estimations were performed for every algorithm.

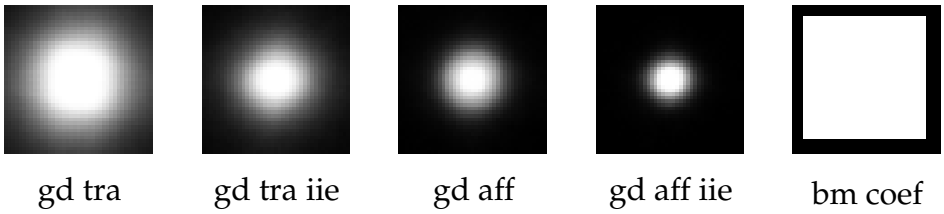


Figure 6.6: The images in this figure illustrate the shape of the basin of convergence of five motion estimation algorithms for the test images with identical exposure settings. From left to right, the gradient descent algorithms estimate two, four, six, and eight parameters. Further details on this figure are given in the text below.

histograms in Figure 6.4 and Figure 6.5, we have to conclude that the intensity changes in the second experiment impair the accuracy of the motion estimation, even though the tested algorithms are designed to be robust to intensity changes.

In the following experiments, we evaluate the basin of convergence of the individual motion estimation algorithms. To this end, we deem the tracking of a feature point successful when the translation error, i.e, the distance between the true position and the estimated position of a feature point, lies below a certain threshold. Taking the accuracy of the algorithms presented in Figure 6.4 and Figure 6.5 into account, we set this threshold to 0.4 pixels for the images with identical exposure settings and to 0.8 pixels for the images with differing exposure settings.

Figure 6.6 illustrates the shape of the basin of convergence of five motion estimation algorithms. For this experiment, 2000 feature points were tracked with different initial displacements using the test images with identical exposure settings. The pixel position in the images represents the initial displacement of the feature

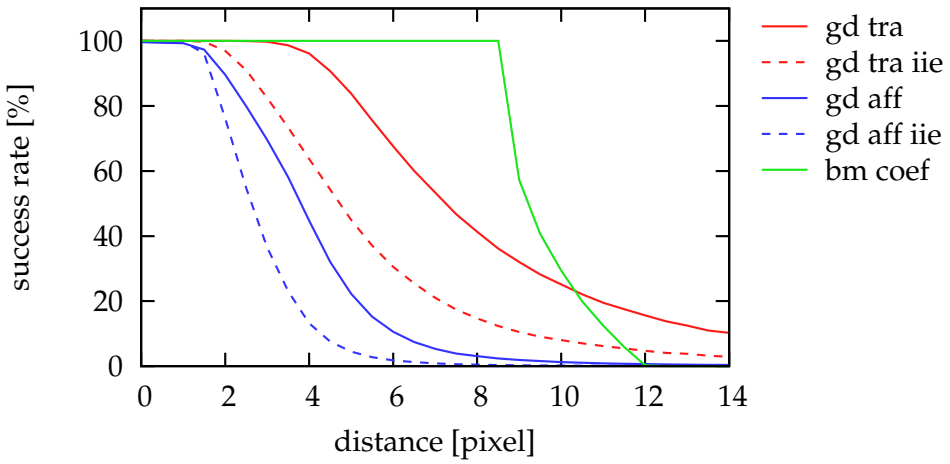


Figure 6.7: The basin of convergence of five motion estimation algorithms for the test images with identical exposure settings is visualized by plotting the success rate against the distance of the initial displacement.

points. In the center of the images, the initial displacement is zero. It increases to a maximum of ten pixels in both horizontal and vertical direction at the borders of the images. The gray value of the pixels is directly proportional to the success rate of the motion estimation. Figure 6.6 shows that the shape of the basin of convergence is approximately circular for the gradient descent algorithms and closely resembles the rectangular search window for the block matching algorithm. In addition to that, it visualizes that the basin of convergence of the gradient descent algorithms decreases when the number of estimated parameters increases.

As the shape of the basin of convergence is very regular for our motion estimation algorithms, we analyze the basin of convergence with respect to the distance of the initial displacements in the remaining experiments of this subsection. For every feature point and every distance of the initial displacement, we generate 50 starting positions by randomly choosing the direction of the initial displacement. Consequently, every observation in the following experiments is based on 100000 estimations for images with identical exposure settings and on 200000 estimations for images with differing exposure settings.

Like the images in Figure 6.6, the graph in Figure 6.7 visualizes the basin of convergence of five motion estimation algorithms for the test images with identical exposure settings. When the distance of the initial displacement is larger than the size of the search range $\delta_{\text{bm\_rng}}$, the random direction of the initial displacement influences whether the positions evaluated by the block matching algorithm include the correct feature position. Consequently, the progressive decline of the success rate of "bm coef" for simulated translations between eight and twelve pixels is a direct result of the geometric shape of its search range. For the application of the motion estimation algorithms in our feature point tracking system, only success rates of

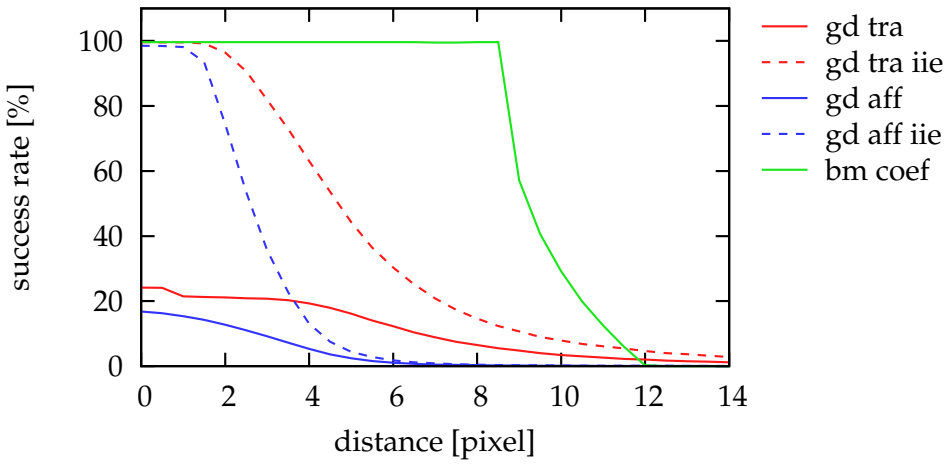


Figure 6.8: The basin of convergence of five motion estimation algorithms for the test images with differing exposure settings is illustrated by plotting the success rate against the distance of the initial displacement.

more than 95% allow the estimation of sufficiently long feature trails. Therefore, with the current set of user-specified parameters, the gradient descent algorithm "gd tra" with the largest basin of convergence is restricted to translations of less than four pixels.

Figure 6.8 presents the basin of convergence of the same motion estimation algorithms as in the preceding experiment for the test images with differing exposure settings. Evidently, the gradient descent algorithms without intensity equalization are not able to cope with the intensity changes in the test images. Furthermore, the gradient descent algorithm "gd aff iie" fails to achieve a success rate of 100% even for very small distances. Figure 6.15 illustrates this problem more clearly and shows that it can be mitigated by increasing the size of the feature windows for the affine motion estimation.

Figure 6.9 provides a direct comparison of the basin of convergence of all gradient descent algorithms for affine motion estimation listed in Table 6.1. As this experiment is based on the test images with differing exposure settings, the algorithm "gd aff" without intensity equalization has a very low success rate for all distances of the initial displacement. For distances of more than two pixels, the algorithm "gd aff die", which is based on the normalization of the intensity distributions, is clearly superior to the other algorithms. However, in practice, the success rates of all four algorithms are not sufficient for reliably estimating distances of more than two pixels. As the three algorithms with intensity equalization yield quite similar results for distances of less than two pixels, the choice of the default algorithm for our feature point tracking system depends on additional performance criteria, like the computational efficiency. To this end, we analyze the computational efficiency of the algorithms for affine motion estimation in Subsection 6.1.3.

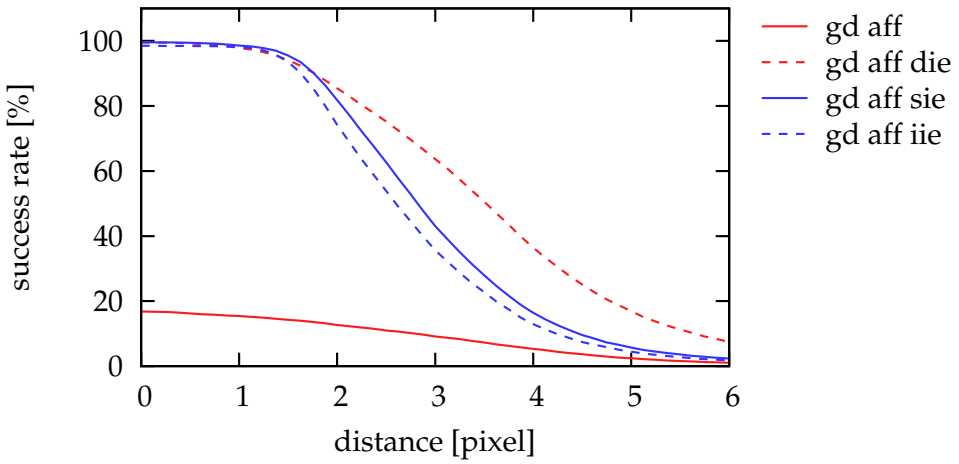


Figure 6.9: This graph visualizes the basin of convergence of the four gradient descent algorithms for affine motion estimation itemized in Table 6.1. The evaluation is based on the test images with differing exposure settings.

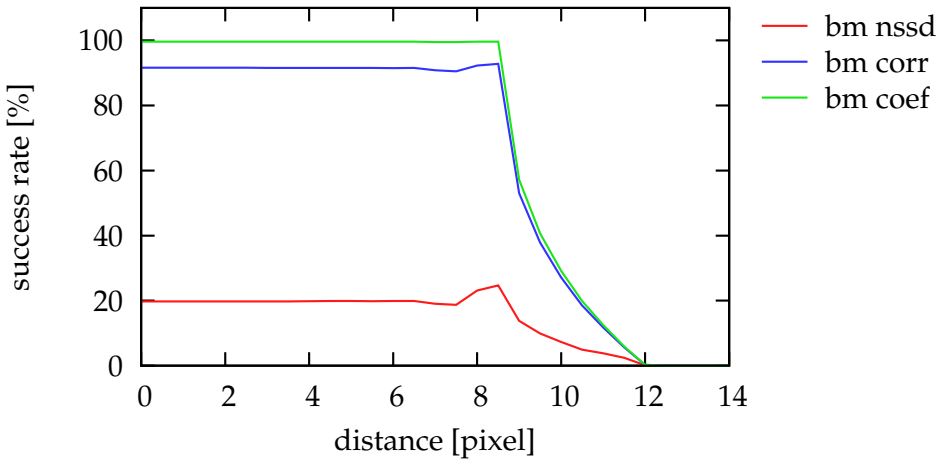


Figure 6.10: This graph illustrates the basin of convergence of the three block matching algorithms listed in Table 6.1. The evaluation is based on the test images with differing exposure settings.

The basin of convergence of the three block matching algorithms listed in Table 6.1 is compared in Figure 6.10. The test images with differing exposure settings are too difficult for the algorithms based on the normalized sum of squared differences “bm nssd” and the normalized cross correlation “bm corr”. In contrast to this, the normalized correlation coefficient “bm coef” demonstrates its superior robustness to intensity changes in this experiment. Furthermore, our experiments indicate that the computational efficiency of the three block matching algorithms is comparable. Therefore, we use the algorithm “bm coef” as our default block

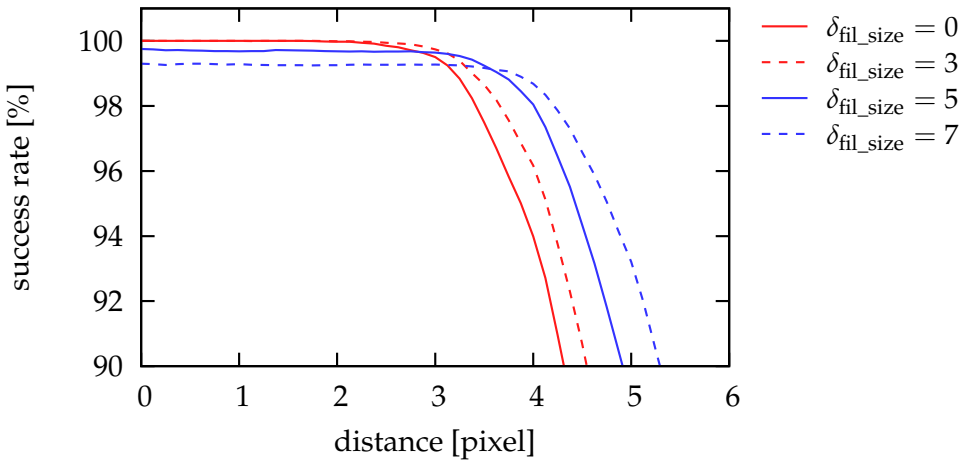


Figure 6.11: This graph shows the basin of convergence of the algorithm "gd tra" for different filter sizes $\delta_{\text{fil\_size}}$. The evaluation is based on the test images with identical exposure settings.

matching algorithm, even though the intensity changes from one image to the next are usually quite small in practice.

In the following experiments, we evaluate the influence of several important user-specified parameters on the performance of the respective motion estimation algorithms. As explained in Subsection 3.3.7, our feature point tracking system smoothes the input images with a Gaussian filter. Figure 6.11 illustrates the effects of the window size $\delta_{\text{fil\_size}}$ of this filter on the basin of convergence of the algorithm "gd tra" for the test images with identical exposure settings. For small distances, window sizes $\delta_{\text{fil\_size}}$ of more than three pixels reduce the success rate of the gradient descent algorithm, because the blurring becomes so strong that an increasing number of feature points cannot be reliably localized anymore. In contrast to this, the additional smoothing provided by larger window sizes facilitates the motion estimation for distances of more than four pixels. As we set a high value on an excellent success rate for small distances, we recommend adhering to our default value of $\delta_{\text{fil\_size}} = 3$ for most applications of our feature point tracking system.

As described in Subsection 3.3.7, the parameter $\delta_{\text{ftr\_type}}$ controls the type of the feature detector. Our feature point tracking system supports the Tomasi-Kanade feature detector as well as the Harris feature detector. The respective quality measures are detailed in Subsection 3.2.2. In addition to that, the parameter $\delta_{\text{ftr\_size}}$ allows the specification of the size of the feature windows. The basin of convergence of the algorithm "gd tra" is presented for both feature detectors and two different window sizes in Figure 6.12. Evidently, the type of the feature detector has no tangible effect on the success rates of the motion estimation algorithm. For historical reasons, we employ the Tomasi-Kanade feature detector by default in our feature point tracking system. Figure 6.12 also shows that the larger window size results in

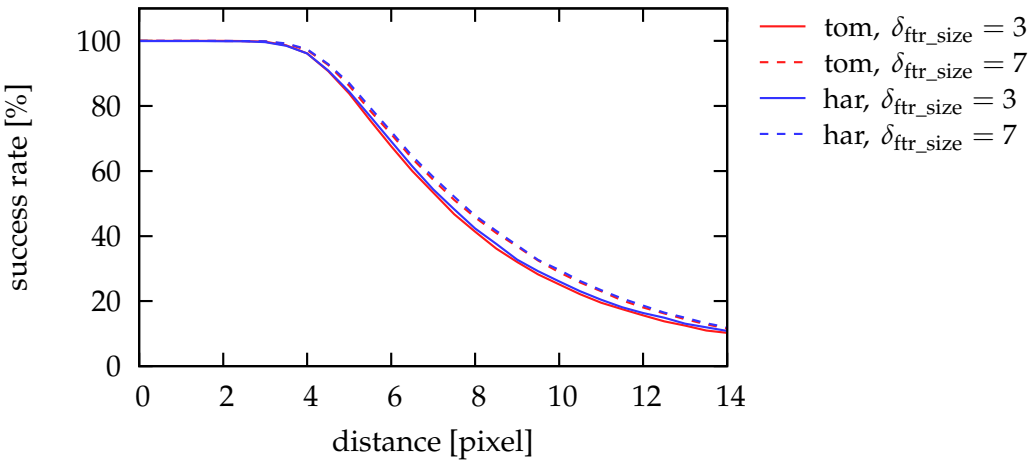


Figure 6.12: This graph presents the basin of convergence of the algorithm "gd tra" for two feature detectors $\delta_{\text{ftr\_type}}$ and window sizes $\delta_{\text{ftr\_size}}$. The feature detectors are the Tomasi-Kanade detector (tom) and the Harris detector (har), respectively. The evaluation is based on the test images with identical exposure settings.

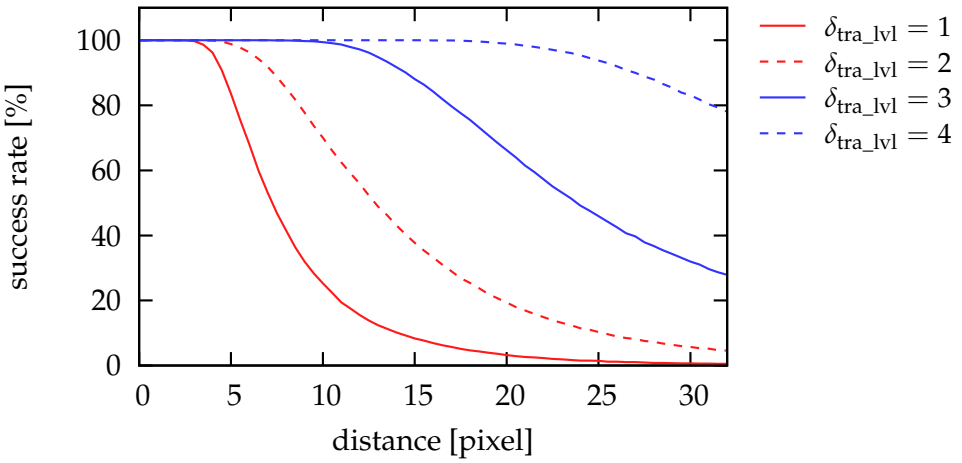


Figure 6.13: This graph illustrates the effects of the number $\delta_{\text{tra\_lvl}}$ of hierarchy levels of the Gaussian image pyramid on the basin of convergence of the algorithm "gd tra" for a feature window size of 11×11 pixels. The evaluation is based on the test images with identical exposure settings.

a slightly higher success rate for large distances. As we prefer the feature windows for feature detection to be smaller than the feature windows for motion estimation due to the issues described in Subsection 3.2.2, we use a default value of $\delta_{\text{ftr\_size}} = 3$.

As discussed in the analysis of Figure 6.7, the practical application of the gradient descent algorithms is limited to translations of less than four pixels when the feature window has a size of 11×11 pixels. Our approach for hierarchical translation

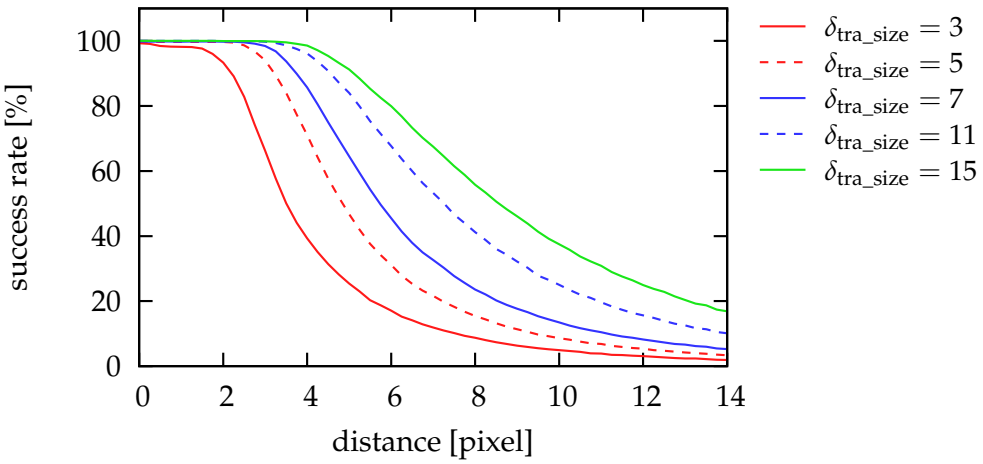


Figure 6.14: This graph visualizes the basin of convergence of the algorithm "gd tra" for five different feature window sizes $\delta_{\text{tra\_size}}$. The evaluation is based on the test images with identical exposure settings.

estimation, which is detailed in Subsection 3.3.5, uses a Gaussian image pyramid to increase the basin of convergence of the gradient descent algorithms for translation estimation. Figure 6.13 illustrates the basin of convergence of the algorithm "gd tra" for different values of the number $\delta_{\text{tra\_lvl}}$ of hierarchy levels. Every additional level of the image pyramid considerably increases the basin of convergence of the gradient descent algorithm. However, on the fourth level of the image pyramid, an input image with a resolution of 640×480 pixels is reduced to an image with a resolution of 80×60 pixels. Analogously, as the physical size of the feature window remains constant on all levels of the image pyramid, a feature window with a size of 11×11 pixels on the fourth level of the image pyramid covers an area of 88×88 pixels in the input image. In practice, feature windows of this size are prone to occlusions, depth discontinuities, and independently moving scene objects. Furthermore, the prediction of the feature positions described in Subsection 3.3.5 reduces the required basin of convergence when the motion of the feature points is not completely erratic. Consequently, the optimum value of the number $\delta_{\text{tra\_lvl}}$ of hierarchy levels depends on the characteristics of the input image sequence. In our experience, $\delta_{\text{tra\_lvl}} = 3$ is a viable setting for most applications.

Finally, we evaluate the effect of the size of the feature windows on the basin of convergence of the gradient descent algorithms. The first experiment is concerned with the gradient descent algorithm "gd tra" for translation estimation. Its results are presented in Figure 6.14. A feature window size of 3×3 pixels is clearly insufficient for a reliable translation estimation. In contrast to this, a feature window size of 5×5 pixels results in an acceptable basin of convergence of the gradient descent algorithm. Larger feature window sizes continue to improve the basin of convergence by small quantities. In the second experiment, we analyze the basin

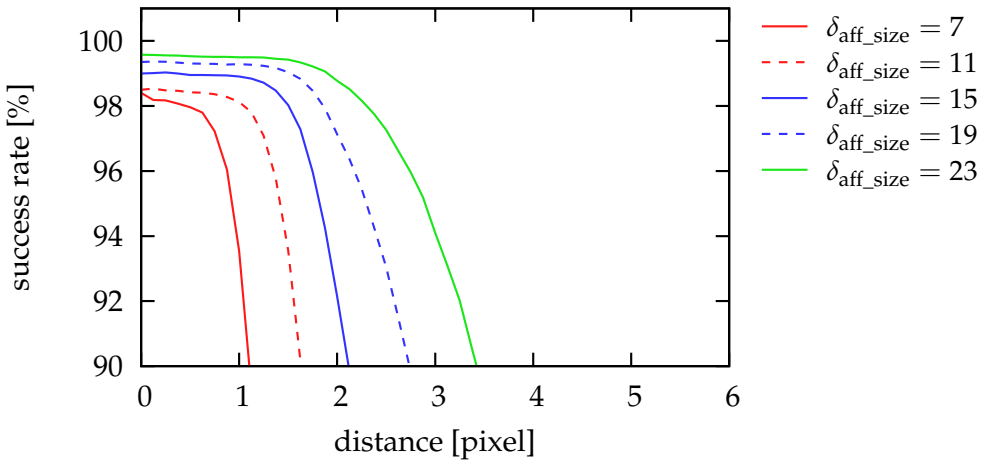


Figure 6.15: This graph shows the basin of convergence of the algorithm "gd aff iie" for five different feature window sizes $\delta_{\text{aff\_size}}$. The evaluation is based on the test images with differing exposure settings.

of convergence of the gradient descent algorithm "gd aff iie" for affine motion estimation. The corresponding results are provided in Figure 6.15. In this experiment, the success rate of the algorithm improves with every increase of the feature window size. Furthermore, it is obvious that the algorithm for affine motion estimation requires larger feature window sizes than the algorithm for translation estimation. As described in the preceding paragraph, large feature windows have disadvantages that cannot be uncovered by our experimental setup. In addition to that, the computation time of the motion estimation algorithms increases with the size of the feature windows. Consequently, we recommend the default values of $\delta_{\text{tra\_size}} = 7$ and $\delta_{\text{aff\_size}} = 15$ as a good compromise between the basin of convergence and the computational efficiency of the respective motion estimation algorithms.

6.1.3 Evaluation of the Tracking System

We complement the evaluation of the individual motion estimation algorithms in the preceding subsection with the evaluation of our feature point tracking system in this subsection. First and foremost, the use of an image sequence of a real-world scene results in more realistic conditions for the evaluation of the motion estimation algorithms, because the image sequence introduces problems like the perspective distortion of the feature windows. However, as the input sequence contains no ground truth data for the correct feature positions, its use also prevents the precise evaluation of the accuracy and the basin of convergence of the algorithms. As a consequence, we focus on other performance criteria in this subsection.

For the evaluation of our feature point tracking system, we use the test image sequence illustrated in Figure 6.16. The image sequence consists of 100 images and

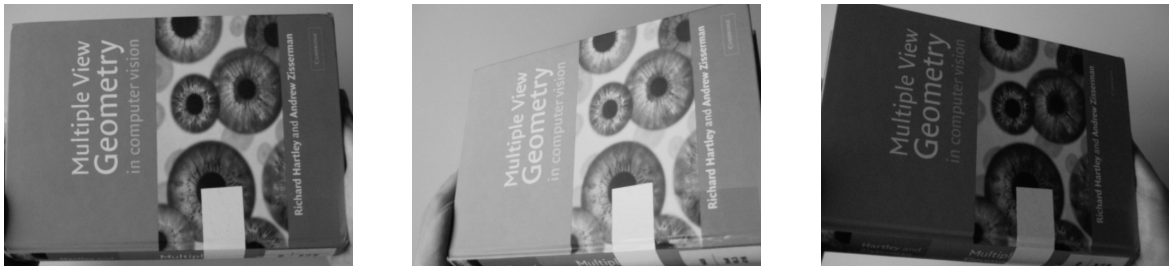


Figure 6.16: From left to right, this figure shows image 1, 50, and 100 of the test image sequence used for the evaluation of our feature point tracking system.

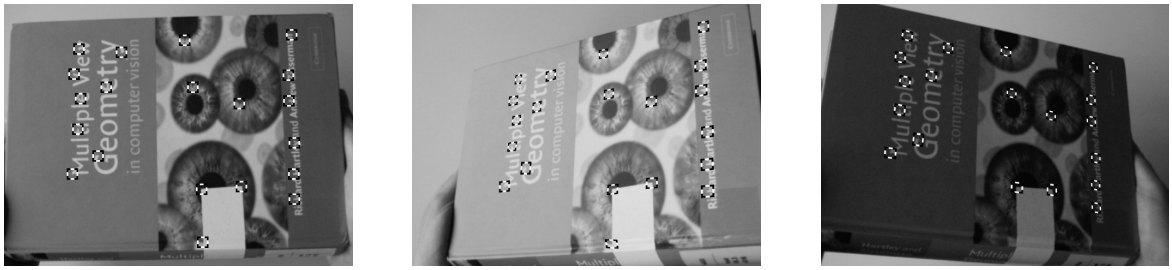


Figure 6.17: The feature windows in the three sample images represent the experimental result of test run 3 with 20 feature points (cf. Table 6.2). One feature point was rejected in image 91 of the test image sequence. A visual inspection of the remaining feature points does not reveal any significant tracking errors.

includes strong intensity changes and perspective distortions of a planar scene object. In most image sequences, a large number of feature points leave the field of view of the camera or are occluded by other scene objects. In contrast to this, nearly all potential feature points remain visible in all images of our test video sequence. Consequently, a perfect tracking system should be able to track the selected feature points throughout all images of the test sequence. In Section 6.4, we use further test image sequences with different properties to analyze the performance of our feature point tracking system.

Table 6.2 contains the results of twelve test runs of our tracking system. In general, we rely on the default values of the parameters of our feature point tracking system, which are specified in Table 3.1. However, we increased the minimum distance of the feature windows to the image border to $\delta_{\text{ftr\_bord}} = 40$, so that all selected feature points remain visible throughout the whole image sequence. Furthermore, we deactivated the selection of additional feature points after the first image by changing $\delta_{\text{ftr\_step}}$ to 0. This setting prevents additional runs of the feature selection algorithm, which possibly distort the measured computation times. Each test run was performed with 20, 100, and 400 feature points by adjusting the parameter $\delta_{\text{ftr\_num}}$ accordingly. Finally, the parameter settings of individual test runs are documented in the second column of Table 6.2.

The average length of the feature trails cannot be larger than the length of the

no.	configuration	t 20	t 100	t 400	atl	ait tra	ait aff
1	gd tra + gd aff die	4.59	8.12	21.06	99.60	9.375	2.187
2	gd tra + gd aff sie	4.48	8.09	20.97	99.52	9.377	2.191
3	gd tra + gd aff iie	4.47	7.52	18.98	99.74	9.377	2.072
4	• $\delta_{\text{tra\_min}} = 1$	4.35	7.46	18.43	99.74	7.317	2.073
5	• $\delta_{\text{tra\_min}} = 2$	4.40	7.21	17.67	99.14	5.357	2.073
6	• $\delta_{\text{tra\_size}} = 11$	4.81	9.13	24.93	99.76	8.945	2.034
7	gd tra iie + gd aff iie	4.66	8.76	24.11	99.76	9.345	2.010
8	bm coef + gd aff iie	3.25	6.70	20.07	93.90	-	2.077
9	• $\delta_{\text{bm\_chk}} = \text{true}$	3.43	7.33	21.84	99.75	-	2.068
10	• $\delta_{\text{fil\_size}} = 0$	2.73	6.70	21.42	99.29	-	2.067
11	• $\delta_{\text{tra\_size}} = 11$	3.57	8.17	25.40	98.11	-	2.025
12	• $\delta_{\text{bm\_rng}} = 12$	3.93	9.50	30.64	99.75	-	2.068

Table 6.2: This table contains the experimental results of the evaluation of our feature point tracking system. The computation times of our feature point tracking system were measured for 20, 100, and 400 feature points. They are specified as average computation times per image in milliseconds in the third, fourth, and fifth column. The remaining columns refer to the test runs with 400 feature points. The sixth column (atl) contains the average length of the feature trails. The final two columns state the average number of iterations of the gradient descent algorithms for translation estimation and affine motion estimation, respectively.

test image sequence. Although the average trail length does not discriminate between accurate feature trails and erroneous feature trails, it indicates if the basin of convergence and the robustness of the applied algorithms are sufficient for the test image sequence. A visual inspection of the results of several test runs corroborated this hypothesis by showing very few erroneous feature trails. Figure 6.17 depicts the feature windows computed in one test run in three images of the test image sequence. In a similar vein, the average number of iterations of the affine motion estimation can be interpreted as measure for the accuracy of the translation estimation, because the affine motion estimation is initialized with the results of the translation estimation. However, this inference is only valid for test runs in which the affine motion estimation was performed with the same algorithm.

The first three test runs presented in Table 6.2 provide a comparison of three algorithms for affine motion estimation with intensity equalization. The average trail length produced by all three algorithms is almost identical. In contrast to this, the algorithm "gd aff iie" requires less iterations and provides a better computational efficiency than the other two algorithms. The test runs 3, 4, and 5 illustrate how increasing the value of $\delta_{\text{tra\_min}}$ reduces the number of iterations of the algorithm for translation estimation. What is more, test run 5 represents the most efficient config-

uration for the tracking of 400 feature points in Table 6.2. However, the decrease of the average trail length in test run 5 indicates that the respective parameter setting starts to affect the success rate of the tracking. The functionality of the parameter $\delta_{\text{tra\_min}}$ is detailed in Subsection 3.3.5.

Compared to test run 3, test run 6 uses larger feature windows for translation estimation. As a result, the computation times increase considerably. Furthermore, the average number of iterations of the algorithm for affine motion estimation is reduced, which suggests that the accuracy of the translation estimation is slightly better. The average trail length is already very good for test run 3. Therefore, the margin for its improvement is very small, so that the increased feature window size has only a negligible impact on the average trail length for the current test image sequence. In test run 7, we replace the algorithm "gd tra" with the algorithm "gd tra iie" with integrated intensity compensation. Comparing test run 7 to test run 3 shows that this replacement also results in longer computation times, higher accuracy, and a negligible increase of the average trail length. Consequently, the robustness of the algorithm "gd tra" without intensity compensation is sufficient to cope with the intensity changes encountered in consecutive images of the test image sequence.

The remaining test runs evaluate the performance of the block matching algorithm "bm coef". A comparison of the average trail lengths of the test runs 8 and 9 clearly shows that the mismatch prevention strategy described in Subsection 3.4.3 considerably improves the performance of the block matching algorithm for image sequences of real-world scenes. Interestingly, the setup of the experiments in the preceding subsection prevented the detection of this behavior. Compared to the gradient descent algorithm in test run 4, the block matching algorithm in test run 9 achieves similar results for the average trail length and the accuracy of the feature positions. However, the block matching algorithm requires less computation time per image, but more computation time per feature point. As a consequence, its computational efficiency is better for 20 feature points, but worse for 400 feature points. The omission of the smoothing of the input images further reduces the computational overhead per image in test run 10, which represents the most efficient configuration for the tracking of 20 feature points in Table 6.2. Finally, the test runs 11 and 12 illustrate the effects of increasing the feature window size and the search range of the block matching algorithm on the computation times of the feature point tracking system.

6.1.4 Summary

The experimental evaluation in Subsection 6.1.2 concentrates on the accuracy and the basin of convergence of the individual motion estimation algorithms. All implemented algorithms achieve subpixel accuracy in the performed experiments. While the basin of convergence of the block matching algorithms is determined by the size of the search range, the basin of convergence of the gradient descent algorithms for

translation estimation can be increased with a hierarchical approach. As the algorithms for affine motion estimation are exclusively used to refine the feature positions estimated by the algorithms for translation estimation in our feature point tracking system, their small basin of convergence is not detrimental.

The experiments in Subsection 6.1.3 evaluate the feature point tracking system with respect to its computational efficiency and its robustness to intensity changes. The most important results can be summarized as follows:

- The algorithm "gd aff iie" exhibits the best computational efficiency of all algorithms for affine motion estimation with intensity equalization.
- The algorithm "gd tra" without intensity equalization is sufficiently robust to small intensity changes in consecutive images.
- When the number of tracked feature points is large, the gradient descent algorithm "gd tra" is the most efficient algorithm for translation estimation. However, due to its low computational overhead per input image, the block matching algorithm "bm coef" takes first place when the number of tracked feature points is small.

Both the test images and the test image sequence in this section were selected to facilitate the evaluation of certain aspects of our feature point tracking system. We complement this evaluation with further experiments using image sequences of more complex real-world scenes in Section 6.4.

6.2 Structure and Motion Estimation

The experimental evaluation of our work on structure and motion estimation in this section is based on artificial test data, because this approach allows the generation of specific configurations of scene geometry and camera motion, as well as the customization of the associated feature trails. In addition to that, this approach facilitates the quantitative evaluation of the algorithms. We start this section with a description of the generation of the test data in Subsection 6.2.1. In the second subsection, we use one data set to demonstrate the performance of the POSIT algorithm with virtual reference point, which has been proposed in Subsection 4.2.2. Subsection 6.2.3 is dedicated to the systematic experimental evaluation of our structure and motion estimation system with the generated test data. Moreover, we use three image sequences of real-world scenes to evaluate our structure and motion estimation system in conjunction with our feature point tracking system in Section 6.4.

6.2.1 Experimental Setup

Our approach for generating the artificial test data for the evaluation of our algorithms can be divided into three steps. First, the scene structure is established by

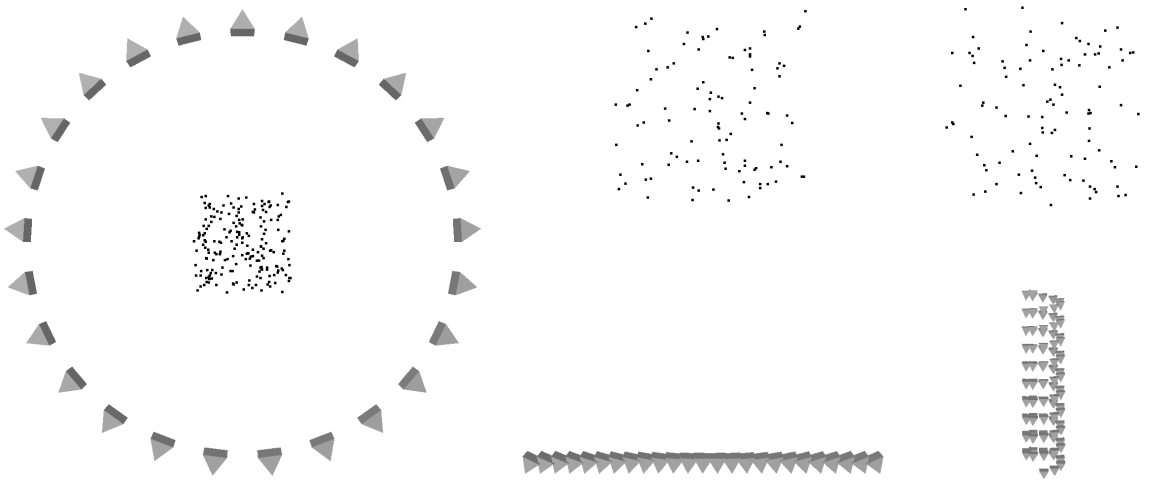


Figure 6.18: This figure illustrates three different artificial test scenes. From left to right, the images show the test scene "circle" with 25 views and 200 feature points, the test scene "simple" with 25 views and 100 feature points, and the test scene "wobble" with 100 views and 100 feature points.

generating a specified number of feature positions in the world coordinate system. Second, the camera motion is defined by generating one camera pose for every image of the (virtual) image sequence. Finally, the feature points are projected into the sensor coordinate systems defined by the camera poses to generate the feature trails. In the following, we describe each of these three steps in more detail.

We generate the coordinates of the feature points by randomly sampling a uniform distribution in a given interval of every axis of the world coordinate system. This approach requires the specification of the number N_u of feature points and the borders of the three intervals. The resulting feature points lie in a cuboid, whose size and position are determined by the borders of the intervals. By default, we sample the feature points of our test scenes in a unit cube at the center of the world coordinate system. In some experiments, we deviate from this default, for example, to evaluate the effects of a planar scene structure.

For the generation of the camera positions, we use a parametric definition of the trajectory of the optical center of the camera. Furthermore, we specify the trajectory of one additional point on the optical axis to fix the orientation of the camera. Technically, we use these two trajectories (and the convention that the x-axis of the camera coordinate system is perpendicular to the y-axis of the world coordinate system) to compute the extrinsic camera parameters for every view. This approach makes it possible to adjust the number M of views, while retaining the basic shape of the camera trajectory.

The feature trails are generated by projecting the feature points into the sensor coordinate system of every view according to equation (2.7). Furthermore, we apply the radial distortion defined in (2.8). In all experiments of this section, we simulate

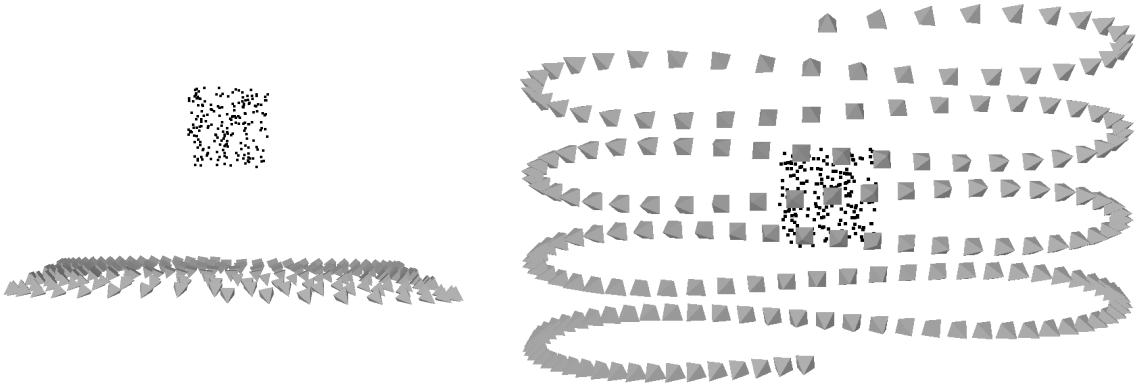


Figure 6.19: The test scene "slalom" is shown with 200 views and 200 feature points.

the digital camera Sony DFW-VL 500 with the help of the intrinsic camera parameters

$$\boldsymbol{K} = \begin{pmatrix} 770 & 0 & 320 \\ 0 & 770 & 240 \\ 0 & 0 & 1 \end{pmatrix}, \quad D_1 = -0.275\,, \quad D_2 = 0.320\,. \tag{6.1}$$

In order to make our experiments more realistic, we perturb the computed feature positions. To this end, we emulate the accuracy of the feature point tracking algorithms by independently adding Gaussian noise with a standard deviation of σ_{inl} to the coordinates of the feature positions. Furthermore, we simulate tracking errors by adding Gaussian noise with a higher standard deviation of σ_{out} to the coordinates of a specified percentage p_{out} of the feature positions. Finally, the created feature trails are split into independent parts to simulate the loss of feature points and to limit the length of the feature trails. The probability of a split between any two consecutive elements of the feature trail is given by p_{loss} . As a consequence, the actual number N of feature points processed by our structure and motion estimation system is larger than the number N_u of unique feature points of the test scene for $p_{\text{loss}} > 0$. However, the total number MN_u of feature positions remains constant for different values of p_{loss}.

We employ five different test scenes for the experimental evaluation discussed in this section. By default, the feature points are uniformly distributed in a unit cube in all five test scenes. Three of these test scenes are illustrated in Figure 6.18. For the evaluation of the POSIT algorithm in Subsection 6.2.2, we generate different sets of input data from the test scene "circle", which features a circular camera trajectory. The remaining test scenes are used for the evaluation of our structure and motion estimation system in Subsection 6.2.3. In the test scene "simple", the camera moves in a straight line that is approximately perpendicular to the optical axis of the camera. This configuration is particularly well suited for structure and motion

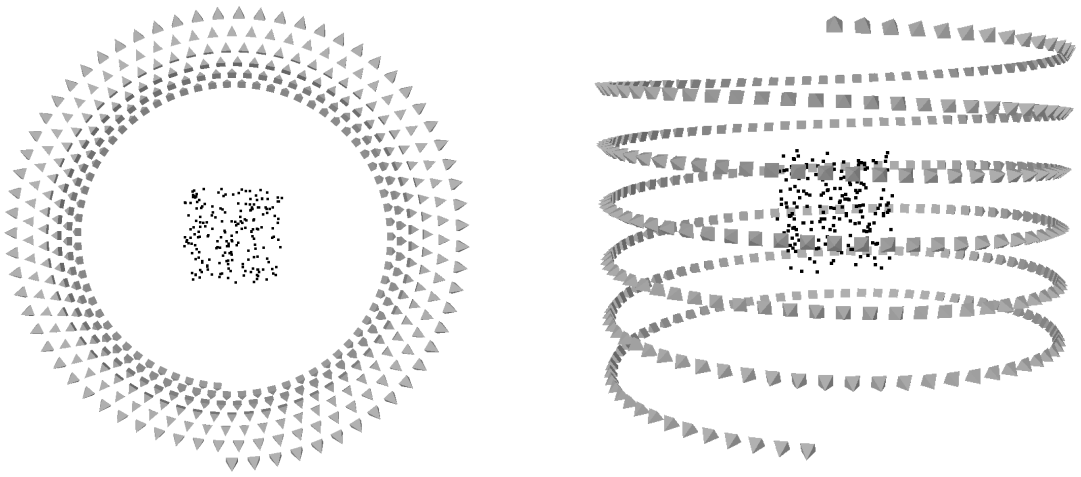


Figure 6.20: The test scene "spiral" is depicted with 400 views and 200 feature points.

estimation. In contrast to this, the dominance of the forward motion of the camera in the test scene "wobble" makes an accurate reconstruction of the scene structure very difficult.

The two remaining test scenes are illustrated in Figure 6.19 and Figure 6.20, respectively. As the basic camera motion of the test scene "slalom" provides a fairly uniform sampling of viewpoints, similar camera motions are often performed with hand-held cameras for the generation of image-based models. In contrast to this, the camera motion of the test scene "spiral" requires the use of a turntable in practice. We employ this test scene to evaluate the performance of our structure and motion estimation system for very long image sequences and high feature loss probabilities, i. e., short feature trails.

The parameters σ_{inl} and σ_{out} specify the amount of noise in the coordinates of the feature positions in pixels. However, a meaningful analysis of the experimental results requires that the amount of noise is considered in relation to the dispersion of the feature points in the test images. In our test scenes, we simulate images with a resolution of 640×480 pixels. Furthermore, the feature points are guaranteed to lie completely inside the image associated with the first view of every test scene. Figure 6.21 illustrates the positions of all feature points in the first view of the test scene "circle". In addition to that, this image visualizes the effects of setting σ_{inl} to 0.5 pixels on the positions of the feature points. We also provide the first image of the test scene "slalom" in Figure 6.22. In this image, the standard deviation of the noise in the feature positions equals 2.0 pixels. Finally, Figure 6.23 shows the first image of the test scene "wobble". Here, outliers are simulated by strongly perturbing the coordinates of 20% of the feature positions.

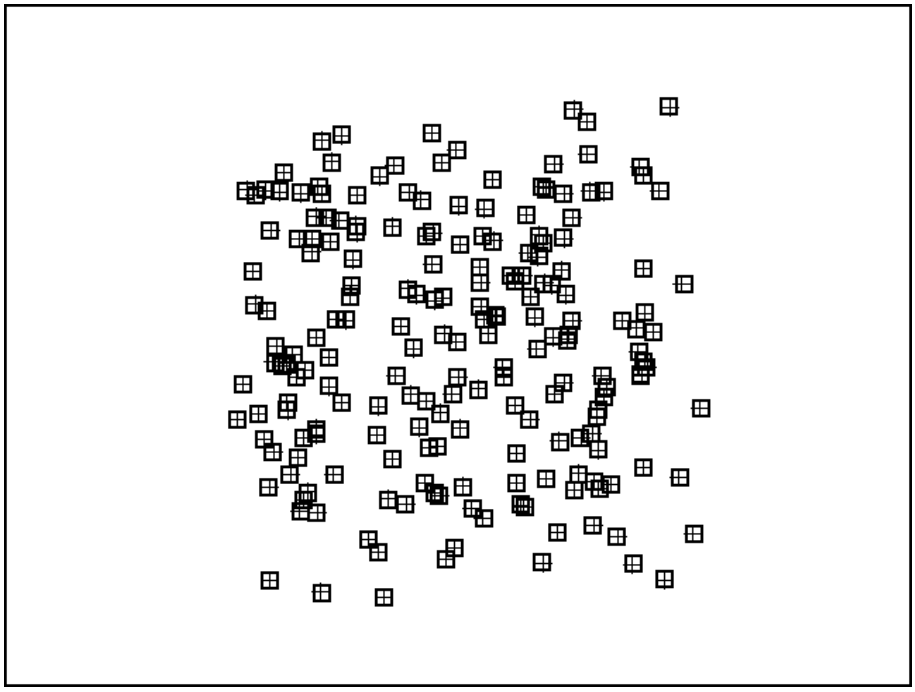


Figure 6.21: The first image of the test scene "circle" has a resolution of 640×480 pixels and contains 200 feature points. The correct feature positions are marked with a box, while the perturbed feature positions are marked with a cross. The specified values for the perturbation parameters are $\sigma_{\text{inl}} = 0.5$, $\sigma_{\text{out}} = 0.0$, and $p_{\text{out}} = 0.0$.

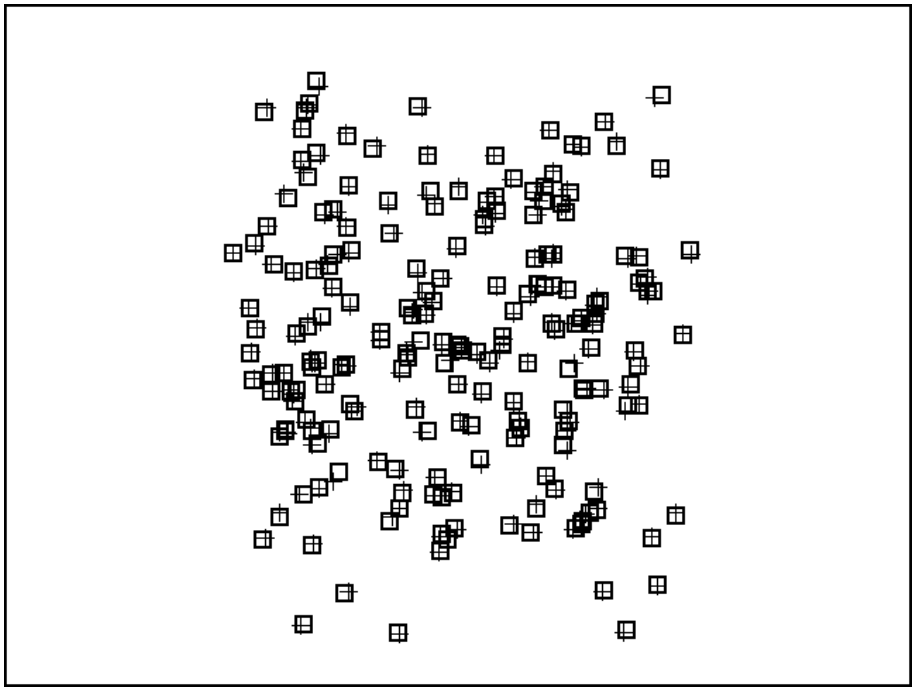


Figure 6.22: The first image of the test scene "slalom" contains 200 feature points. The perturbation parameters are set to $\sigma_{\text{inl}} = 2.0$, $\sigma_{\text{out}} = 0.0$, and $p_{\text{out}} = 0.0$.

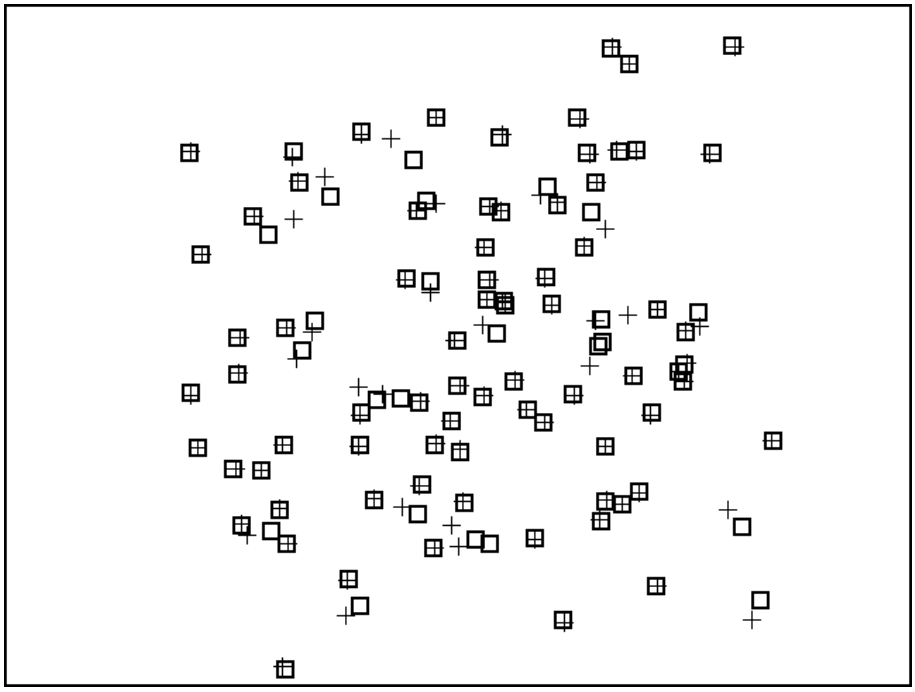


Figure 6.23: The first image of the test scene "wobble" contains 200 feature points. The perturbation parameters are set to $\sigma_{\text{inl}} = 1.0$, $\sigma_{\text{out}} = 10.0$, and $p_{\text{out}} = 0.2$.

6.2.2 Evaluation of the POSIT Algorithm

The standard POSIT algorithm and our proposed extension, the POSIT algorithm with virtual reference point, are both described in Subsection 4.2.2. The experimental setup for the comparison of these two algorithms is based on the test scene "circle" with 100 views and 500 feature points. The algorithms were run with 10000 sets of a specified number N of randomly selected feature points for each of the 100 views, resulting in a total of one million estimations per parameter configuration. For the standard POSIT algorithm, the reference point was also randomly determined from the set of selected feature points.

The POSIT algorithm estimates the extrinsic camera parameters $(\boldsymbol{R}, \boldsymbol{t})$ from the positions of N feature points given both in the world coordinate system and one image coordinate system. The orientation and the position of the camera can be derived from the extrinsic camera parameters as

$$\tilde{\boldsymbol{R}} = \boldsymbol{R}^{\mathrm{T}}, \quad \tilde{\boldsymbol{t}} = -\boldsymbol{R}^{\mathrm{T}}\boldsymbol{t}\,. \tag{6.2}$$

When the ground truth for the camera pose is given by $\hat{\boldsymbol{R}}$ and $\hat{\boldsymbol{t}}$, we compute the absolute translation error of the camera position as

$$\epsilon_{\text{tra}} = \|\hat{\boldsymbol{t}} - \tilde{\boldsymbol{t}}\|_2\,. \tag{6.3}$$

Furthermore, we define the absolute rotation error ϵ_{rot} as the rotation angle encoded

type	N	σ_{inl}	$\epsilon_{\text{tra}}^{\langle 05 \rangle}$	$\epsilon_{\text{tra}}^{\langle 50 \rangle}$	$\epsilon_{\text{tra}}^{\langle 95 \rangle}$	$\epsilon_{\text{rot}}^{\langle 50 \rangle}$	time
std	10	0.2	0.0013	0.0037	0.0083	0.0838°	0.040
std	10	2.0	0.0133	0.0368	0.0828	0.8378°	0.041
std	100	0.2	0.0006	0.0019	0.0046	0.0420°	0.069
std	100	2.0	0.0061	0.0188	0.0462	0.4199°	0.070
vrp	10	0.2	0.0012	0.0032	0.0069	0.0728°	0.042
vrp	10	2.0	0.0116	0.0317	0.0694	0.7283°	0.042
vrp	100	0.2	0.0003	0.0008	0.0016	0.0189°	0.068
vrp	100	2.0	0.0032	0.0083	0.0155	0.1888°	0.067

Table 6.3: This table details the results of our experimental evaluation of two variants of the POSIT algorithm. The standard POSIT algorithm (std) uses one input feature point as the reference point for the weak perspective projection model, whereas the POSIT algorithm with virtual reference point (vrp) uses the center of mass of the feature points for this purpose. The second column contains the number of feature points in the input data. The parameter σ_{inl} is described in Subsection 6.2.1. The error measures in columns four to seven are defined in the text below. The last column of this table specifies the mean computation time per estimation in milliseconds.

in the rotation matrix

$$\Delta \boldsymbol{R} = \hat{\boldsymbol{R}} \tilde{\boldsymbol{R}}^{\mathrm{T}}, \tag{6.4}$$

which represents the difference between the ground truth orientation and the estimated orientation of the camera.

The results of our experimental evaluation are presented in Table 6.3. We measure the accuracy of the algorithms with the median $\epsilon_{\text{tra}}^{\langle 50 \rangle}$ of the absolute translation errors and the median $\epsilon_{\text{rot}}^{\langle 50 \rangle}$ of the absolute rotation errors. We also indicate the variation of the measurements with the 5th percentile $\epsilon_{\text{tra}}^{\langle 05 \rangle}$ and the 95th percentile $\epsilon_{\text{tra}}^{\langle 95 \rangle}$ of the absolute translation errors. The accuracy of both variants of the POSIT algorithm increases with the number of input feature points and decreases with the amount of noise perturbing the feature positions. When applied to ten feature points, our proposed POSIT algorithm with virtual reference point consistently exhibits a higher accuracy than the standard algorithm. As explained in Subsection 4.2.2, the influence of the noise in the feature positions on the position of the virtual reference point decreases with the number of the input feature points. Consequently, the POSIT algorithm with virtual reference point is considerably more accurate than the standard POSIT algorithm when the input data consists of 100 feature points. Finally, the last column of Table 6.3 shows that the differences in the computation times of the two variants of the POSIT algorithm are negligible.

position	param.	description	values	reference
X|-----|-	$\delta_{\text{seg\_vra}}$	compute view ray angles	-/A	Subs. 4.3.2
-|X---|-	$\delta_{\text{bun\_type}}$	optimize LMedS estimation	-/B	Table 4.2
-|-X--|-	$\delta_{\text{bun\_type}}$	optimize single segments	-/B	Figure 4.16
-|--X-|-	$\delta_{\text{bun\_type}}$	optimize merged segments	-/B	Figure 4.16
-|---X|-	$\delta_{\text{bun\_type}}$	optimize reconstruction	-/B	Figure 4.16
-|-----|X	$\delta_{\text{rob\_type}}$	specify M-estimator	-/C/F/H	Subs. 4.3.4

Table 6.4: The configuration code of our structure and motion estimation system corresponds to three user-specified parameters. The computation of the view ray angles and the applications of our bundle adjustment approach can be deactivated (-) or activated (A/B) independently. When one of the three supported M-estimators (Cauchy, Fair, Huber) is specified, it is used in every activated application of our bundle adjustment approach.

6.2.3 Evaluation of the Estimation System

Our structure and motion estimation system is described in detail in Section 4.3. In particular, the default values of its user-specified parameters are enumerated in Table 4.3 in Subsection 4.3.5. In order to simplify the presentation of the experimental results in this subsection, we use a configuration code to summarize the values of three important user-specified parameters of our structure and motion estimation system. The configuration code is explained in Table 6.4.

6.2.3.1 Basic Approach

The output of our structure and motion estimation system consists of feature positions, which represent the structure of the scene, and extrinsic camera parameters, which represent the camera motion. It is difficult to evaluate the accuracy of the reconstruction with the feature positions, because a combination of noisy input data and short feature trails often leads to inevitable outliers. These outliers complicate several aspects of the evaluation, including the necessary alignment of the reconstruction and the ground truth data. Therefore, we evaluate the accuracy of our structure and motion estimation system with the extrinsic camera parameters. First, we have to transform the reconstruction into the world coordinate system of the ground truth data. The required transformation parameters comprise a rotation, a translation, and a scale factor. For the estimation of the transformation parameters, we consider the optical centers of the camera in the reconstruction and the ground truth data. When the reconstructed optical centers form point set $\mathcal{A}$, and the ground truth optical centers form point set $\mathcal{B}$, corresponding optical centers simply represent the same view in the reconstruction and the ground truth data. As a consequence, we are able to estimate the transformation parameters in closed form with the algorithm described in Subsection 5.4.

For the evaluation of the accuracy of the reconstructed camera motion, we adopt the error measures used in [Sch08a]. When the estimated and transformed camera pose of one view m is given by the rotation matrix $\tilde{\boldsymbol{R}}_m$ and the translation vector $\tilde{\boldsymbol{t}}_m$, and the ground truth camera pose of this view is given by the rotation matrix $\hat{\boldsymbol{R}}_m$ and the translation vector $\hat{\boldsymbol{t}}_m$, the absolute pairwise translation error of the view pair (m_1, m_2) is defined as

$$\epsilon_{\text{apt}}(m_1, m_2) = \|(\hat{\boldsymbol{t}}_{m_1} - \hat{\boldsymbol{t}}_{m_2}) - (\tilde{\boldsymbol{t}}_{m_1} - \tilde{\boldsymbol{t}}_{m_2})\|_2 \,. \tag{6.5}$$

A normalization of this error measure with the ground truth distance of the two camera positions yields the relative pairwise translation error

$$\epsilon_{\text{rpt}}(m_1, m_2) = \frac{\|(\hat{\boldsymbol{t}}_{m_1} - \hat{\boldsymbol{t}}_{m_2}) - (\tilde{\boldsymbol{t}}_{m_1} - \tilde{\boldsymbol{t}}_{m_2})\|_2}{\|\hat{\boldsymbol{t}}_{m_1} - \hat{\boldsymbol{t}}_{m_2}\|_2} \,. \tag{6.6}$$

Similarly, the absolute pairwise rotation error $\epsilon_{\text{apr}}(m_1, m_2)$ is given by the angle of the rotation matrix

$$\Delta\boldsymbol{R}_{m_1,m_2} = \left(\hat{\boldsymbol{R}}_{m_1}\hat{\boldsymbol{R}}_{m_2}^{\mathrm{T}}\right)\left(\tilde{\boldsymbol{R}}_{m_1}\tilde{\boldsymbol{R}}_{m_2}^{\mathrm{T}}\right)^{\mathrm{T}} . \tag{6.7}$$

For every reconstruction, we compute the average values of the error measures with 10000 randomly selected view pairs, which yields the error measures $\bar{\epsilon}_{\text{apt}}$, $\bar{\epsilon}_{\text{rpt}}$, and $\bar{\epsilon}_{\text{apr}}$. In addition to that, we perform 100 reconstructions for every parameter configuration with different seeds for the random number generator, which influences both the generation of the test scenes and the random sampling step of the LMedS technique. Consequently, we evaluate our structure and motion estimation system with the median of the average relative pairwise translation error, the median of the average absolute pairwise translation error, and the median of the average absolute pairwise rotation error. In addition to that, we use the 5th percentile and the 95th percentile of the average relative pairwise translation error to check the variation of the errors.

Our final error measure is the root mean squared back-projection error ϵ_{rbp}. It is computed as the quadratic mean of the back-projection errors of all available feature points in all available views. The squared back-projection error of a single feature point is defined in equation (4.45). The root mean squared back-projection error has some interesting properties. First of all, it can be computed without any ground truth data. Furthermore, it indicates the amount of noise in the feature positions when the reconstruction was successful. Finally, this error measure is closely related to the cost function of our bundle adjustment approach, which is defined in (4.47). We use it to analyze the performance of the implemented M-estimators.

6.2.3.2 Test Scene Simple

Our first three experiments are concerned with the reconstruction of the test scene "simple". This test scene is illustrated in the middle image of Figure 6.18. The trajectory of the camera is a straight line and has a length of two units in all three

code	N_u	$\epsilon_{\text{rbp}}^{\langle 50 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 05 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 50 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 95 \rangle}$	$\bar{\epsilon}_{\text{apt}}^{\langle 50 \rangle}$	$\bar{\epsilon}_{\text{apr}}^{\langle 50 \rangle}$	time
-|----|-	25	0.86	2.76%	3.48%	4.95%	0.0086	0.3110°	213
-|----|-	100	0.76	0.92%	1.09%	1.69%	0.0028	0.1204°	315
-|----|-	400	0.76	0.43%	0.50%	1.41%	0.0014	0.0707°	728
-|B---|-	25	0.76	2.56%	2.89%	3.28%	0.0069	0.2221°	253
-|B---|-	100	0.71	0.85%	0.96%	1.12%	0.0024	0.0765°	469
-|B---|-	400	0.71	0.39%	0.44%	0.50%	0.0011	0.0348°	1378
-|---B|-	25	0.67	1.57%	1.68%	1.81%	0.0040	0.1184°	706
-|---B|-	100	0.70	0.76%	0.82%	0.90%	0.0019	0.0562°	1466
-|---B|-	400	0.71	0.39%	0.42%	0.46%	0.0010	0.0286°	5052

Table 6.5: The results in this table are based on the test scene "simple" with 100 views and a varying number N_u of unique feature points. The feature trails were generated with the parameter values $\sigma_{\text{inl}} = 0.5$, $\sigma_{\text{out}} = 0.0$, $p_{\text{out}} = 0.0$, and $p_{\text{loss}} = 0.0$. The last column of the table specifies the median value of the computation times of one hundred reconstructions in milliseconds.

experiments. The purpose of the experiments is to analyze the performance of the algorithms in our structure and motion estimation system with respect to the reconstruction of a single segment. Thus, we enforce the creation of a single segment by increasing the maximum segment length to $\delta_{\text{frm\_max}} = 1000$ and by disabling the computation of the view ray angles. It is important to note that the three applications of our bundle adjustment approach that handle the optimization of the single segments, the optimization of the merged segments, and the optimization of the reconstruction perform identical operations when only one segment is created. All other user-specified parameters that are not part of the configuration code detailed in Table 6.4 were set to their default values listed in Table 4.3.

The purpose of our first experiment is to analyze the effects of the number of feature points on the accuracy and the computational efficiency of our structure and motion estimation system. As detailed in Table 6.5, we evaluate three different configurations of our estimation system. For all three configurations, the accuracy of the estimated camera poses increases with the number N_u of feature points. When we compare the configurations to each other, the first configuration, which does not apply bundle adjustment, exhibits the lowest accuracy. Independently optimizing the LMedS estimation of the outer motion, the inner motion, and the triangulation of the feature points moderately increases the accuracy of the reconstruction. The third configuration, which uses our bundle adjustment approach to optimize all extrinsic camera parameters and all feature positions of the segment at once, achieves the highest accuracy.

For all experiments in this subsection, the median value of the computation times of the 100 performed reconstructions is specified in milliseconds in the last column of the respective table. In Table 6.5, the least accurate configuration exhibits the best

code	M	$\epsilon_{\text{rbp}}^{\langle 50 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 05 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 50 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 95 \rangle}$	$\bar{\epsilon}_{\text{apt}}^{\langle 50 \rangle}$	$\bar{\epsilon}_{\text{apr}}^{\langle 50 \rangle}$	time
-|----|-	25	0.77	0.60%	0.82%	1.49%	0.0030	0.1307°	99
-|----|-	100	0.76	0.92%	1.09%	1.69%	0.0028	0.1204°	314
-|----|-	400	0.75	1.26%	1.43%	2.16%	0.0029	0.1211°	1182
-|B---|-	25	0.69	0.53%	0.66%	0.84%	0.0023	0.0833°	142
-|B---|-	100	0.71	0.85%	0.96%	1.12%	0.0024	0.0765°	472
-|B---|-	400	0.71	1.16%	1.26%	1.39%	0.0024	0.0752°	1841
-|---B|-	25	0.69	0.49%	0.57%	0.67%	0.0019	0.0608°	174
-|---B|-	100	0.70	0.76%	0.82%	0.90%	0.0019	0.0562°	1570
-|---B|-	400	0.71	1.03%	1.08%	1.13%	0.0019	0.0554°	35547

Table 6.6: The results in this table are based on the test scene "simple" with a varying number M of views and 100 feature points. The feature trails were generated with the parameter values $\sigma_{\text{inl}} = 0.5$, $\sigma_{\text{out}} = 0.0$, $p_{\text{out}} = 0.0$, and $p_{\text{loss}} = 0.0$. The last column of the table specifies the median value of the computation times of one hundred reconstructions in milliseconds.

computational efficiency. Expectedly, the optimization of the complete segment considerably increases the computation times of the third configuration. Nevertheless, a computation time of approximately five seconds for a test scene with 100 views and 400 feature points should be sufficient for most applications. It is interesting to note that the computation times increase sublinearly with the number of feature points for all three configurations.

Unlike the first experiment, our second experiment is performed with a varying number M of views and a fixed number N_u of feature points. Its results are presented in Table 6.6. Surprisingly, the average relative pairwise translation error increases with the number of views for all three configurations. In contrast to this, the average absolute pairwise translation error remains approximately constant. The definitions of these error measures in (6.5) and (6.6) show that the increase of the average relative pairwise translation error has to be caused by the decrease of the average distance of the camera positions in the view pairs. The average absolute pairwise rotation error slightly decreases with the number of views, which substantiates the notion that the number of the views only has a negligible effect on the accuracy of the estimated camera poses. As in the first experiment, the accuracy achieved by the three configurations increases in the order of their appearance in Table 6.6.

The computation times of all three configurations of our structure and motion estimation system are more sensitive to the number of views than to the number of feature points. Table 6.6 shows that the computation times of the first two configurations increase approximately linearly with the number of views. In contrast to this, the computation times of the third configuration reflect that the computational complexity of our bundle adjustment approach is governed by the cube of the num-

code	depth	$\epsilon_{\text{rbp}}^{\langle 50 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 05 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 50 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 95 \rangle}$	$\bar{\epsilon}_{\text{apt}}^{\langle 50 \rangle}$	$\bar{\epsilon}_{\text{apr}}^{\langle 50 \rangle}$	time
-|----|-	0.4	0.77	1.18%	1.60%	2.26%	0.0056	0.2029°	109
-|----|-	0.2	0.78	2.28%	3.05%	4.30%	0.0107	0.3490°	115
-|----|-	0.0	1.05	11.2%	14.1%	18.8%	0.0468	1.4034°	119
-|B---|-	0.4	0.69	1.05%	1.26%	1.62%	0.0044	0.1381°	184
-|B---|-	0.2	0.69	1.53%	1.92%	2.54%	0.0066	0.2046°	195
-|B---|-	0.0	0.69	2.17%	2.63%	3.67%	0.0090	0.2771°	199
-|---B|-	0.4	0.69	0.92%	1.16%	1.33%	0.0040	0.1135°	185
-|---B|-	0.2	0.69	1.46%	1.75%	2.02%	0.0059	0.1673°	193
-|---B|-	0.0	0.69	2.08%	2.37%	2.87%	0.0080	0.2272°	219

Table 6.7: The results in this table are based on the test scene "simple" with 25 views and 100 feature points. The depth of the cuboid containing the feature points was reduced to the values shown in the second column of the table to simulate different levels of scene planarity. The feature trails were generated with the parameter values $\sigma_{\text{inl}} = 0.5$, $\sigma_{\text{out}} = 0.0$, $p_{\text{out}} = 0.0$, and $p_{\text{loss}} = 0.0$.

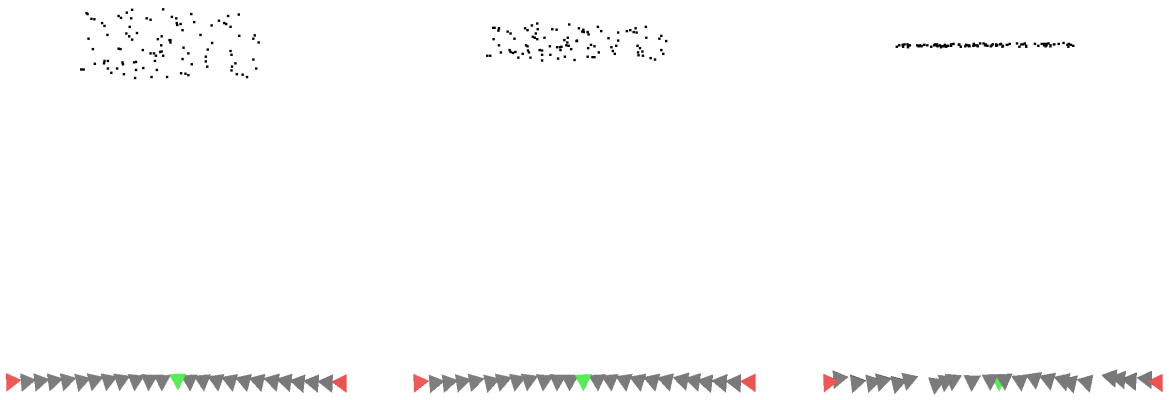


Figure 6.24: The three exemplary reconstructions of the test scene "simple" with 25 views and 100 feature points correspond to the first three lines of Table 6.7. Consequently, from left to right, the depth of the cuboid containing the feature points is 0.4 units, 0.2 units, and 0.0 units. In the same order, the reconstructions exhibit average relative pairwise translation errors of 2.07%, 3.14%, and 13.4%, respectively.

ber of views. As a consequence, when our structure and motion estimation system is configured to perform the optimization of the single segments or the optimization of the merged segments, its computational efficiency strongly depends on the size of the segments.

In our third experiment, which is based on the test scene "simple" with 25 views and 100 feature points, we simulate different levels of scene planarity by reducing the depth of the cuboid containing the feature points. The results of this experiment are presented in Table 6.7. In addition to that, three exemplary reconstructions,

which correspond to the first three lines of Table 6.7, are depicted in Figure 6.24. For all three configurations of our estimation system, the accuracy of the estimated camera poses decreases when the depth of the scene structure is reduced. In particular, the average relative pairwise translation error of the first configuration is very high when the scene structure is planar. As described in Subsection 4.3.4, the POSIT algorithm fails to refine the camera poses estimated by the three-point algorithm when all feature points lie on a single plane. Consequently, the noise in the feature positions has a much stronger effect on the estimated camera poses. The second configuration, which refines the estimated camera poses with our bundle adjustment approach, does not exhibit this behavior. The computation times of the third configuration in Table 6.7 benefit from the small number of views in the test scene. As a result, this configuration provides a highly competitive combination of accuracy and computational efficiency in the current experiment.

6.2.3.3 Test Scene Slalom

The following three experiments are based on the test scene "slalom", which is illustrated in Figure 6.19. The trajectory of the camera has a width of five units and a height of three units. In all three experiments, the generated test scene comprises 200 views and 200 unique feature points. Furthermore, we specify a relatively small feature loss probability of $p_{\text{loss}} = 0.01$, which leads to the generation of approximately 600 feature trails with an average length of 67 frames. Our structure and motion estimation system reconstructs one feature point for every input feature trail. Consequently, on average, three reconstructed feature points share the same ground truth position.

The purpose of our first experiment with the test scene "slalom" is to verify the performance of our key frame selection algorithm, which has been proposed in Subsection 4.3.2. To this end, we reconstruct the test scene with two different configurations of our structure and motion estimation system. In the first configuration, we deactivate the computation of the view ray angles, which yields the alternate key frame selection algorithm described at the end of Subsection 4.3.2. The second configuration uses our proposed key frame selection algorithm, which considers the computed view ray angles to improve the quality of the selected key frames. We evaluate both configurations with different values of the maximum number of frames in a segment. Furthermore, we deactivate all applications of our bundle adjustment approach to highlight the differences between the two key frame selection algorithms. The results of this experiment are presented in Table 6.8.

For all values of the maximum number $\delta_{\text{frm\_max}}$ of frames in a segment, the computed error measures indicate that the computation of the view ray angles in our key frame selection algorithm improves the accuracy of the subsequent reconstruction. This observation is independent of the number of created segments, because the second configuration creates a smaller average number of segments than the first configuration for $\delta_{\text{frm\_max}} = 32$, but a larger average number of segments for

code	$\delta_{\text{frm\_max}}$	$\epsilon_{\text{rbp}}^{\langle 50 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 05 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 50 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 95 \rangle}$	$\bar{\epsilon}_{\text{apr}}^{\langle 50 \rangle}$	$\bar{T}$	time
-|----|-	32	3.25	2.84%	5.14%	10.5%	1.9310°	14.00	1519
-|----|-	48	2.70	2.09%	4.10%	8.04%	1.6939°	9.00	1188
-|----|-	64	2.32	1.74%	3.60%	7.67%	1.3461°	7.00	1082
-|----|-	96	3.06	2.62%	4.58%	9.32%	1.6412°	5.00	967
-|----|-	128	2.39	1.82%	3.48%	6.53%	1.4086°	4.00	894
A|----|-	32	2.33	1.83%	3.43%	5.90%	1.3216°	10.02	3519
A|----|-	48	2.35	1.70%	3.27%	6.46%	1.3234°	8.21	3751
A|----|-	64	2.21	1.84%	3.16%	5.35%	1.2185°	7.86	4025
A|----|-	96	2.05	1.30%	2.75%	6.19%	1.0737°	5.38	3354
A|----|-	128	1.98	1.25%	2.44%	4.93%	0.9937°	4.01	3079

Table 6.8: The results in this table are based on the test scene "slalom" with 200 views and 200 unique feature points. The generation of the feature trails was performed with the parameter values $\sigma_{\text{inl}} = 1.0$, $\sigma_{\text{out}} = 0.0$, $p_{\text{out}} = 0.0$, and $p_{\text{loss}} = 0.01$. The second to last column of the table shows the average number of created segments. The last column specifies the median value of the computation times in milliseconds.

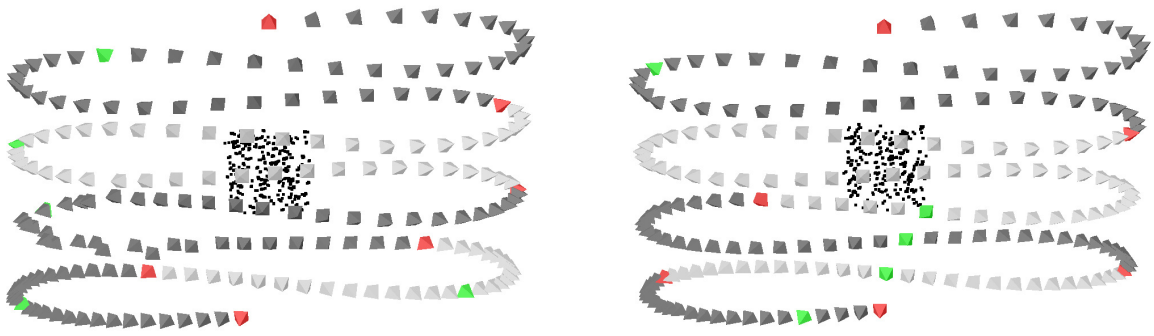


Figure 6.25: The two exemplary reconstructions of the test scene "slalom" correspond to the first and the second configuration in Table 6.8. The specified value of the maximum number of frames in a segment is 96. The average relative pairwise translation errors of the two reconstructions are 4.70% and 2.64%, respectively. We visualize the created segments by alternating between the use of light gray pyramids and dark gray pyramids to represent the views of one segment. The camera trajectory starts at the top of the image in both reconstructions.

$\delta_{\text{frm\_max}} = 64$. While the second configuration specifically selects key frames that facilitate the reconstruction of the corresponding segments, the first configuration has no means to ensure that the selected key frames are suitable for the subsequent reconstruction of the created segments. Figure 6.25 illustrates one exemplary segmentation for each configuration. In particular, the camera positions corresponding to the key frames of the second and the third segment of the first configuration are very close to each other, which decreases the accuracy of the reconstruction of these

no.	code	$\epsilon_{\text{rbp}}^{\langle 50 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 05 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 50 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 95 \rangle}$	$\bar{\epsilon}_{\text{apr}}^{\langle 50 \rangle}$	$\bar{T}$	time
1	A|-----|-	5.18	4.54%	8.18%	15.0%	3.3080°	7.88	3872
2	A|B---|-	2.99	1.33%	1.91%	3.62%	0.6716°	7.88	4751
3	A|-B--|-	2.88	0.80%	0.93%	1.24%	0.2846°	7.88	7547
4	A|--B-|-	2.81	0.70%	0.75%	0.82%	0.2190°	7.88	7593
5	A|B-B-|-	2.81	0.70%	0.75%	0.82%	0.2190°	7.88	8310
6	A|---B|-	2.81	0.69%	0.74%	0.78%	0.2142°	7.88	20224
7	A|--BB|-	2.81	0.69%	0.74%	0.78%	0.2142°	7.88	14397

Table 6.9: The results in this table are based on the test scene "slalom" with 200 views and 200 unique feature points. The generation of the feature trails was performed with the parameter values $\sigma_{\text{inl}} = 2.0$, $\sigma_{\text{out}} = 0.0$, $p_{\text{out}} = 0.0$, and $p_{\text{loss}} = 0.01$.

segments. The computation of the view ray angles prevents this problem in the segmentation of the second configuration. The last column of Table 6.8 shows that the computation of the view ray angles for all eligible key frame pairs considerably increases the computation time of the reconstruction.

In our second experiment with the test scene "slalom", we examine the performance of our estimation system with respect to the different applications of our bundle adjustment approach. In addition to that, we evaluate the robustness of our algorithms to highly inaccurate feature positions by increasing the value of the parameter σ_{inl} to two pixels. The results of this experiment are detailed in Table 6.9. The high level of noise in the input data strongly affects the accuracy of the first configuration, which does not rely on bundle adjustment at all. In exchange for moderately increased computation times, the second configuration provides considerably more accurate reconstructions. When even more computation time is available, the fourth configuration, which optimizes the merged segments, is a very good compromise between accuracy and computational efficiency. A comparison of the fourth and the fifth configuration illustrates that, for the current experimental setup, the reconstruction of the segments without bundle adjustment is sufficiently accurate for the successful merging of the segments and the subsequent optimization of the merged segments. The optimization of the complete reconstruction in the sixth configuration yields only a negligible improvement of the accuracy. Interestingly, the seventh configuration shows that the additional optimization of the merged segments reduces the computation time of the the sixth configuration. Apparently, the more accurate initialization reduces the number of time-consuming iterations of the optimization of the complete reconstruction.

Our final experiment with the test scene "slalom" evaluates the performance of the implemented M-estimators, which are detailed in Subsection 4.3.4. In this experiment, we activate the optimization of the merged segments with our bundle adjustment approach. In order to facilitate the comparison of the implemented M-estimators with the standard cost function of our bundle adjustment approach, we

code	p_{out}	$\epsilon_{\text{rbp}}^{\langle 50 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 05 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 50 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 95 \rangle}$	$\bar{\epsilon}_{\text{apr}}^{\langle 50 \rangle}$	$\bar{T}$	time
A| - - B - | -	0.0	2.81	0.70%	0.75%	0.82%	0.2190°	7.88	7562
A| - - B - | -	0.4	9.15	2.27%	2.45%	2.71%	0.7159°	7.88	10122
A| - - B - |C	0.0	2.81	0.72%	0.77%	0.84%	0.2243°	7.88	9755
A| - - B - |C	0.4	9.26	1.22%	1.32%	1.42%	0.3844°	7.88	15238
A| - - B - |F	0.0	2.81	0.72%	0.76%	0.83%	0.2231°	7.88	9382
A| - - B - |F	0.4	9.20	1.43%	1.52%	1.68%	0.4459°	7.88	12676
A| - - B - |H	0.0	2.82	0.73%	0.78%	0.85%	0.2264°	7.88	8628
A| - - B - |H	0.4	9.23	1.24%	1.33%	1.45%	0.3882°	7.88	12602

Table 6.10: The results in this table are based on the test scene "slalom" with 200 views and 200 unique feature points. The generation of the feature trails was performed with the parameter values $\sigma_{\text{inl}} = 2.0$, $\sigma_{\text{out}} = 10.0$, and $p_{\text{loss}} = 0.01$. The value of the parameter p_{out} is specified in the second column of the table.

first test all configurations without outliers. In addition to that, we evaluate all configurations with input data containing outliers in 40% of the feature positions. The results of this experiment are listed in Table 6.10. When the input data is free of strong outliers, the accuracy of all four configurations is almost identical. However, all three M-estimators increase the computation time of our bundle adjustment approach. When p_{out} is set to 0.4 to simulate a high level of outliers, the accuracy of the first configuration decreases considerably. In contrast to this, the M-estimators retain a much higher level of accuracy. In particular, the Cauchy estimator exhibits the best accuracy, while the Huber estimator provides a well-balanced compromise between accuracy and computational efficiency. As the standard cost function of our bundle adjustment approach minimizes the sum of the squared back-projections errors of the feature points, it is not surprising that the use of the M-estimators actually increases the root mean squared back-projection errors.

6.2.3.4 Test Scene Spiral

The following two experiments are based on the test scene "spiral", which is illustrated in Figure 6.20. The camera trajectory of the test scene has a height of three units and a diameter of four units. We use this test scene, which comprises 400 views and 200 unique feature points, to simulate scene configurations with a large number of views and high feature loss rates. The feature loss probability of $p_{\text{loss}} = 0.04$ in the first experiment leads to the generation of approximately 3400 feature trails with an average length of 24 frames. In the second experiment, we generate feature trails with $p_{\text{loss}} = 0.02$ and $p_{\text{loss}} = 0.06$, which yields average feature trail lengths of 44 frames and 16 frames, respectively.

The results of our first experiment with the test scene "spiral" are presented in Table 6.11. Identical configurations of our structure and motion estimation system

no.	code	$\epsilon_{\text{rbp}}^{\langle 50\rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 05\rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 50\rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 95\rangle}$	$\bar{\epsilon}_{\text{apr}}^{\langle 50\rangle}$	$\bar{T}$	time
1	A| - - - - | -	1.97	1.59%	2.43%	3.68%	1.5570°	20.77	6768
2	A|B- - - | -	1.51	0.57%	0.72%	1.09%	0.4538°	20.77	8193
3	A| -B- - | -	1.44	0.25%	0.35%	0.54%	0.2113°	20.77	10144
4	A| - -B- | -	1.39	0.16%	0.17%	0.18%	0.0954°	20.77	10684
5	A|B-B- | -	1.39	0.16%	0.17%	0.18%	0.0954°	20.77	12021
6	A| - - -B| -	1.39	0.15%	0.16%	0.18%	0.0914°	20.77	24017
7	A| - -BB| -	1.39	0.15%	0.16%	0.18%	0.0914°	20.77	22403

Table 6.11: The results in this table are based on the test scene "spiral" with 400 views and 200 unique feature points. The generation of the feature trails was performed with the parameter values $\sigma_{\text{inl}} = 1.0$, $\sigma_{\text{out}} = 0.0$, $p_{\text{out}} = 0.0$, and $p_{\text{loss}} = 0.04$. The last column shows the median value of the computation times in milliseconds.

code	p_{loss}	$\epsilon_{\text{rbp}}^{\langle 50\rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 05\rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 50\rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 95\rangle}$	$\bar{\epsilon}_{\text{apr}}^{\langle 50\rangle}$	$\bar{T}$	time
A| - -B- | -	0.02	6.41	0.67%	0.70%	0.74%	0.3949°	18.96	12463
A| - -B- | -	0.06	6.24	0.80%	0.88%	1.00%	0.5030°	26.51	11719
A| - -B- |C	0.02	6.56	0.18%	0.18%	0.19%	0.1028°	18.96	24841
A| - -B- |C	0.06	6.56	0.21%	0.23%	0.25%	0.1299°	26.51	25210
A| - -B- |F	0.02	6.50	0.23%	0.24%	0.25%	0.1349°	18.96	19668
A| - -B- |F	0.06	6.41	0.28%	0.31%	0.35%	0.1765°	26.51	19201
A| - -B- |H	0.02	6.53	0.19%	0.20%	0.21%	0.1109°	18.96	21701
A| - -B- |H	0.06	6.47	0.23%	0.25%	0.28%	0.1435°	26.51	23625

Table 6.12: The results in this table are based on the test scene "spiral" with 400 views and 200 unique feature points. The generation of the feature trails was performed with the parameter values $\sigma_{\text{inl}} = 1.0$, $\sigma_{\text{out}} = 10.0$, and $p_{\text{out}} = 0.2$. The value of the parameter p_{loss} is specified in the second column of the table.

have already been evaluated in an experiment with the test scene "slalom". The results of this experiment can be found in Table 6.9. A comparison of the results of both experiments shows that the relative performance of the configurations with respect to their accuracy and their computational efficiency is approximately the same in both experiments. In the current experiment, the reduced noise in the feature positions of the feature trails results in a higher accuracy of all configurations. Furthermore, this experiment proves that our structure and motion estimation system is perfectly capable of efficiently processing test scenes with a large number of views and high feature loss rates.

Our second experiment with the test scene "spiral" also corresponds to an experiment with the test scene "slalom". Both experiments evaluate the performance of the implemented M-estimators. In the current experiment, we keep the outlier probability constant and generate the feature trails with two different feature

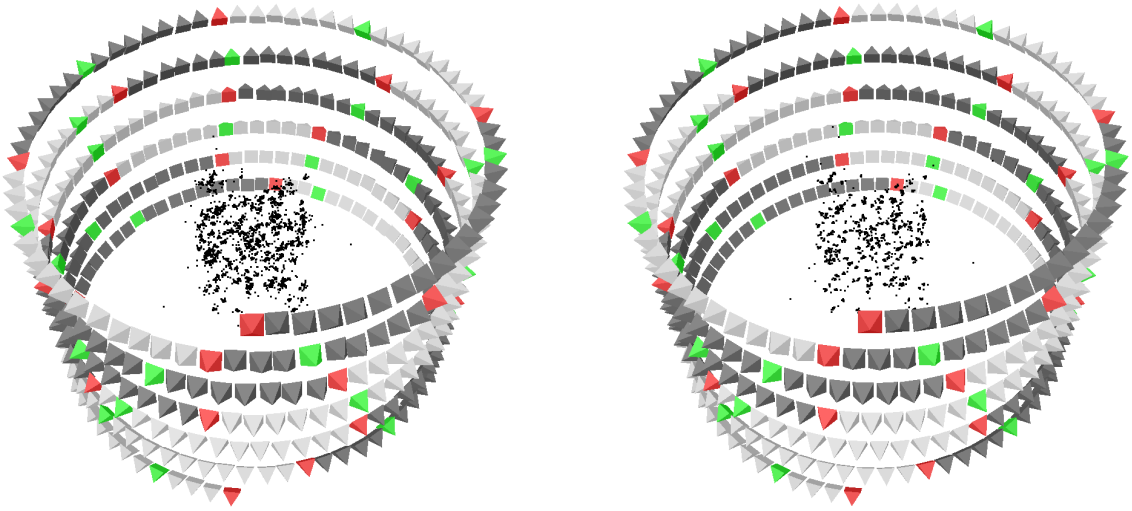


Figure 6.26: The two exemplary reconstructions of the test scene "spiral" correspond to the second and the fourth line of Table 6.12. The average relative pairwise translation errors of the two reconstructions are 0.96% and 0.23%, respectively.

loss probabilities instead. The results of the current experiment are specified in Table 6.12, whereas the results of the corresponding experiment with the test scene "slalom" are given in Table 6.10. The relative accuracy of the three M-estimators is consistent in both experiments. However, in the current experiment, the activation of any M-estimator roughly doubles the computation time of the standard bundle adjustment approach. Figure 6.26 contrasts two reconstructions of the test scene "spiral", which were optimized with the standard cost function and the Cauchy estimator, respectively. Due to the specified feature loss probability of $p_{\text{loss}} = 0.06$, there are approximately 25 reconstructed feature points for every unique feature position. Thus, the better accuracy of the Cauchy estimator is visualized by the much tighter clusters of feature points that share the same ground truth position.

6.2.3.5 Test Scene Wobble

The final two experiments in this subsection are based on the test scene "wobble", which is illustrated in Figure 6.18. In this test scene, the camera moves towards the feature points in a spiraling motion with a length of one unit and a diameter of 0.2 units. As explained in Subsection 4.1.1, this camera trajectory is disadvantageous to structure and motion estimation in general. Consequently, we use this test scene with 100 views and 100 unique feature points to evaluate the robustness of our estimation system to difficult scene configurations. In both experiments, a feature loss probability of $p_{\text{loss}} = 0.01$ results in the generation of approximately 200 feature trails with an average length of 50 frames. In addition to that, we simulate a moderate level of strong outliers with $p_{\text{out}} = 0.1$ and $\sigma_{\text{out}} = 10.0$.

code	σ_{inl}	$\epsilon_{\text{rbp}}^{\langle 50 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 05 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 50 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 95 \rangle}$	$\bar{\epsilon}_{\text{apr}}^{\langle 50 \rangle}$	$\bar{T}$	time
-|----|-	0.2	4.68	1.04%	1.71%	3.39%	0.2004°	4.00	463
-|----|-	0.5	4.98	2.61%	4.66%	7.65%	0.5514°	4.00	458
-|----|-	1.0	5.92	6.37%	9.39%	16.7%	1.0788°	4.00	459
-|----|-	2.0	8.63	13.1%	18.1%	37.4%	2.0938°	4.00	450
A|----|-	0.2	4.66	0.65%	1.21%	3.15%	0.1460°	3.97	1605
A|----|-	0.5	4.92	2.04%	4.20%	10.9%	0.4843°	4.76	1759
A|----|-	1.0	6.21	5.41%	10.3%	24.6%	1.2052°	5.31	1877
A|----|-	2.0	9.67	10.6%	21.9%	50.5%	2.5716°	5.86	1855

Table 6.13: The results in this table are based on the test scene "wobble" with 100 views and 100 unique feature points. The generation of the feature trails was performed with the parameter values $\sigma_{\text{out}} = 10.0$, $p_{\text{out}} = 0.1$, and $p_{\text{loss}} = 0.01$. The value of the parameter σ_{inl} is specified in the second column of the table. The last column shows the median value of the computation times in milliseconds.

In our first experiment with the test scene "wobble", we examine the performance of our key frame selection algorithm with respect to the noise level in the coordinates of the feature positions in the feature trails. The results of this experiment are listed in Table 6.13. The computation of the view ray angles improves the median error values for $\sigma_{\text{inl}} = 0.2$ and $\sigma_{\text{inl}} = 0.5$, but degrades the median error values for higher levels of noise. In addition to that, the average relative pairwise translation error exhibits a higher variation for all levels of noise. As described in Subsection 4.3.4, the median of the view ray angles is computed with a robust version of the five point algorithm, which yields up to ten putative solutions of the relative camera pose. Even though we disambiguate these solutions using an additional view, which resolves all ambiguities in theory, a combination of strong noise in the feature positions and adverse scene geometry can lead to the selection of the wrong solution in practice. When the correct view ray angles of all tentative key frames are very small, an incorrect reconstruction results in the largest angle, and thus the highest quality measure, with a very high probability. Consequently, the random occurrence of a single reconstruction error possibly distorts the results of our key frame selection algorithm. As the probability of these errors increases with the level of noise in the feature positions, the average number of the created segments in Table 6.13 also increases with the level of noise.

In our second experiment with the test scene "wobble", we further increase the difficulty of the reconstruction by reducing the depth of the cuboid that contains the feature points to 0.1 units. The experimental results in Table 6.14 illustrate that the quality of the reconstruction is considerably impaired by the reduction of the depth of the scene structure. When the scene geometry is planar and one camera position is closer to all feature points than the other camera position, there are two physically valid solutions to the relative camera pose problem. However, [Seg07] reports that

code	depth	$\epsilon_{\text{rbp}}^{\langle 50 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 05 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 50 \rangle}$	$\bar{\epsilon}_{\text{rpt}}^{\langle 95 \rangle}$	$\bar{\epsilon}_{\text{apr}}^{\langle 50 \rangle}$	$\bar{T}$	time
-|----|-	1.0	5.92	6.37%	9.39%	16.7%	1.0788°	4.00	457
-|----|-	0.1	6.27	32.5%	39.2%	45.7%	4.6264°	4.00	483
-|B---|C	1.0	4.90	1.82%	2.65%	3.80%	0.3039°	4.00	1070
-|B---|C	0.1	4.96	9.92%	13.5%	25.6%	1.5878°	4.00	1190
-|--B-|C	1.0	4.81	0.86%	0.96%	1.15%	0.0954°	4.00	2208
-|--B-|C	0.1	4.81	3.36%	3.66%	4.08%	0.3404°	4.00	2560
-|--BB|C	1.0	4.81	0.83%	0.91%	0.98%	0.0895°	4.00	5704
-|--BB|C	0.1	4.81	3.34%	3.60%	3.96%	0.3323°	4.00	6004

Table 6.14: The results in this table are based on the test scene "wobble" with 100 views and 100 unique feature points. The depth of the cuboid containing the feature points is shown in the second column of the table. The feature trails were generated with the parameter values $\sigma_{\text{inl}} = 1.0$, $\sigma_{\text{out}} = 10.0$, $p_{\text{out}} = 0.1$, and $p_{\text{loss}} = 0.01$.

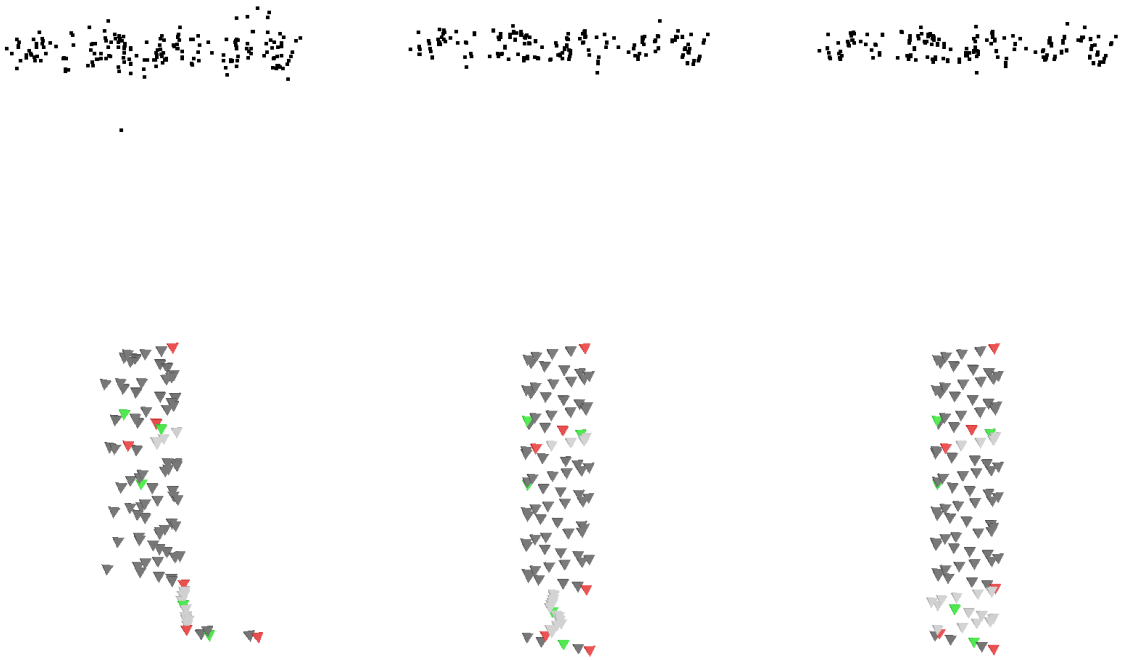


Figure 6.27: This figure shows three reconstructions of the test scene "wobble". The left and the right image of this figure correspond to the second and the sixth line of Table 6.14, respectively. Thus, the reconstruction in the right image is based on the same reconstructed segments as the one in the left image, but illustrates the effects of the optimization of the merged segments with our bundle adjustment approach. The middle image depicts an intermediate step of the optimization, in which the number of bundle adjustment iterations was limited to eight. The average relative pairwise translation errors of the three reconstructions are 30.3%, 10.2%, and 4.06%, respectively. The camera moves towards the feature points in this test scene.

the accuracy of the five-point algorithm is already affected when these conditions are only approximately satisfied. Consequently, even the disambiguation of the solutions of the five-point algorithm with an additional view is prone to return the wrong solution when the noise in the feature positions is too high. The second segment of the reconstruction in the left image of Figure 6.27 is an example for this problem. The other two images of this figure demonstrate the robustness of our bundle adjustment approach, which manages to improve the initial reconstruction significantly.

6.2.4 Summary

The experimental evaluation of our structure and motion estimation system in this section is based on artificial test scenes. Our approach for generating these test scenes is detailed in Subsection 6.2.1. One test scene is used to evaluate our proposed extension of the POSIT algorithm in Subsection 6.2.2. Compared to the standard POSIT algorithm, our POSIT algorithm with virtual reference point exhibits the same computational efficiency and a considerably higher accuracy. Finally, we use four test scenes to evaluate the performance of our structure and motion estimation system in Subsection 6.2.3. The following results are especially noteworthy:

- Our key frame selection algorithm improves the accuracy of the reconstruction, as long as the input data allows a successful estimation of the relative camera poses with the five-point algorithm.

- Our approach for the merging of the segments works reliably. It was tested with reconstructions that consist of more than 20 segments.

- The implemented robust estimation techniques enable our estimation system to successfully cope with outlier percentages of up to 40%.

- Our estimation system allows the independent activation of four different applications of our bundle adjustment approach. This feature provides the flexibility to adapt the balance between the accuracy and the computational efficiency of our estimation system to specific requirements.

- Activating the bundle adjustment optimization of the merged segments in combination with the Cauchy estimator results in a very high level of accuracy and robustness. With this configuration, our estimation system demonstrates its computational efficiency by performing the reconstructions with median computation times that equal a computation rate of more than twelve views per second in all conducted experiments.

We complete the evaluation of our structure and motion estimation system with the additional experiments in Section 6.4.

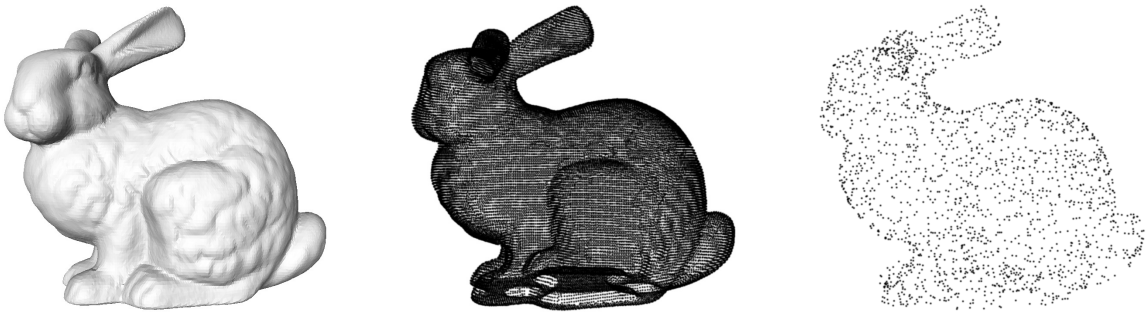


Figure 6.28: The left image visualizes the point set "bunny" with the help of a filled polygon mesh. The middle image shows the original version of the point set, while the right image depicts a subsampled version with 2000 points. For our experiments, we scale the point set "bunny" to fit into a sphere with a diameter of 100 units.

6.3 Point Set Registration

In this section, we describe the experimental evaluation of our work on point set registration, which is presented in Chapter 5. In particular, the structure of our variant of the ICP algorithm is illustrated in Figure 5.3 and the default values of its user-specified parameters are listed in Table 5.1. The first subsection of this section contains an explanation of our experimental setup, which covers the input point sets, the generation of the test data, and the considered error measures. The second subsection details the experimental evaluation of our variant of the standard ICP algorithm. In addition to that, we evaluate the implemented extensions for robust correspondence estimation and integrated scale estimation in Subsection 6.3.3 and Subsection 6.3.4, respectively.

6.3.1 Experimental Setup

We use two different point sets for the evaluation of the registration algorithms in this section. Both point sets are scaled to fit into a sphere with a diameter of 100 units. Our first point set is widely known as the "Stanford Bunny". It is available from The Stanford 3D Scanning Repository [Sta11]. The generation of this point set is detailed in [Tur94]. The original version of the point set "bunny" consists of 35947 points. We use randomly subsampled versions of this point set. Two exemplary versions of the point set "bunny" are illustrated in the middle image and the right image of Figure 6.28. The shape of this point set is particularly suitable for point set registration. Thus, we expect the corresponding experimental results to demonstrate the upper limit of the performance of the registration algorithms.

Our second point set is illustrated in Figure 6.29. The point set "bench" was reconstructed from an image sequence of a cluttered work bench with a structure and motion estimation algorithm. As a consequence, this input point set is representative of many reconstructions computed with our structure and motion estimation

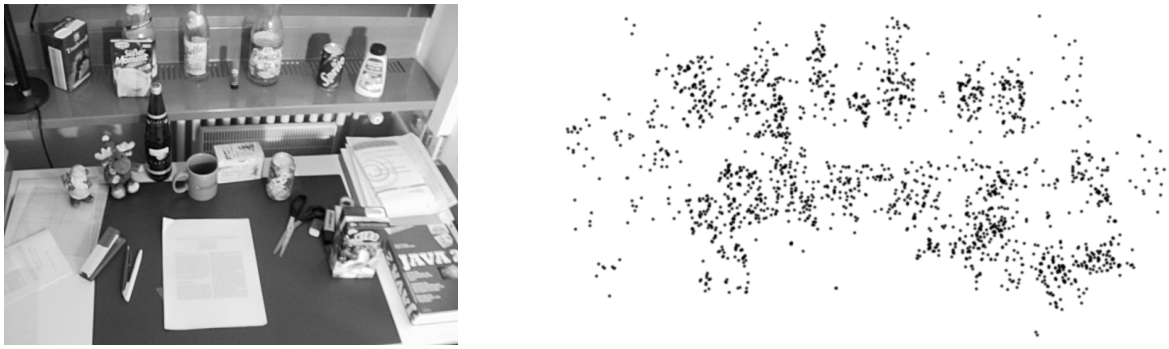


Figure 6.29: One image of the image sequence used for the reconstruction of the point set "bench" is shown on the left. The right image illustrates a subsampled version of the point set "bench". This version contains 2000 points. The point set "bench" is also scaled to fit into a sphere with a diameter of 100 units.

system, which is described in Chapter 4. The original version of this point set consists of 3584 points, but we exclusively use a subsampled version with 2000 points. Due to the irregular shape of the point set "bench", we expect the registration of this point set to be very demanding.

As the evaluated registration algorithms require both a data point set and a model point set as input, we use an automated approach for the generation of the artificial test data from the point sets described in the preceding paragraphs. In the first step of this approach, the input point set is moved to the origin of the coordinate system and resized to fit into a sphere with a diameter of 100 units. The resulting point set is used as the basis for both the data point set $\mathcal{A}$ and the model point set $\mathcal{B}$. The second step is concerned with the simulation of a partial overlap of the two point sets. To this end, a specified percentage p_{cut} of the points is deleted from both point sets. In particular, the outmost points with respect to a random direction are deleted from the model point set, and the outmost points with respect to the opposite direction are deleted from the data point set. After the deletion of the points, the relative percentage of inliers in the data point set is given by

$$p_{\text{overlap}} = \frac{1 - 2p_{\text{cut}}}{1 - p_{\text{cut}}} \,. \tag{6.8}$$

The described approach ensures that no point is deleted in both point sets, as long as $p_{\text{cut}} \leq 0.5$. Furthermore, compared to the deletion of randomly selected points, it results in a considerably more difficult registration problem. This step concludes the generation of the model point set.

In the third step, we add Gaussian noise to the coordinates of the data point positions. With a probability of p_{out}, we apply Gaussian noise with a large standard deviation of σ_{out} to simulate strong outliers. For the remaining points, we use a smaller standard deviation of σ_{inl}. The fourth and final step consists of generating and applying the initial motion to the data point set. To this end, the rotation

position	parameter	possible values	reference
XX| - | -	point set	BU (bunny) / BE (bench)	Subs. 6.3.1
- |XXX| -	δ_{type}	- / LFS / LMS (LMedS) / LTS	Section 5.3
- | - |X	δ_{scale}	- / S	Section 5.4

Table 6.15: The configuration code of the experimental evaluation of our registration algorithms displays the name of the point set, as well as the values of two important user-specified parameters of our variant of the ICP algorithm. The extensions for robust correspondence estimation are detailed in Section 5.3. They can be activated with the user-specified parameter δ_{type}. The integrated scale estimation, which is presented in Section 5.4, can be deactivated (-) or activated (S) with the parameter δ_{scale}.

angle θ_{angle}, the translation vector length θ_{trans}, and the scale factor θ_{scale} have to be specified. Our automated approach randomly determines a rotation axis and a translation direction for every test run. The combination of these values yields the motion parameters $(\hat{\boldsymbol{R}}, \hat{\boldsymbol{t}}, \hat{s})$, where $\hat{s} = \theta_{\text{scale}}$. We apply the motion parameters to the data points as follows

$$\hat{\boldsymbol{a}}_i = \frac{1}{\hat{s}} \hat{\boldsymbol{R}}^{\mathrm{T}} \boldsymbol{a}_i - \hat{\boldsymbol{R}}^{\mathrm{T}} \hat{\boldsymbol{t}} \,, \quad \forall \boldsymbol{a}_i \in \mathcal{A} \,. \tag{6.9}$$

As a result, the estimated motion $(\tilde{\boldsymbol{R}}, \tilde{\boldsymbol{t}}, \tilde{s})$ has to be compared to the ground truth motion $(\hat{\boldsymbol{R}}, \hat{s}\hat{\boldsymbol{t}}, \hat{s})$. This rather intricate approach ensures that the actual translation of the data point set has a length of θ_{trans} when $\theta_{\text{scale}} \neq 1$.

We evaluate the estimated motion $(\tilde{\boldsymbol{R}}, \tilde{\boldsymbol{t}}, \tilde{s})$ with three error measures. When the corresponding ground truth motion is given by $(\hat{\boldsymbol{R}}, \hat{s}\hat{\boldsymbol{t}}, \hat{s})$, the rotation error ϵ_{rer} is defined as the angle of the rotation matrix

$$\Delta \boldsymbol{R} = \hat{\boldsymbol{R}} \tilde{\boldsymbol{R}}^{\mathrm{T}}. \tag{6.10}$$

Furthermore, the translation error ϵ_{ter} is specified as

$$\epsilon_{\text{ter}} = \| \hat{s}\hat{\boldsymbol{t}} - \tilde{\boldsymbol{t}} \|_2 \,. \tag{6.11}$$

Finally, the scale error ϵ_{ser} is defined as

$$\epsilon_{\text{ser}} = \begin{cases} \hat{s}/\tilde{s} & : \quad \hat{s} \geq \tilde{s} \\ \tilde{s}/\hat{s} & : \quad \hat{s} < \tilde{s} \end{cases} , \tag{6.12}$$

so that ϵ_{ser} is always greater than or equal to 1.0.

In order to simplify the presentation of the experimental results in the subsequent subsections, we specify the most important aspects of the experimental setup with the configuration code detailed in Table 6.15. In addition to that, the experimental setup is influenced by the parameters σ_{inl}, σ_{out}, and p_{out}, which specify the amount

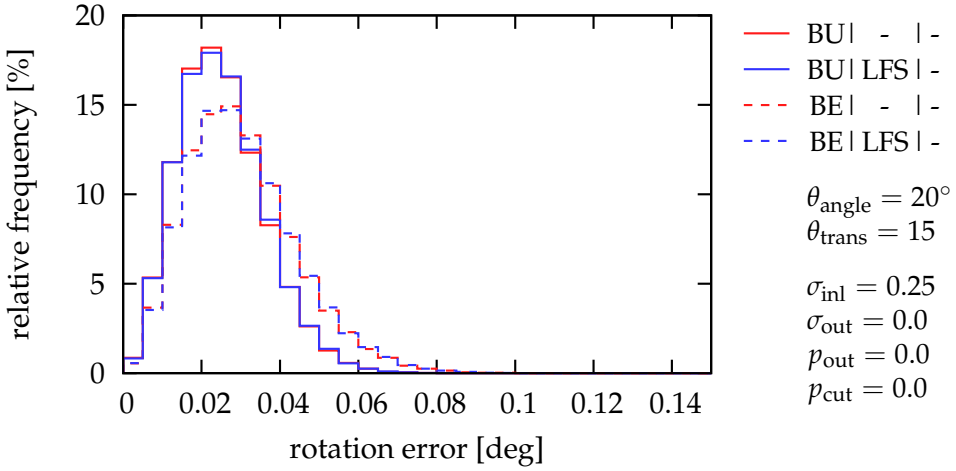


Figure 6.30: The accuracy of two configurations of our variant of the ICP algorithm is illustrated with a histogram of the rotation error for two point sets. The results were obtained by performing 50000 test runs for every configuration.

of noise in the coordinates of the data point positions. Furthermore, the parameter p_{cut} determines the degree of overlap of the data point set and the model point set. Finally, the rotation angle θ_{angle}, the translation vector length θ_{trans}, and the scale factor θ_{scale} define the initial motion between the two point sets. Unless explicitly stated otherwise, the size of the input point sets is 2000 points in all experiments of the following subsections.

6.3.2 Evaluation of the Standard ICP Algorithm

The integrated scale estimation of our variant of the ICP algorithm is not used in any experiment of this subsection. Consequently, the scale factor of the initial motion is set to $\theta_{scale} = 1$ in these experiments. The values of the remaining parameters are presented directly in the graphs that illustrate the experimental results. In this subsection, we determine the accuracy of the algorithm with the help of histograms of the relevant error measures. Furthermore, we analyze the basin of convergence of the algorithm with respect to the rotation angle θ_{angle} and the translation vector length θ_{trans} by calculating the percentage of successful registrations.

In our first experiment, we evaluate the accuracy of our registration algorithm with the point sets “bunny” and “bench”. The results of this experiment are illustrated in Figure 6.30 and Figure 6.31. The values of the parameters θ_{angle} and θ_{trans} were selected to ensure that the evaluated algorithm converges to the correct solution. Furthermore, we perturb the coordinates of the data point positions with Gaussian noise with a standard deviation of $\sigma_{inl} = 0.25$. In this experiment, we evaluate the accuracy of our variant of the ICP algorithm both with and without the LFS extension for robust correspondence estimation. The histograms in both

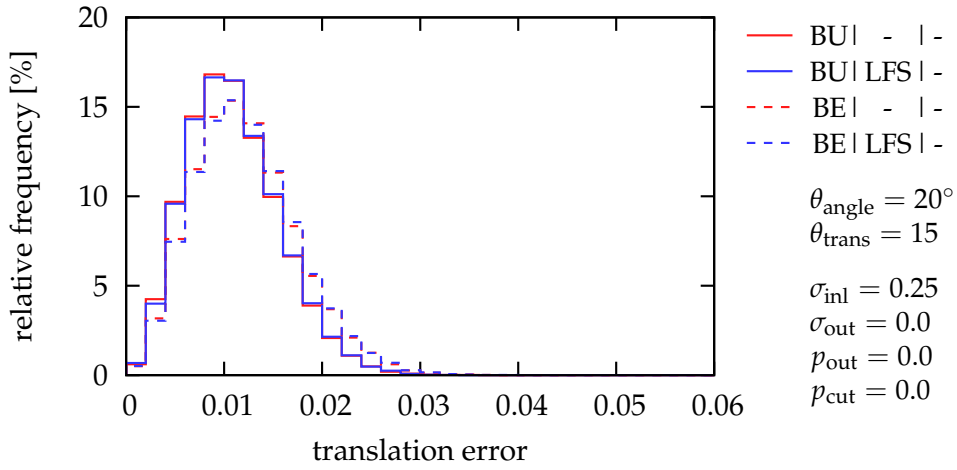


Figure 6.31: The accuracy of two configurations of our variant of the ICP algorithm is illustrated with a histogram of the translation error for two point sets. The results were obtained by performing 50000 test runs for every configuration.

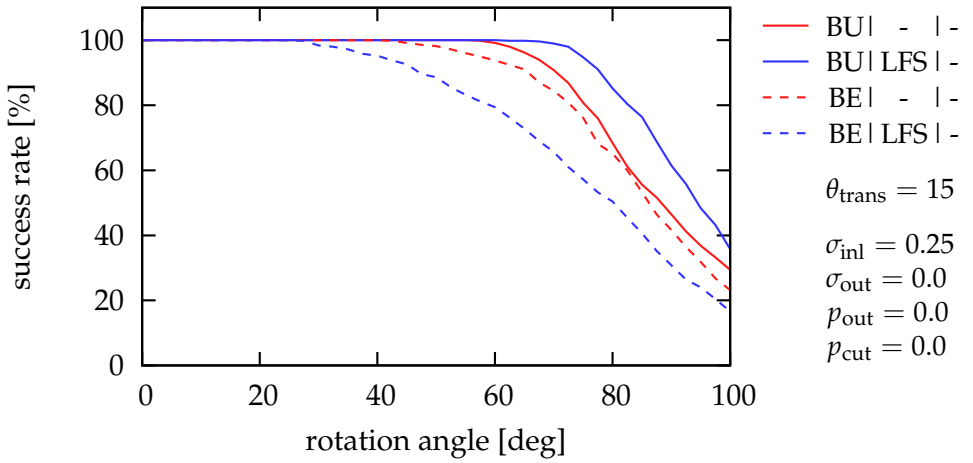


Figure 6.32: This graph visualizes the basin of convergence of two configurations of our variant of the ICP algorithm with respect to the rotation angle of the initial motion for the point sets “bunny” and “bench”. Every value of the success rate in this graph was determined by performing 500 test runs.

figures show that the activation of the LFS extension results in a negligible decrease of the accuracy of the algorithm. In contrast to this, the shape of the input point set has a more pronounced influence on the achieved accuracy, especially with respect to the rotation error.

In the next two experiments, we analyze the basin of convergence of our variant of the ICP algorithm. To this end, we consider the percentage of successful registrations with respect to the rotation angle and the translation vector length of the

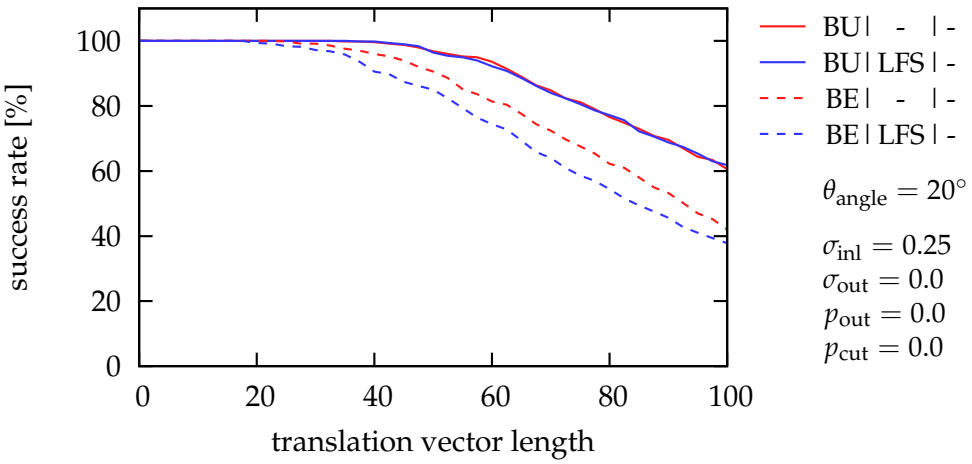


Figure 6.33: This graph visualizes the basin of convergence of two configurations of our variant of the ICP algorithm with respect to the translation vector length of the initial motion for the point sets "bunny" and "bench". Every value of the success rate in this graph was determined by performing 500 test runs.

initial motion. We identify successful registrations by verifying that the values of the error measures ϵ_{rer} and ϵ_{ter} both lie below specified thresholds. For the specification of these thresholds, we take the histograms in Figure 6.30 and Figure 6.31 into account. In order to include all successful registrations, we set the thresholds to $0.1°$ for the rotation error and to 0.04 for the translation error. With these settings, we obtain the results presented in Figure 6.32 and Figure 6.33. In both experiments, the evaluations with the point set "bench" result in a noticeably smaller basin of convergence. For this point set, the activation of the LFS extension reduces the basin of convergence even further. In contrast to this, the LFS extensions yields a larger basin of convergence with respect to the rotation angle of the initial motion for the point set "bunny".

6.3.3 Evaluation of Robust Correspondence Estimation

For the evaluation of the implemented extensions for robust correspondence estimation, we simulate outliers and partially overlapping point sets with the parameter values $\sigma_{\text{inl}} = 0.25$, $\sigma_{\text{out}} = 2.5$, $p_{\text{out}} = 0.1$, and $p_{\text{cut}} = 0.1$. Compared to the experiments in the preceding subsection, we also reduce the translation vector length to $\theta_{\text{trans}} = 7.5$. Figure 6.34 shows an exemplary setup with partially overlapping point sets before and after the registration of the data point set and the model point set.

As a first step, we analyze the accuracy of our variant of the ICP algorithm with the implemented extensions. The results of the conducted experiment are presented in Figure 6.35 and Figure 6.36. The accuracy of the LFS extension and the LTS extension is almost identical. In contrast to this, the LMedS extension is less accurate with

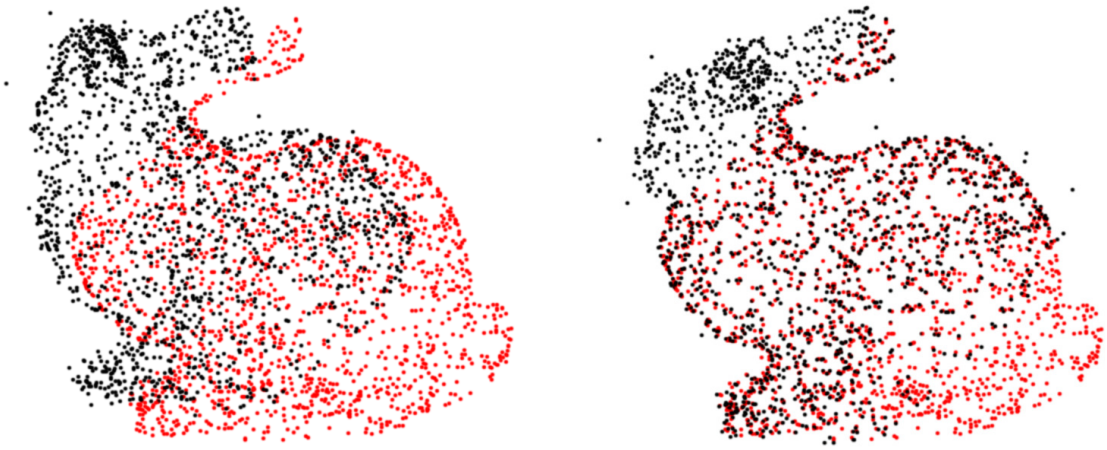


Figure 6.34: The left image illustrates the initial motion between the data point set, which is represented by black points, and the model point set for a rotation angle of $\theta_{\text{angle}} = 20°$ and a translation vector length of $\theta_{\text{trans}} = 7.5$. The data point set was generated with the parameter values $\sigma_{\text{inl}} = 0.25$, $\sigma_{\text{out}} = 2.5$, $p_{\text{out}} = 0.1$, and $p_{\text{cut}} = 0.2$. The registration of the data point set and the model point set in the right image was computed with our variant of the ICP algorithm using the LFS extension.

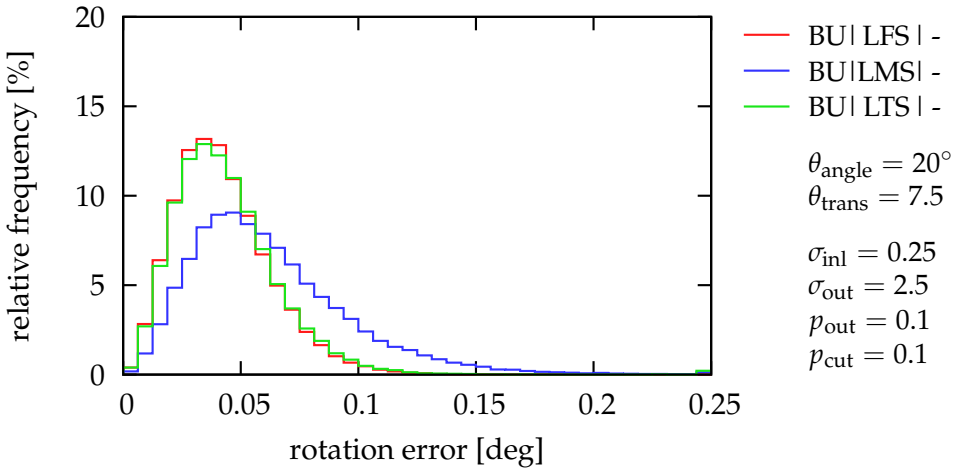


Figure 6.35: The accuracy of the implemented extensions for robust correspondence estimation is illustrated with a histogram of the rotation error for the point set "bunny". The results were obtained by performing 50000 test runs for every configuration of our variant of the ICP algorithm.

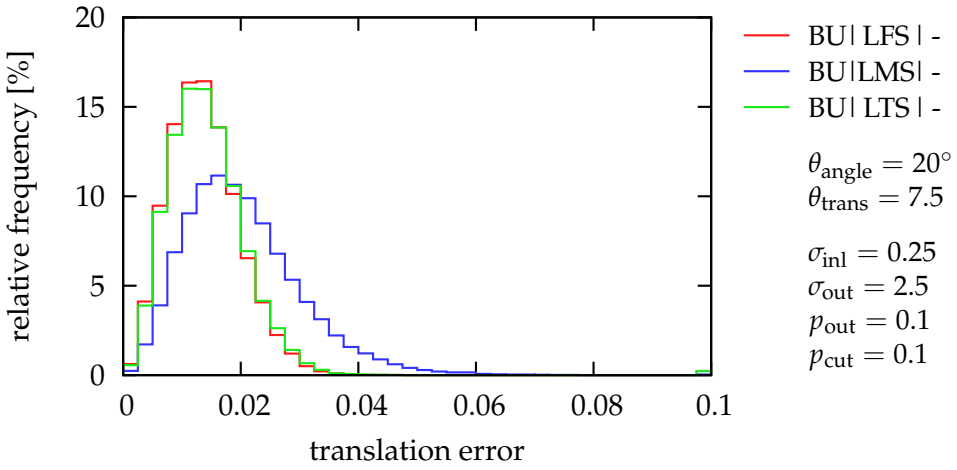


Figure 6.36: The accuracy of the implemented extensions for robust correspondence estimation is shown with a histogram of the translation error for the point set "bunny". The results were obtained by performing 50000 test runs for every configuration of our variant of the ICP algorithm.

respect to both the rotation error and the translation error. As in the previous subsection, we consider the histograms in both figures to determine the thresholds for identifying successful registrations in the subsequent experiments. Consequently, we set the thresholds to 0.2° for the rotation error and to 0.08 for the translation error.

In order to determine suitable values for the exponents θ_{elfs} and θ_{elts} of the objective functions of the LFS extension and the LTS extension, we evaluate the basin of convergence of the extensions for different values of the respective exponent. The results of the performed experiments are illustrated in Figure 6.37 and Figure 6.38. The basin of convergence of the LTS extension is similar for all tested values of θ_{elts}. As a consequence, we adopt a default value of $\theta_{\text{elts}} = 4$, which has already been proposed in the original work [Che05]. In contrast to this, the basin of convergence of the LFS extension strongly depends on the value of the exponent θ_{elfs}. As the smallest evaluated value of $\theta_{\text{elfs}} = 3$ rejects the highest fraction of corresponding point pairs as outliers, it yields the best basin of convergence with respect to the parameter p_{cut} in Figure 6.38. For the same reason, this value results in the smallest basin of convergence with respect to the rotation angle of the initial motion in Figure 6.37. Our default value of $\theta_{\text{elfs}} = 5$ balances the performance of the LFS extension in both experiments.

In the three remaining experiments of this subsection, we compare the basin of convergence of the three extensions for robust correspondence estimation with respect to different parameters. Figure 6.39 illustrates that the basin of convergence with respect to the rotation angle of the initial motion is very similar for all three extensions. The results for the translation vector length of the initial motion in Fig-

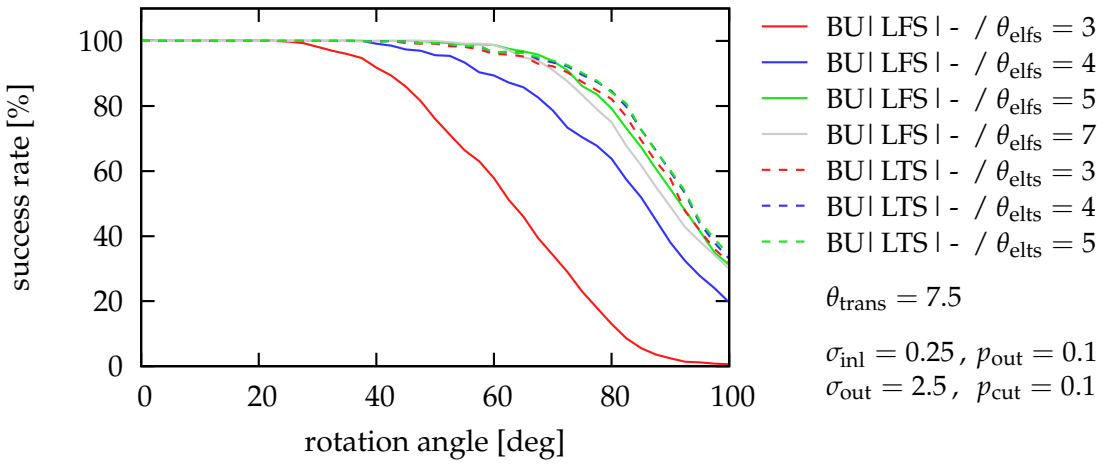


Figure 6.37: This graph visualizes the basin of convergence of our variant of the ICP algorithm with the LFS extension and the LTS extension with respect to the rotation angle of the initial motion. For both extensions, several values of the exponent in the respective objective function are evaluated. Every value of the success rate in this graph was determined with 500 test runs.

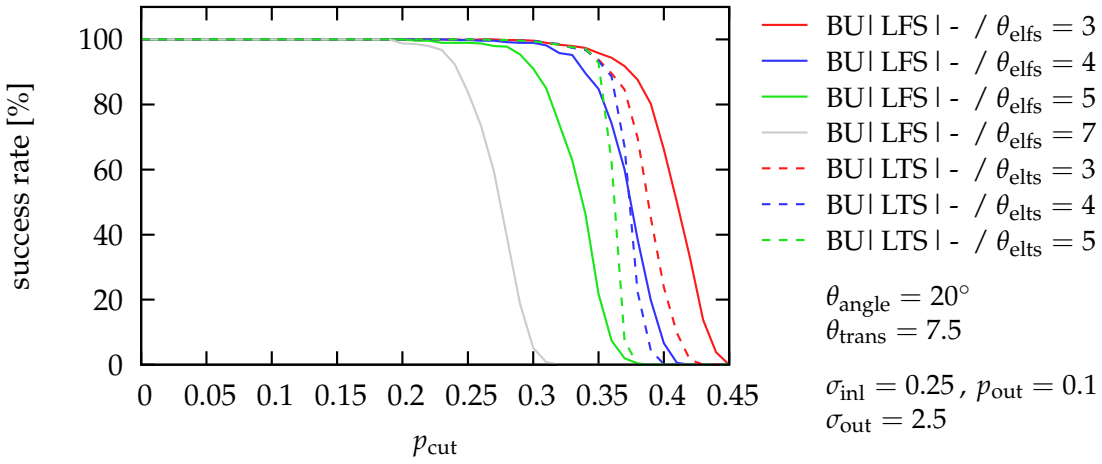


Figure 6.38: This graph presents the basin of convergence of our variant of the ICP algorithm with the LFS extension and the LTS extension with respect to the value of the parameter p_{cut}. For both extensions, several values of the exponent in the respective objective function are evaluated. Every value of the success rate in this graph was determined with 500 test runs.

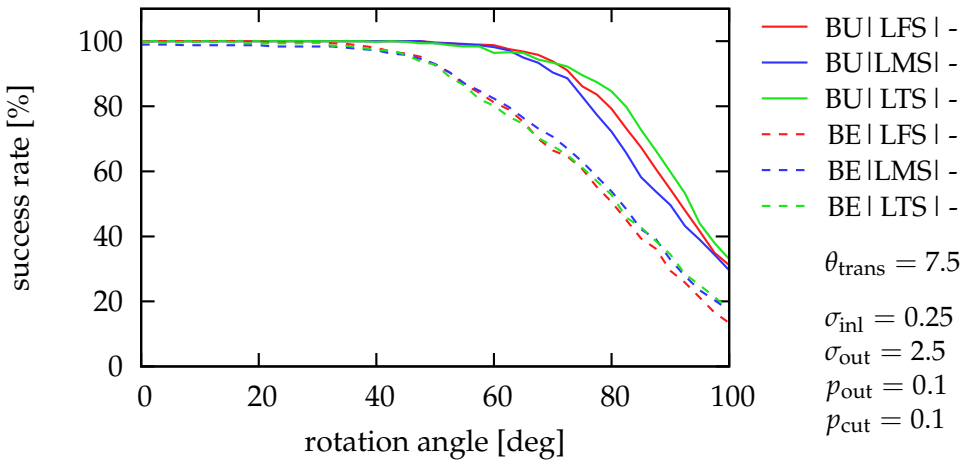


Figure 6.39: This graph illustrates the basin of convergence of the implemented extensions for robust correspondence estimation with respect to the rotation angle of the initial motion for the point sets "bunny" and "bench". Every value of the success rate in this graph was determined by performing 500 test runs.

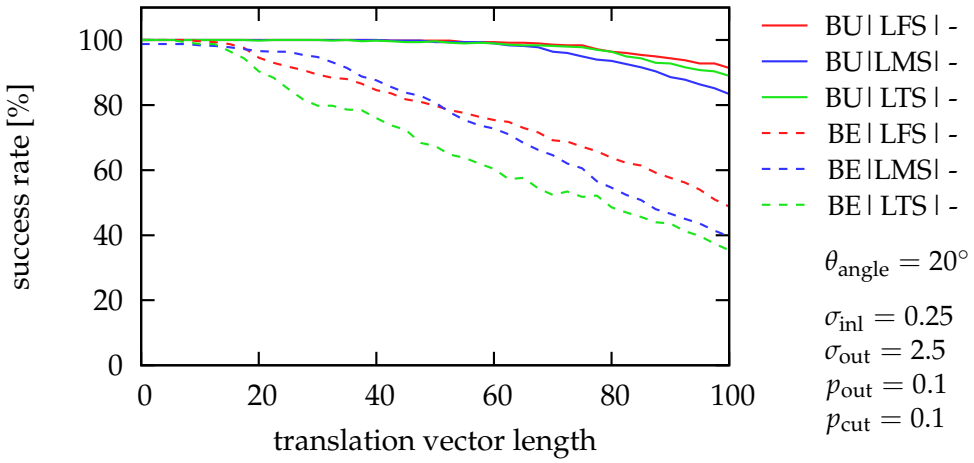


Figure 6.40: This graph depicts the basin of convergence of the implemented extensions for robust correspondence estimation with respect to the translation vector length of the initial motion for the point sets "bunny" and "bench". Every value of the success rate in this graph was determined by performing 500 test runs.

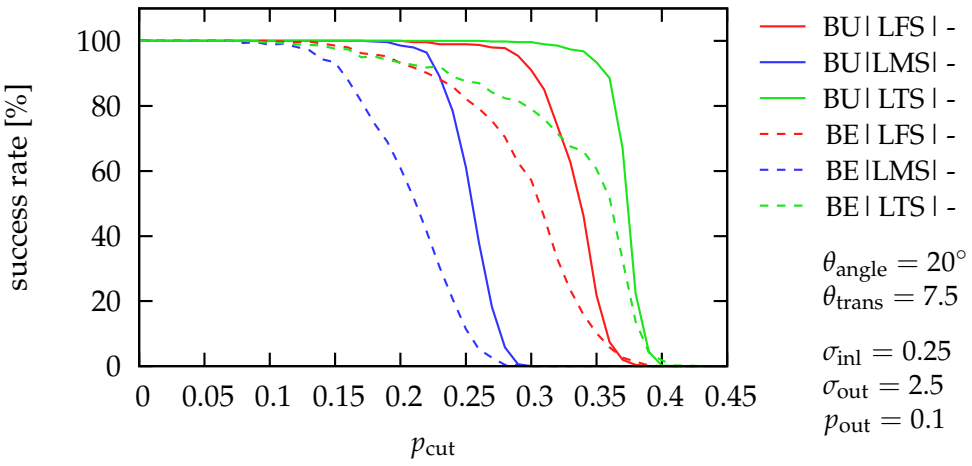


Figure 6.41: This graph visualizes the basin of convergence of the implemented extensions for robust correspondence estimation with respect to the value of the parameter p_{cut} for the point sets "bunny" and "bench". Every value of the success rate in this graph was determined by performing 500 test runs.

ure 6.39 show that the basin of convergence of the LTS extension is smaller than that of the other extensions for the point set "bench". However, in both experiments, the shape of the input point set has a larger influence on the basin of convergence of our registration algorithm than the choice of the extension for robust correspondence estimation. Finally, Figure 6.41 presents the basin of convergence of the extensions with respect to the fraction of systematic outliers in the data point set and the model point set. Evidently, the LMedS extension is least capable of dealing with large numbers of systematic outliers. For our default values of θ_{elfs} and θ_{elfs}, the LTS extensions achieves the largest basin of convergence in this experiment. Again, the shape of the input point set has a very strong influence on the basin of convergence of all evaluated extensions.

6.3.4 Evaluation of Integrated Scale Estimation

Our proposed extension for integrated scale estimation is described in detail in Section 5.4. We evaluate its accuracy both with and without the LFS extension for robust correspondence estimation in the first experiment of this subsection. To this end, we generate the data point set with the parameter values $\sigma_{\text{out}} = 0.0$, $p_{\text{out}} = 0.0$, and $p_{\text{cut}} = 0.0$ for the configuration without the LFS extension. In contrast to this, we use the values $\sigma_{\text{out}} = 2.5$, $p_{\text{out}} = 0.1$, and $p_{\text{cut}} = 0.1$ for the configuration with the LFS extension. The results of this experiment are shown in Figure 6.42. Expectedly, the accuracy of the estimated scale factor is reduced by the outliers that are generated for the configuration with the LFS extension. In addition to that, the results for the point set "bunny" and the point set "bench" deviate by a perceptible

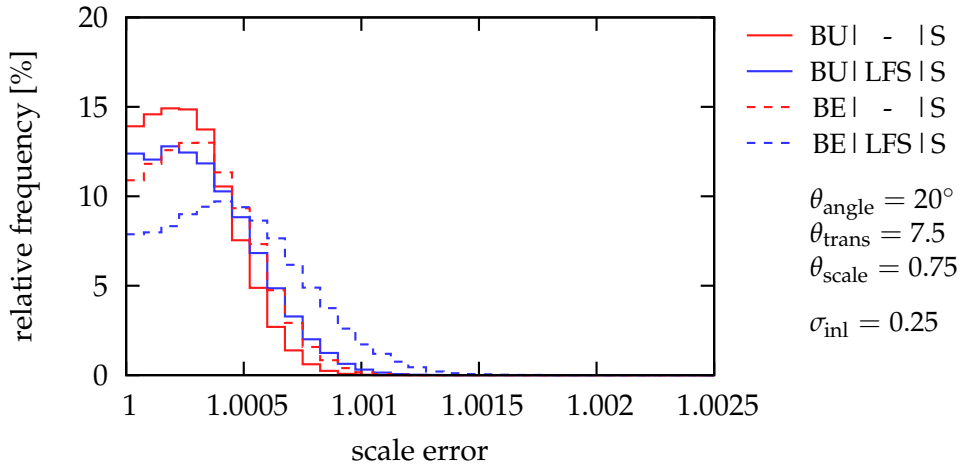


Figure 6.42: The accuracy of our variant of the ICP algorithm with integrated scale estimation is illustrated with a histogram of the scale error for the point sets "bunny" and "bench". The results were obtained by performing 50000 test runs for every configuration.

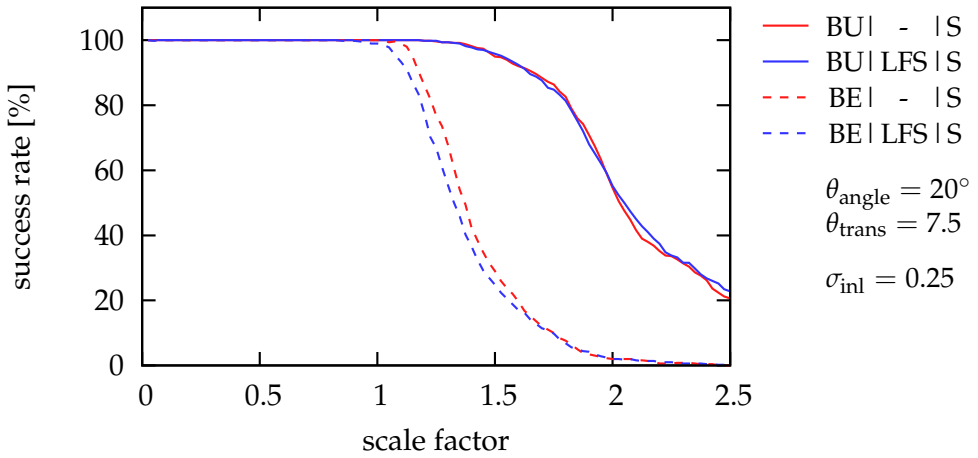


Figure 6.43: This graph visualizes the basin of convergence of our variant of the ICP algorithm with integrated scale estimation with respect to the scale factor of the initial motion for the point sets "bunny" and "bench". Every value of the success rate in this graph was determined by performing 500 test runs.

margin. For the subsequent evaluation of the basin of convergence of the integrated scale estimation, we determine successful registrations with a threshold of 0.2° for the rotation error, 0.08 for the translation error, and 1.002 for the scale error.

In the experiment illustrated in Figure 6.43, we evaluate the basin of convergence of our variant of the ICP algorithm with the extension for integrated scale estimation with respect to the scale factor of the initial motion. This experiment uses the same parameter values for the generation of the data point sets as the preceding

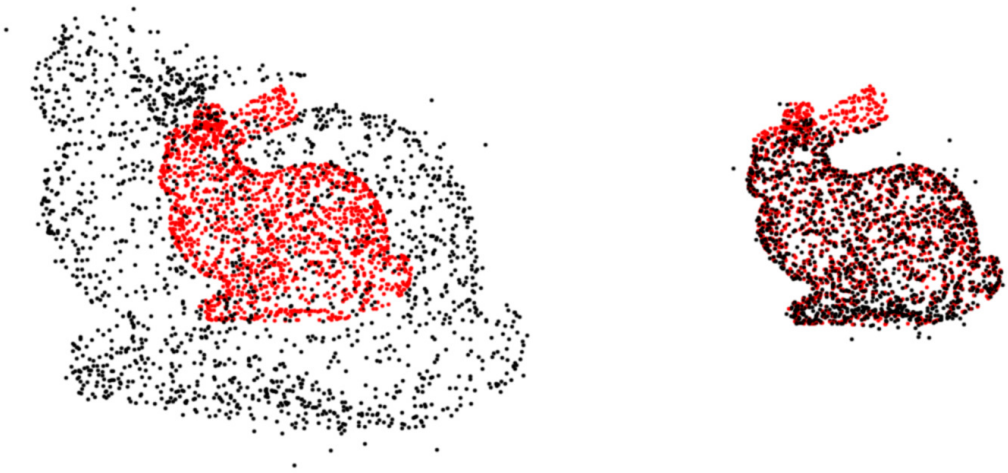


Figure 6.44: The left image visualizes the initial motion between the data point set, which is represented by black points, and the model point set for values of $\theta_{\text{angle}} = 20°$, $\theta_{\text{trans}} = 7.5$, and $\theta_{\text{scale}} = 0.5$. The data point set was generated with the parameter values $\sigma_{\text{inl}} = 0.25$, $\sigma_{\text{out}} = 2.5$, $p_{\text{out}} = 0.1$, and $p_{\text{cut}} = 0.1$. The registration in the right image was computed with our variant of the ICP algorithm using the LFS extension and the integrated scale estimation.

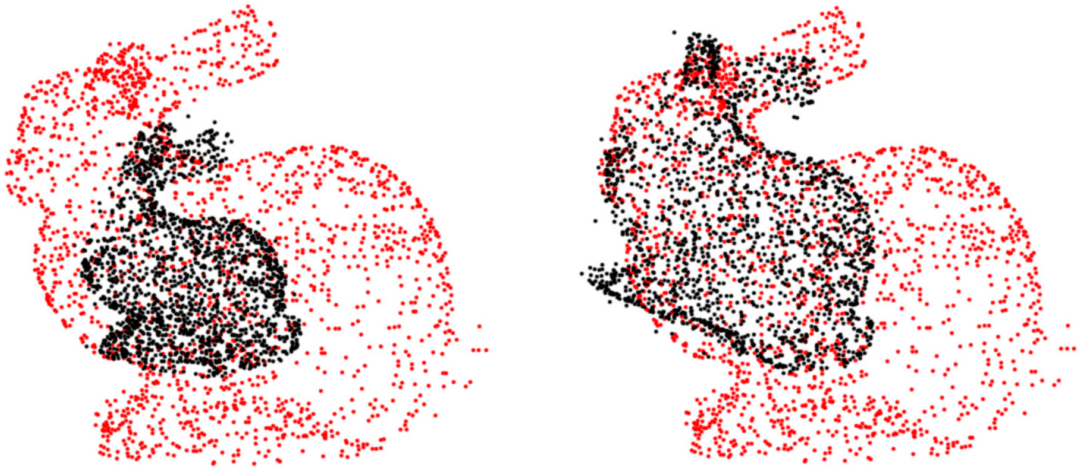


Figure 6.45: The left image illustrates the initial motion between the data point set, which is represented by black points, and the model point set for values of $\theta_{\text{angle}} = 20°$, $\theta_{\text{trans}} = 7.5$, and $\theta_{\text{scale}} = 2.0$. The data point set was generated with the parameter values $\sigma_{\text{inl}} = 0.25$, $\sigma_{\text{out}} = 2.5$, $p_{\text{out}} = 0.1$, and $p_{\text{cut}} = 0.1$. The registration in the right image was computed with our variant of the ICP algorithm using the LFS extension and the integrated scale estimation.

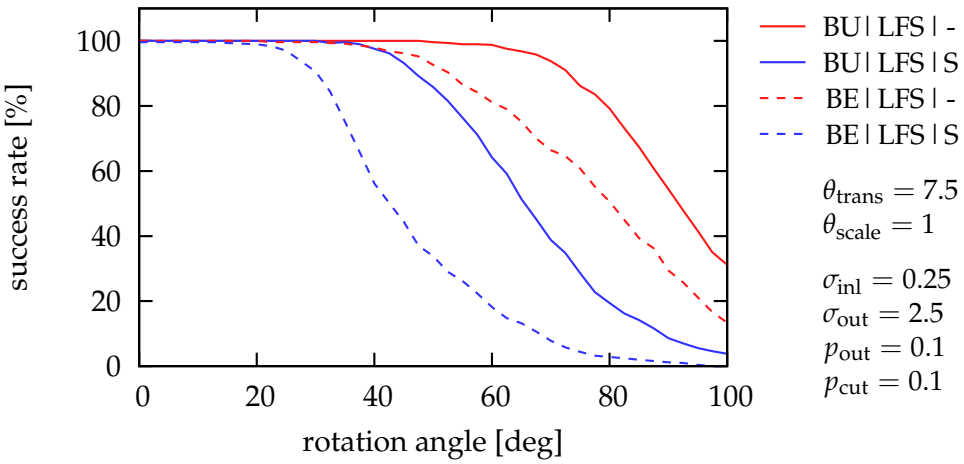


Figure 6.46: This graph illustrates the effect of the integrated scale estimation on the basin of convergence of our variant of the ICP algorithm with respect to the rotation angle of the initial motion for the point sets "bunny" and "bench". Every value of the success rate in this graph was determined by performing 500 test runs.

experiment. First of all, the activation of the LFS extension does not result in any significant change of the basin of convergence. What is more, the graphs in Figure 6.43 clearly show that the success rate of the registration is very high as long as the scale factor is below one, which means that the scale of the data point set is larger than that of the model point set. As the corresponding point pairs are generated by finding the closest model point for every data point in our variant of the ICP algorithm, the scale of the data point set is quickly reduced in the first iterations of the algorithm. This configuration is illustrated in Figure 6.44. In contrast to this, the success rate rapidly diminishes when the relative scale of the model point set increases. In this case, it is possible that only a small fraction of the model points is assigned to the corresponding point pairs, so that the scale of the data point set is not increased adequately. As a result, the data point set gets stuck inside the model point set. An example of the described configuration is presented in Figure 6.45.

In the next two experiments, we examine the effect of the scale estimation on the basin of convergence of our variant of the ICP algorithm with respect to the rotation angle and the translation vector length of the initial motion. The results of these experiments are presented in Figure 6.46 and Figure 6.47, respectively. The basin of convergence of the ICP algorithm with respect to the rotation angle is considerably reduced by the activation of the scale estimation. However, the activation of the scale estimation reduces the basin of convergence with respect to the translation vector length even more dramatically. All data points that are translated beyond the edge of the model point set result in corresponding point pairs with one of the points on the edge of the model point set. Consequently, the scale of the data point set is reduced in the first iteration of the ICP algorithm to reflect the reduced spread

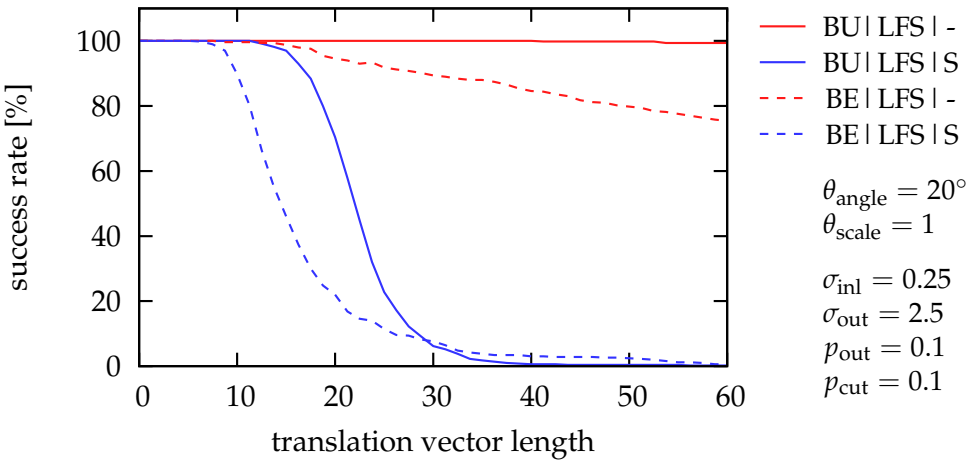


Figure 6.47: This graph demonstrates the effect of the integrated scale estimation on the basin of convergence of our variant of the ICP algorithm with respect to the translation vector length of the initial motion for the point sets "bunny" and "bench". Every value of the success rate in this graph was determined by performing 500 test runs.

size	250		1000		4000		16000	
code	iter	time	iter	time	iter	time	iter	time
BU | LFS | S	10.4	2.25	16.0	14.4	24.4	96.3	42.3	936
BU | LFS | -	9.64	1.85	14.9	11.0	23.1	84.4	40.0	824
BU | LMS | -	9.60	1.07	14.9	7.97	22.4	62.7	36.9	615
BU | LTS | -	136	16.9	225	122	343	1001	563	10089

Table 6.16: This table presents the average number of iterations (iter) and the average computation time (time) in milliseconds for four different sizes of the input point set "bunny". The initial motion between the data point set and the model point set was specified as $\theta_{angle} = 20°$, $\theta_{trans} = 7.5$, and $\theta_{scale} = 1.0$. The data point set was generated with the parameter values $\sigma_{inl} = 0.25$, $\sigma_{out} = 2.5$, $p_{out} = 0.1$, and $p_{cut} = 0.0$. The results in this table were averaged over 1000 test runs.

of the model points in the corresponding point pairs. As demonstrated in the preceding experiment, the basin of convergence of our variant of the ICP algorithm with respect to the scale factor between the point sets is limited when the scale of the data point set is smaller than that of the model point set. While the scale estimation reduces the basin of convergence of our variant of the ICP algorithm with respect to the initial motion between the point sets, rotations of 20° and translations of about 10% of the size of the point sets are still estimated reliably.

In the final experiment of this subsection, we analyze the computational efficiency of the extensions of our variant of the ICP algorithm for different sizes of the point set "bunny". Table 6.16 presents both the average number of iterations and the average computation time of the ICP algorithm for the configurations shown in

the first column of the table. The extension for integrated scale estimation increases the required computation time by at most 30% percent. While the LFS extension requires a complete sorting of the residuals of the corresponding point pairs, the LMedS extension only has to find the median value of the residuals. As a consequence, the LMedS extension is noticeably faster than the LFS extension. Finally, due to the repeated invocation of the ICP algorithm for the estimation of the optimum value of the fraction of outliers, the computation time of the LTS extension is much higher than that of the other extensions for robust correspondence estimation. Here, the stated numbers of iterations are summed over all required invocations of the ICP algorithm.

6.3.5 Summary

We rely on artificial test data to evaluate our variant of the ICP algorithm in this section. The generation of the test data is described in Subsection 6.3.1. In the next three subsections, we evaluate the accuracy, the basin of convergence, and the computational efficiency of our variant of the ICP algorithm with and without the implemented extensions for robust correspondence estimation and integrated scale estimation. In addition to that, we substantiate our choice for the default values of the exponents θ_{elfs} and θ_{elts} of the objective functions of the LFS extension and the LTS extension. The following results of our evaluation are especially important:

- Both the accuracy and the basin of convergence of our variant of the ICP algorithm strongly depend on the shape of the point sets. In particular, shapes with smooth surfaces are much better suited for registration than unstructured point clouds.
- The LMedS extension is the most efficient extension for robust correspondence estimation. However, it is also least robust to systematic outliers in the data point set.
- The LTS extension is most robust to systematic outliers, but its computation times are an order of magnitude longer than those of the other extensions for robust correspondence estimation.
- The LFS extension has no significant weakness and provides the best overall performance of the extensions for robust correspondence estimation.
- Our proposed extension for integrated scale estimation works best when the scale of the data point set is larger than that of the model point set. It also reduces the basin of convergence of the ICP algorithm with respect to the rotation angle and the translation vector length of the initial motion.

As the ICP algorithm is a locally convergent algorithm, there will always be applications that require a basin of convergence that lies beyond its capabilities. In

this case, it is possible to improve the initial alignment of the two point sets with a suitable pre-alignment algorithm. A short overview of these algorithms is given in Subsection 5.1.2. The basin of convergence of our variant of the ICP algorithm can be influenced by interchanging the data point set and the model point set. In general, the point set with the smaller relative number of outliers should be used as the data point set. When this information is not available, it is advisable to use the smaller point set as the data point set. In contrast to this, the additional estimation of the relative scale greatly benefits from choosing the point set with the larger relative scale as the data point set. Finally, the basin of convergence of our extension for integrated scale estimation can also be increased by combining it with other extensions of the ICP algorithm. Some extensions listed in Subsection 5.2.2 exploit additional information in the input data or trade computational efficiency for a larger basin of convergence.

6.4 Joint Operation

This section is concerned with the experimental evaluation of the joint operation of our feature point tracking system and our structure and motion estimation system. In Subsection 6.4.1, we describe the generation of the three test image sequences with known ground truth data and explain the selected parameter configurations of our systems. The experimental results for the reconstructions of the three image sequences are presented separately in the three subsequent subsections. Finally, Subsection 6.4.5 summarizes the most important results of the experimental evaluation in this section.

6.4.1 Experimental Setup

The experimental evaluation in this section is based on three image sequences, which were kindly provided by Ingo Scholz. All images were captured with a Sony DSR-PD100AP DV camera in the PAL format, but the provided images had already been symmetrically cropped to a resolution of 512×512 pixels. For every captured image, the camera pose was measured with an optical tracking system. To this end, a target with reflective markers was fixed to the camera and tracked by two infrared cameras with integrated infrared light sources. Further details on the image sequences and the generation of the ground truth data for the camera poses can be found in [Sch08a] and [Vog06].

As we evaluate the accuracy of our algorithms with the help of the available ground truth data for the camera poses, it is crucial to consider the accuracy of the ground truth data itself. First, the accuracy of the optical tracking system is limited. According to its manufacturer, the translation error is notably smaller than 1.0mm, and the rotation error lies below 0.14° [Sch08a]. Second, the estimation of the transformation between the coordinate systems of the tracked target and the

code	algorithm	parameters		
DEF	gd tra + gd aff iie	-		
STD	gd tra + gd aff iie	$\delta_{\text{ftr\_num}} = 300$	$\delta_{\text{ftr\_step}} = 4$	$\delta_{\text{ftr\_dist}} = 10$
		$\delta_{\text{tra\_min}} = 1$	$\delta_{\text{tra\_pred}} = *$	$\delta_{\text{tra\_size}} = 11$
HSP	gd tra + gd aff iie	$\delta_{\text{ftr\_num}} = 150$	$\delta_{\text{ftr\_step}} = 4$	$\delta_{\text{ftr\_dist}} = 20$
		$\delta_{\text{tra\_min}} = 1$	$\delta_{\text{tra\_pred}} = *$	$\delta_{\text{tra\_size}} = 11$

Table 6.17: We evaluate our feature point tracking system with the listed parameter configurations. For parameters not specified in this table, we use the default values defined in Table 3.1. The available motion estimation algorithms are presented in Table 6.1. The selected motion prediction type $\delta_{\text{tra\_pred}}$ is discussed in the text below.

camera requires a hand-eye calibration, which further reduces the accuracy of the ground truth data. A detailed description of the approach used for performing the hand-eye calibration can be found in [Vog06].

Furthermore, the measurement of the camera pose with the optical tracking system has to be synchronized with the capturing of the images. Our analysis of the image sequences strongly indicates that a problem with the synchronization of the camera and the optical tracking system affects the accuracy of the ground truth data. We provide further details on this problem in Subsection 6.4.4. Finally, as described in Subsection 4.1.1, our structure and motion estimation system requires the intrinsic camera parameters as input data. Due to the lack of dependable intrinsic camera parameters for the three provided image sequences, we use the following approximate intrinsic camera parameters for all experiments in this section

$$\boldsymbol{K} = \begin{pmatrix} 920 & 0 & 256 \\ 0 & 920 & 256 \\ 0 & 0 & 1 \end{pmatrix}, \quad D_1 = 0.0\,, \quad D_2 = 0.0\,. \tag{6.13}$$

Although we are not able to quantify the overall accuracy of the ground truth data exactly, the three provided image sequences represent a unique opportunity to evaluate our algorithms with highly realistic input data.

In this section, we evaluate our feature point tracking system with three parameter configurations, which are presented in Table 6.17. The motion estimation algorithms of our feature point tracking system are listed in Table 6.1. We apply the algorithms recommended in Subsection 6.1.4. The first parameter configuration DEF uses the default parameters of our feature point tracking system, which are specified in Table 3.1. For the second parameter configuration STD, we change six parameter values. We increase the number $\delta_{\text{ftr\_num}}$ of tracked feature points and the distance $\delta_{\text{ftr\_dist}}$ between them to improve the spreading of the feature points in the images. The adjusted values of $\delta_{\text{ftr\_step}}$ and $\delta_{\text{tra\_min}}$ increase the efficiency of our tracking system without affecting its accuracy. Furthermore, we activate both the

code	configuration	parameters
DEF	A|-|-|-|-|-	-
ST1	A|-|-|-|-|-	$\delta_{\text{len\_min}} = 10$
IB1	A|B-|-|-|C	$\delta_{\text{len\_min}} = 10$
BA1	A|-|-B-|-|C	$\delta_{\text{len\_min}} = 10$
FB1	A|-|-BB|C	$\delta_{\text{len\_min}} = 10$
HS1	-|-|-B-|-|C	$\delta_{\text{len\_min}} = 10$ $\delta_{\text{out\_iter}} = 64$ $\delta_{\text{inn\_iter}} = 32$ $\delta_{\text{tri\_iter}} = 16$ $\delta_{\text{sca\_iter}} = 32$ $\delta_{\text{bun\_iter}} = 12$
ST2	-|-|-|-|-|-	$\delta_{\text{len\_min}} = 10$ $\theta_{\text{seg\_rat}} = 0.5$
IB2	-|B-|-|-|C	$\delta_{\text{len\_min}} = 10$ $\theta_{\text{seg\_rat}} = 0.5$
BA2	-|-|-B-|-|C	$\delta_{\text{len\_min}} = 10$ $\theta_{\text{seg\_rat}} = 0.5$
HS2	-|-|-B-|-|C	$\delta_{\text{len\_min}} = 10$ $\theta_{\text{seg\_rat}} = 0.5$ $\delta_{\text{bun\_iter}} = 12$

Table 6.18: This table contains the parameter configurations for the evaluation of our structure and motion estimation system. The default values of all parameters can be found in Table 4.3. The mapping of the configuration code in the second column to three important parameters of our estimation system is explained in Table 6.4.

constant position prediction and the constant motion prediction with the parameter $\delta_{\text{tra\_pred}}$ to increase the robustness of our tracking system to erratic motions of the feature points. Finally, we increase the size $\delta_{\text{tra\_size}}$ of the feature windows to improve the basin of convergence of the translation estimation. The parameter configuration HSP is based on the configuration STD, but reduces the number $\delta_{\text{ftr\_num}}$ of tracked feature points and increases the distance $\delta_{\text{ftr\_dist}}$ between them.

The parameter configurations of our structure and motion estimation system are listed in Table 6.18. The configuration code in the second column of this table is explained in Table 6.4. Furthermore, the default parameters of our structure and motion estimation system are specified in Table 4.3. We increase the minimum trail length $\delta_{\text{len\_min}}$ in all parameter configurations except DEF to prevent the reconstruction of inaccurate feature positions from very short feature trails. The ability to reconstruct very short input sequences with the default value of $\delta_{\text{len\_min}} = 3$ is not required in the experiments in this section. The particular properties of the test sequence "table" necessitate the increase of $\theta_{\text{seg\_rat}}$, which decreases the average length of the generated segments. We further discuss this parameter setting in Subsection 6.4.4. All remaining parameter changes increase the computational efficiency of our estimation system by reducing either the number of LMedS iterations or the maximum number of iterations of our bundle adjustment approach.

In the subsequent subsections, we analyze the performance of our feature point tracking system by considering the number of generated feature trails, the average trail length, and the computation time of the tracking system. Furthermore, the quality of the estimated feature trails has a strong influence on the reconstruction

Figure 6.48: From left to right, this figure shows the images 1, 38, 75, 112, and 150 of the test sequence "glass".

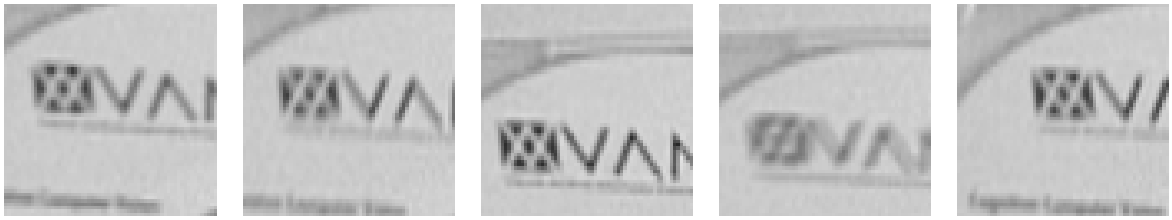


Figure 6.49: The subimages in this figure have a size of 64×64 pixels. They depict the VAMPIRE logo on the compact disc in the test sequence "glass". The subimages lie at the same position in the consecutive images 100, 101, 102, 103, and 104.

of the test sequence. In order to evaluate the quality of the reconstruction, which is computed by our structure and motion estimation system, we observe the average relative pairwise translation error $\bar{\epsilon}_{\text{rpt}}$ and the average absolute pairwise rotation error $\bar{\epsilon}_{\text{apr}}$. Both error measures are described in detail in Subsection 6.2.3. It is important to note that these two error measures are much better suited for evaluating the quality of a reconstruction than the back-projection error, which is commonly used when no ground truth data is available. Finally, we also specify the number T of created segments and the computation time of the reconstruction for every test run.

6.4.2 Test Sequence Glass

The test sequence "glass" comprises 150 images with a resolution of 512×512 pixels. Five images of this test sequence are shown in Figure 6.48. In addition to that, Figure 6.49 depicts five subimages of consecutive images of the test sequence. The subimages illustrate the moderate level of noise, the occurrence of motion blur, and the erratic motion of the camera in the test sequence "glass". These properties make the successful tracking of feature points more difficult. Furthermore, the captured scene contains a highly reflective jewel case, as well as a transparent glass vase, which strongly distorts the objects seen through it. Figure 6.50 illustrates the results of the feature point tracking. It can clearly be seen that the distortions caused by the glass vase affect the estimation of the affine motion of the feature windows. Obviously, the generation of inaccurate or erroneous feature trails also complicates

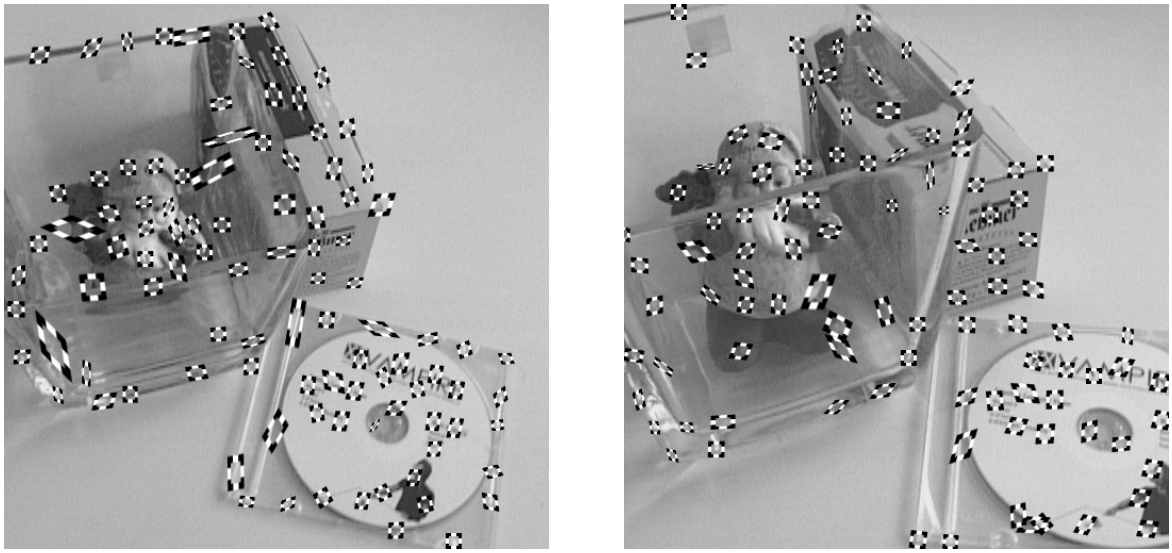


Figure 6.50: This figure illustrates the tracked feature points in the images 75 and 150 of the test sequence "glass". The feature trails were generated with the parameter configuration STD of Table 6.18, except for $\delta_{\text{ftr\_num}} = 100$, $\delta_{\text{ftr\_dist}} = 25$, and $\delta_{\text{ftr\_step}} = 1$. We reduced the number of tracked feature points and increased the distance between them in order to improve the visibility of the feature windows.

no.	fpt	trl	atl	time	sam	$\bar{\epsilon}_{\text{rpt}}$	$\bar{\epsilon}_{\text{apr}}$	T	time
1	DEF	1956	15.34	3450	DEF	6.59%	2.79°	8	2830
2	STD	1889	21.64	3687	ST1	7.79%	1.66°	5	2950
3	STD	1889	21.64	3654	BA1	3.35%	1.45°	5	15962
4	STD	1889	21.64	3696	FB1	3.29%	1.51°	5	41338
5	HSP	825	22.15	2162	HS1	3.35%	1.41°	6	2517

Table 6.19: The results in this table are based on the test sequence "glass". The parameter configurations for our feature point tracking system and our structure and motion estimation system are detailed in Table 6.17 and Table 6.18, respectively. The left part of this table lists the number of generated trails (trl), the average trail length (atl), and the computation time of the feature point tracking system in milliseconds. The right part contains the average relative pairwise translation error, the average absolute rotation error, the number of created segments, and the computation time of our structure and motion estimation system in milliseconds.

the subsequent reconstruction of the scene with our structure and motion estimation system.

The results of our reconstructions of the test sequence "glass" are presented in Table 6.19. Replacing the default configuration DEF of our feature point tracking system with the standard configuration STD substantially increases the average trail length. At the same time, the computational efficiency of the tracking system is

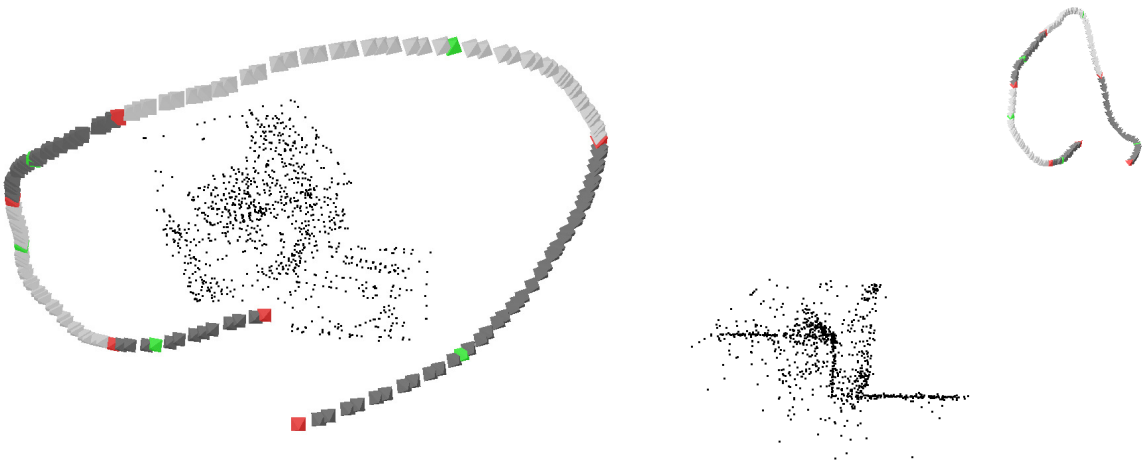


Figure 6.51: The illustrated reconstruction of the test sequence “glass” corresponds to test run 3 in Table 6.19, which resulted in an average relative pairwise translation error of 3.35%. The created segments are visualized by the alternating use of dark gray pyramids and light gray pyramids to represent the camera poses of one segment.

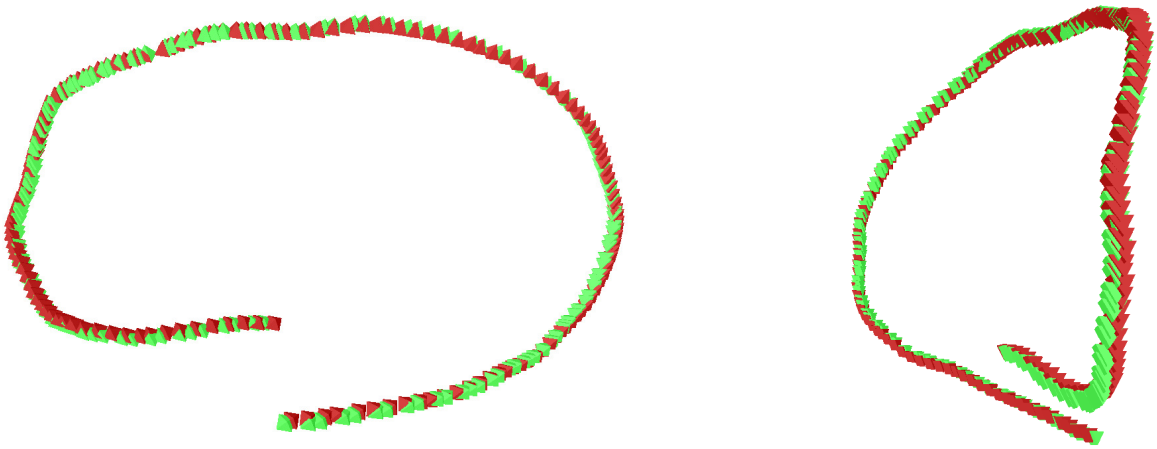


Figure 6.52: This figure illustrates the quality of the reconstruction of the test sequence “glass” by superimposing the camera poses of the reconstruction, which are visualized as bright (green) pyramids, and the camera poses of the ground truth data, which are represented by dark (red) pyramids. The reconstruction was computed with the configuration of test run 3 in Table 6.19.

only reduced marginally. The high speed configuration HSP cuts the number of tracked feature points in half. As a consequence, the computation time of the feature point tracking is reduced to less than 2.2 seconds, which equals a computation rate of more than 60 frames per second.

Although the average trail length is considerably increased in the second test run, the accuracy of the reconstruction improves only with respect to the average

Figure 6.53: From left to right, this figure contains the images 1, 125, 250, 375, and 500 of the test sequence "globe".

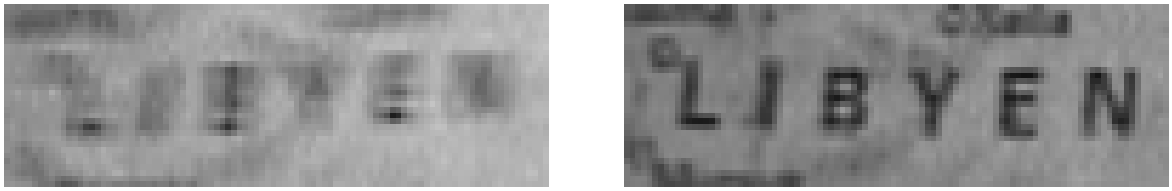


Figure 6.54: The two subimages in this figure have a size of 96×32 pixels. They belong to the images 148 and 158 of the test sequence "globe". The subimages demonstrate the occurrence of strong motion blur in the test sequence.

relative pairwise rotation error. In test run 3, we activate the optimization of the merged segments with our bundle adjustment approach, which reduces the relative pairwise translation error to 3.35%. Even the optimization of the complete reconstruction in test run 4 is unable to improve this error measure much further. Considering the unknown accuracy of the ground truth data and the use of approximate intrinsic camera parameters, we conclude that the accuracy of the last three test runs is close to optimal for the test sequence "glass". The reconstruction of test run 3 is illustrated in Figure 6.51. In addition to that, the accuracy of the reconstruction is visualized with the comparison of the reconstructed camera motion and the ground truth camera motion in Figure 6.52. When our structure and motion estimation system is tuned for high speed processing with the configuration HS1, the computation rate of the reconstruction increases to almost 60 frames per second.

6.4.3 Test Sequence Globe

The test sequence "globe" consists of 500 images with a resolution of 512×512 pixels. As the five exemplary images of this test sequence in Figure 6.53 show, a globe is the only object in the captured scene. The subimages in Figure 6.54 demonstrate the strong motion blur in some images of the test sequence, which constitutes a difficult problem for the tracking of the feature points in the test sequence "globe". In addition to that, the images in Figure 6.53 illustrate that most visible regions of the globe enter and leave the field of view several times throughout the image sequence. This effect, which is caused by the sweeping camera motion, results in relatively short feature trails. Despite the described problems, the test scene "globe" is well suited

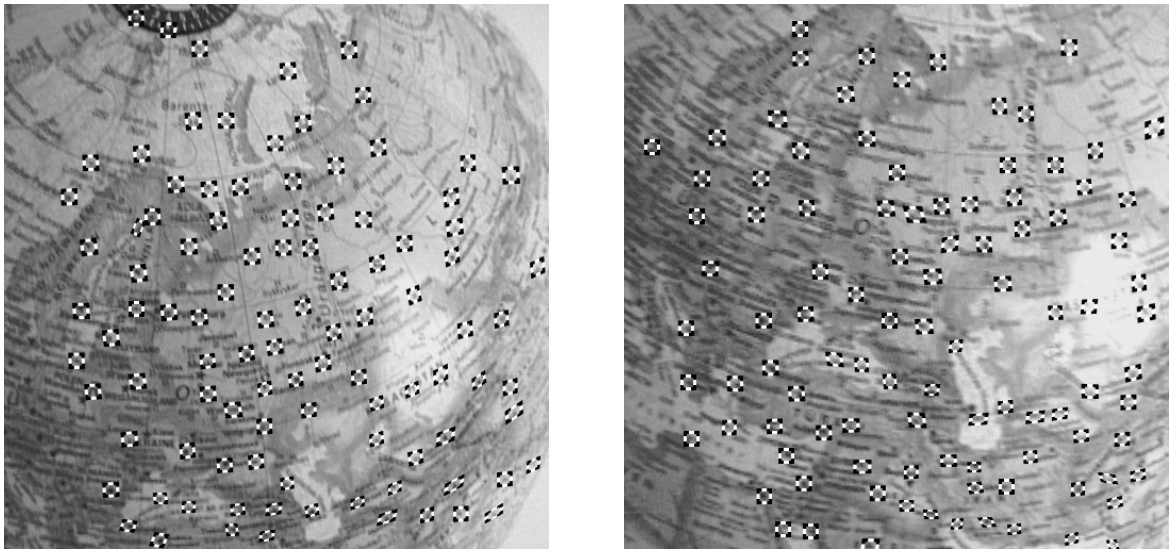


Figure 6.55: This figure illustrates the tracked feature points in the images 250 and 500 of the test sequence "globe". The feature trails were generated with the parameter configuration STD, except for $\delta_{\text{ftr\_num}} = 100$, $\delta_{\text{ftr\_dist}} = 25$, and $\delta_{\text{ftr\_step}} = 1$.

no.	fpt	trl	atl	time	sam	$\bar{\epsilon}_{\text{rpt}}$	$\bar{\epsilon}_{\text{apr}}$	T	time
1	DEF	7713	12.97	11772	DEF	5.70%	2.81°	29	10344
2	STD	5833	23.93	13071	ST1	4.86%	2.57°	20	11718
3	STD	5833	23.93	13092	IB1	3.55%	2.42°	20	15872
4	STD	5833	23.93	13098	BA1	3.47%	2.01°	20	43955
5	HSP	2885	24.22	8026	HS1	3.50%	2.04°	25	10014

Table 6.20: The results presented in this table are based on the test sequence "globe". The structure of this table is identical to the structure of Table 6.19.

for feature point tracking, so that the number of erroneous feature trails is comparatively small. The tracking of the feature points is visualized in Figure 6.55.

Table 6.20 summarizes the experimental results for the test sequence "globe". Although this test sequence is more than three times as long as the test sequence "glass", the average length of the estimated feature trails is only slightly larger. Compared to the default configuration DEF, the standard configuration STD of our feature point tracking system almost doubles the average trail length. We attribute this effect to the motion blur in some images of the test sequence, which necessitates an increase of the size of the feature windows for translation estimation. While the computation rate of our feature point tracking system exceeds 35 frames per second for the parameter configuration STD, the high speed configuration HSP further improves the computation rate to more than 60 frames per second.

The low values of the average relative translation errors for all configurations of our structure and motion estimation system confirm that the test sequence "globe"

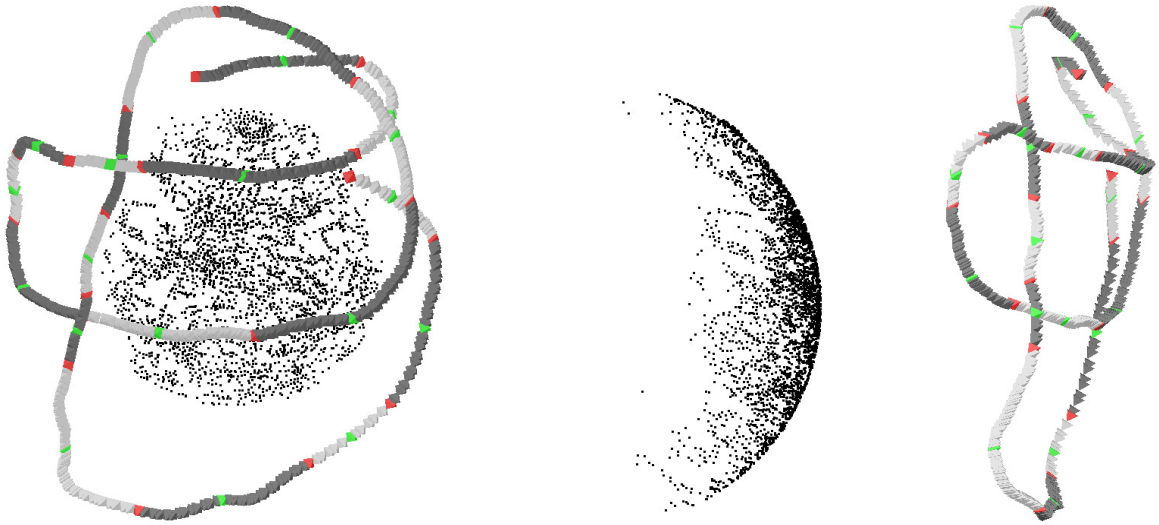


Figure 6.56: This reconstruction of the test sequence "globe" corresponds to test run 4 in Table 6.20, which yielded an average relative pairwise translation error of 3.47%. The created segments are visualized by the alternating use of dark gray pyramids and light gray pyramids to represent the camera poses of one segment.

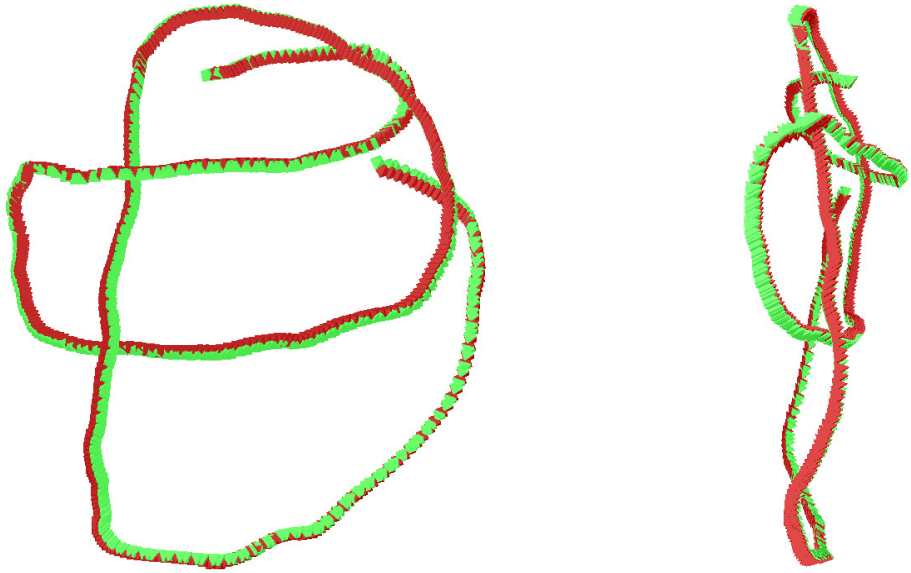


Figure 6.57: This figure illustrates the quality of the reconstruction of the test sequence "globe" by superimposing the camera poses of the reconstruction, which are visualized as bright (green) pyramids, and the camera poses of the ground truth data, which are represented by dark (red) pyramids. The reconstruction was computed with the parameter configuration of test run 4 in Table 6.20.

Figure 6.58: From left to right, this figure shows the images 1, 200, 310, 440, and 500 of the test sequence "table".

Figure 6.59: The first four images of this figure are subimages of the test sequence "table". The subimages have a size of 96 × 96 pixels and lie at the same position in the images 169, 170, 171, and 172. The rightmost image of this figure shows the complete image 379 of the test sequence "table".

is well suited for reconstruction. The best accuracy is achieved with the parameter configuration BA1, which uses the Cauchy estimator to increase the robustness of the optimization of the merged segments. The measured computation time of the corresponding test run is equivalent to a computation rate of more than 11 frames per second. The reconstruction computed in this test run is presented in Figure 6.56. In the right image of this figure, the spherical shape of the reconstructed globe is nicely illustrated. Furthermore, the aligned camera poses of the reconstruction and the ground truth data in Figure 6.57 demonstrate the very high accuracy of the reconstruction. The high speed configuration HS1 of test run 5 increases the computation rate of our structure and motion estimation system to nearly 50 frames per second, while almost retaining the accuracy of the parameter configuration BA1.

6.4.4 Test Sequence Table

Our final test sequence "table" also consists of 500 images with a resolution of 512 × 512 pixels. Five images of this test sequence are depicted in Figure 6.58. For several reasons, the test sequence "table" is the most difficult test sequence in this section. The four subimages in Figure 6.59 demonstrate that the translation of the scene objects in the images is very erratic. The combination of the erratic camera motion with the occurrence of many similar feature windows on the keyboard is prone to cause a large number of erroneous feature trails. This problem is further aggravated by the persistent noise and the recurrent motion blur in the images. In addition to

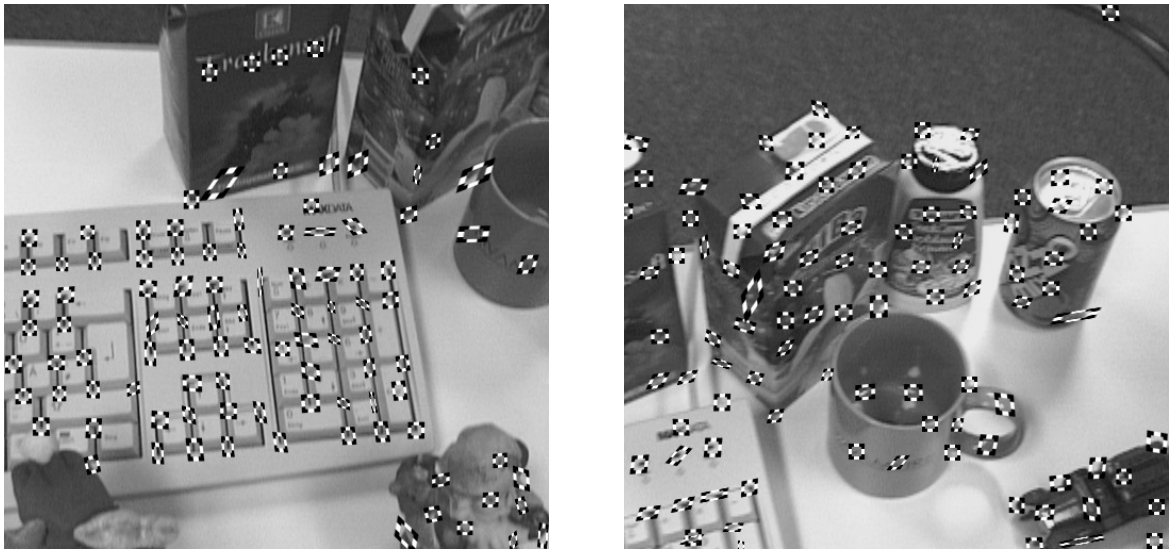


Figure 6.60: This figure illustrates the tracked feature points in the images 250 and 500 of the test sequence "table". The feature trails were generated with the parameter configuration STD, except for $\delta_{\text{ftr\_num}} = 100$, $\delta_{\text{ftr\_dist}} = 25$, and $\delta_{\text{ftr\_step}} = 1$.

that, the last image in Figure 6.59 illustrates the existence of large homogeneous regions in the images, which do not contain any suitable feature points. The two images in Figure 6.60 confirm that no feature points are tracked in the homogeneous regions.

Due to the specific camera motion, most scene objects remain in the field of view for a very short time, which leads to exceptionally short feature trails. Our structure and motion estimation system performs the initial reconstruction of every segment with feature points that are available in all images of the segment. When a new segment starts in the last image of Figure 6.59 and the camera moves to the right, it is very likely that most of the feature points available for the initial reconstruction of the resulting segment lie on the pen. In order to prevent similar disadvantageous situations, we increase the value of the parameter $\theta_{\text{seg\_rat}}$, which is explained in Subsection 4.3.2. As a result, the created segments, whose size is already limited by the short feature trails, become even smaller and more numerous. Finally, the first part of the test sequence "table" is dominated by a keyboard, which results in a mostly planar scene. In combination with the large number of erroneous feature trails, which are caused by the very similar appearance of the keys on the keyboard, the computation of the median view ray angles, which is also described in Subsection 4.3.2, becomes unreliable and has to be deactivated. The described changes of the parameter values are summarized in Table 6.18.

The results of our experiments with the test sequence "table" are presented in Table 6.21. As in the preceding experiments, the standard configuration STD of our feature point tracking system considerably increases the average length of the feature trails. The computation rate of our feature point tracking system is higher than

no.	fpt	trl	atl	time	sam	$\bar{\epsilon}_{\text{rpt}}$	$\bar{\epsilon}_{\text{apr}}$	T	time
1	DEF	8506	11.76	12152	DEF	47.45%	6.26°	32	11101
2	STD	6663	18.56	12047	ST2	21.04%	3.40°	56	5475
3	STD	6663	18.56	12022	IB2	11.24%	2.05°	56	17616
4	STD	6663	18.56	12120	BA2	4.54%	0.96°	56	38473
5	HSP	3232	18.39	7355	HS2	6.40%	1.55°	54	10413

Table 6.21: The results presented in this table are based on the test sequence "table". The structure of this table is identical to the structure of Table 6.19.

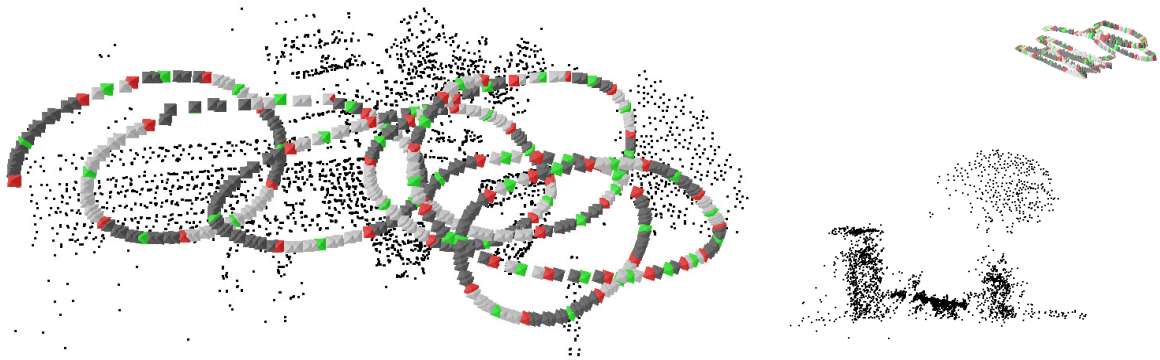


Figure 6.61: This reconstruction of the test sequence "table" corresponds to test run 4 in Table 6.21, which yielded an average relative pairwise translation error of 4.54%. The created segments are visualized by the alternating use of dark gray pyramids and light gray pyramids to represent the camera poses of one segment.

40 frames per second for the standard configuration STD, and improves to more than 60 frames per second for the high speed configuration HSP. Compared to the test sequence "globe", the deactivation of the computation of the median view ray angles decreases the computation time of our structure and motion estimation system for the standard parameter configuration ST2. Despite the substantial increase of the number of created segments, the computation times for the other parameter configurations are very similar.

The value of the average relative pairwise translation error for the default configuration in Table 6.21 confirms that the reconstruction of the test sequence "table" is a very difficult task. The optimization of the merged segments with our bundle adjustment approach reduces this error from 21.04% for the configuration STD to a very good value of 4.54% for the configuration BA2. The corresponding reconstruction is depicted in Figure 6.61. Furthermore, the reconstructed camera poses can be compared with the ground truth camera poses in Figure 6.62. Taking both the difficulty of the reconstruction and the achieved computational efficiency into account, we consider the accuracy of the high speed configuration in test run 5 to be more than acceptable.

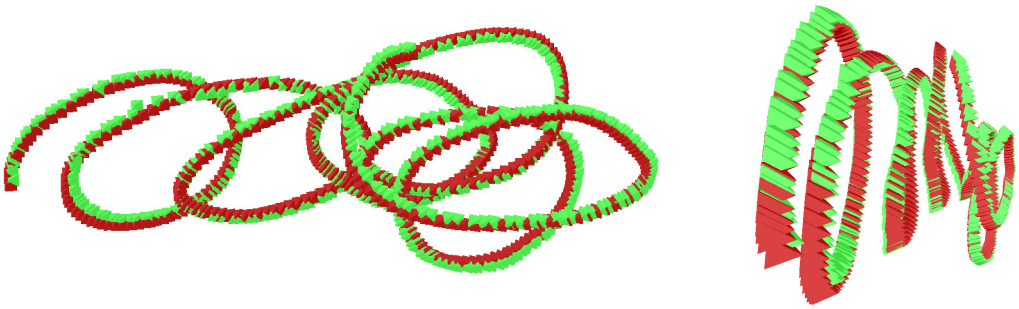


Figure 6.62: This figure illustrates the quality of the reconstruction of the test sequence "table" by superimposing the camera poses of the reconstruction, which are visualized as bright (green) pyramids, and the camera poses of the ground truth data, which are represented by dark (red) pyramids. The reconstruction was computed with the parameter configuration of test run 4 in Table 6.21.

images	mft	ect	gct
169 - 170	22.7	14.1	10.4
170 - 171	11.2	6.9	10.3
171 - 172	21.1	13.0	9.9
172 - 173	10.4	6.6	9.6

Table 6.22: This table provides important measurements for our analysis of the synchronization of the camera and the optical tracking system during the generation of the ground truth data. The first column of the table contains the indices of the examined images of the test sequence "table". The next two columns detail the median translation of all tracked feature points (mft) in pixels and the camera translation (ect) between the specified images for the reconstruction of test run 4. The last column presents the camera translation (gct) obtained from the ground truth data.

As announced in Subsection 6.4.1, we use the test sequence "table" to analyze the accuracy of the synchronization of the camera and the optical tracking system during the generation of the ground truth data. To this end, we consider the motion of the feature points and the motion of the optical center of the camera in the images 169 through 172, which have already been used to generate the subimages in Figure 6.59. The median translations of the feature points in Table 6.22 indicate that there are two different durations for the time intervals between the capturing of two consecutive images. The longer time intervals are apparently twice as long as the short ones. This observation is supported by the reconstructed camera translation. In contrast to this, the ground truth data provided by the optical tracking system represents a smooth camera motion. Our analysis suggests that the reconstructions computed by our structure and motion estimation system are actually more accurate than the provided ground truth data with respect to the relative camera positions of adjacent views.

6.4.5 Summary

The experimental evaluation in this section complements the evaluation of our feature point tracking system in Section 6.1 and the evaluation of our structure and motion estimation system in Section 6.2. In the first subsection, we describe the generation of the test image sequences and the corresponding ground truth data. Each one of the three subsequent subsections details the experimental results of the evaluation of the joint operation of our feature point tracking system and our structure and motion estimation system for one test sequence. We draw the following conclusions from the conducted experiments:

- Our feature point tracking system is able to cope with very difficult conditions, including erratic feature motion and strong motion blur.
- For the evaluated parameter configurations, the computation rate of our tracking system exceeds 35 frames per second with 300 feature points and 60 frames per second with 150 feature points.
- In accordance with the evaluation of our structure and motion estimation system in Section 6.2, the best overall performance is achieved by activating the bundle adjustment optimization of the merged segments in combination with the Cauchy estimator.

In the following, we provide a small set of instructions for processing arbitrary image sequences with our systems:

- For our feature point tracking system, we recommend the parameter configuration STD presented in Table 6.17 as a good starting point. The number $\delta_{\text{ftr\_num}}$ of tracked feature points should be adapted to the resolution of the input images.
- When the quality of the input image sequence is high, it is possible to decrease the feature window size $\delta_{\text{tra\_size}}$ to its default value of seven without affecting the quality of the generated feature trails.
- The computational efficiency of our tracking system can be improved by reducing the number $\delta_{\text{ftr\_num}}$ of tracked feature points. In this case, the distance $\delta_{\text{ftr\_dist}}$ between the feature points should be increased to ensure that they are evenly spread over the input images.
- For our structure and motion estimation system, we recommend the versatile parameter configuration BA1 detailed in Table 6.18.
- When the reconstructed scene is mostly planar, deactivating the computation of the view ray angles potentially improves the segmentation.

- The parameter configuration HS1 in Table 6.18 demonstrates our preferred approach for increasing the computational efficiency of our structure and motion estimation system. In combination with the reduced number of 150 tracked feature points, this parameter configuration increased the computation rate of our system to approximately 50 frames per second in the conducted experiments.

All in all, the experimental evaluation in this section successfully demonstrated the outstanding performance of the joint operation of our feature point tracking system and our structure and motion estimation system.

Chapter 7

Applications

This chapter presents several applications of our work in the area of sparse 3-D reconstruction. As described in Section 1.1, our proposed algorithms for sparse 3-D reconstruction can be categorized into three main work areas. These work areas, which comprise feature point tracking, structure and motion estimation, and point set registration, are detailed in Chapter 3, Chapter 4, and Chapter 5, respectively. In this chapter, we focus on actual applications of our algorithms in order to demonstrate their high level of performance and versatility. Thus, we are less concerned with the large number of potential applications.

Most of the presented applications are related to our feature point tracking system, because we started our work in the area of sparse 3-D reconstruction with the implementation of this system. As a consequence, we had plenty of time to integrate our feature point tracking system into a wide range of different applications. In contrast to this, our structure and motion estimation system and our extensions of the ICP algorithm are more recent developments, which considerably deferred their widespread adoption.

The first section of this chapter is dedicated to applications of our algorithms in the VAMPIRE project [Sag11]. These applications are either part of the integrated demonstrator or belong to one of the supplementary demonstrations. There are no external references for most of these applications. In contrast to this, the applications presented in Section 7.2 are not directly connected to the VAMPIRE project, but have been published in the technical literature.

7.1 Applications in the VAMPIRE project

7.1.1 Hybrid Self-Localization

The hybrid self-localization of the augmented reality gear is the most prominent application of our feature point tracking system in the VAMPIRE project. The requirements of this application have already been detailed in Subsection 3.4.1. In short, a small number of features have to be tracked at a very high frame rate in small subwindows of selectable size and position. A predecessor of the hybrid self-localization approach, which also relies on the combination of optical tracking and

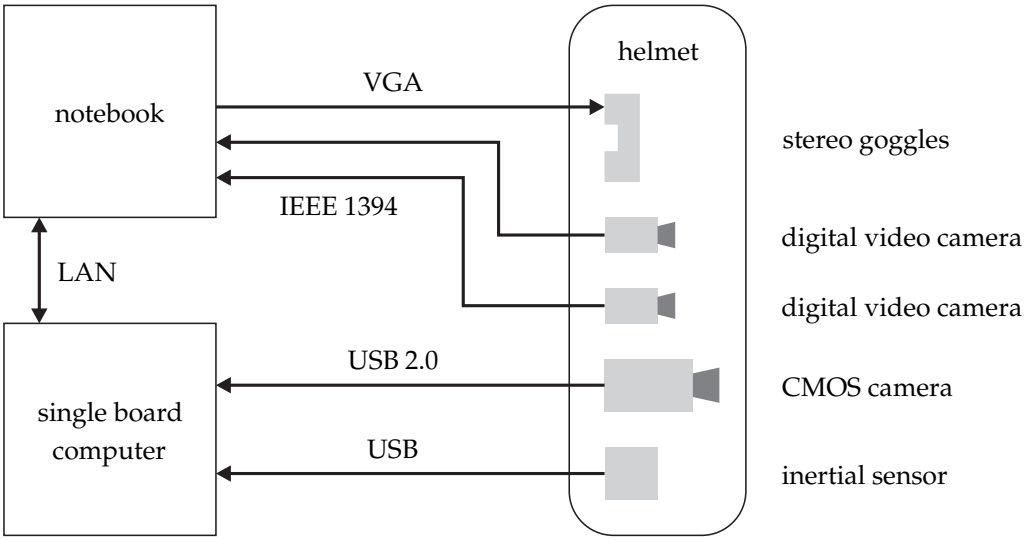


Figure 7.1: This drawing gives a technical overview of the VAMPIRE augmented reality gear, which is depicted in Figure 1.1.

an inertial sensor, is described in [Rib02]. The image data for the optical tracking is captured by the CMOS camera presented in [Mue04].

The augmented reality gear of the VAMPIRE project is depicted in Figure 1.1. Furthermore, Figure 7.1 provides a technical overview of this augmented reality gear. The CMOS camera and the inertial sensor are both directly connected to the single board computer, which is responsible for performing the hybrid self-localization. As the single board computer, which is powered by rechargeable batteries, has considerably less processing speed than a modern workstation, the efficiency of the applied feature point tracking system is very important. The notebook in Figure 7.1 applies the estimated pose of the helmet to control the user interaction with the VAMPIRE cognitive vision system.

Our adapted tracking system for high-speed feature point tracking has been proposed in Subsection 3.4.3. The most important change with respect to our standard feature point tracking system is the substitution of the gradient descent translation estimation with a block matching algorithm, which enables the adapted tracking system to work on independently captured subwindows. Exemplary subwindows are illustrated in Figure 3.16. The performance of the resulting optical tracking subsystem is shown in Figure 7.2. When complete images are captured, the frame rate is limited by the pixel clock of the CMOS sensor of the camera and the bandwidth of the data transfer to the single board computer. In contrast to this, capturing subwindows at the predicted positions of the feature points allows much higher frame rates. Thus, the optical tracking reaches approximately 200 frames per second for 10 tracked feature points and about 100 frames per second for 20 feature points. The achieved frame rates are especially beneficial for tracking fast head movements of the user.

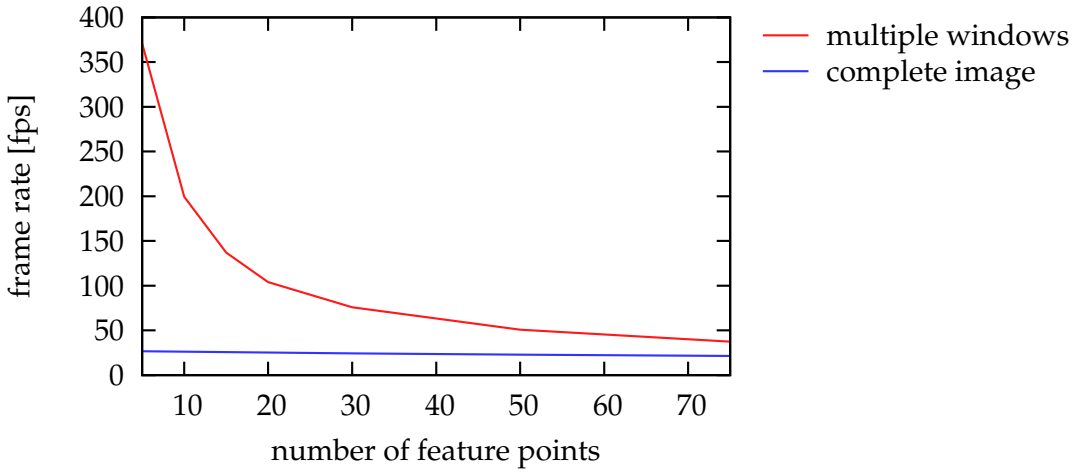


Figure 7.2: This graph plots the frame rate achieved by the optical tracking subsystem of the hybrid self-localization against the number of tracked feature points. In this experiment, our high-speed feature point tracking system processed either complete images or independently captured subwindows. The underlying measurements were kindly provided by Gerald Schweighofer of the University of Technology Graz.

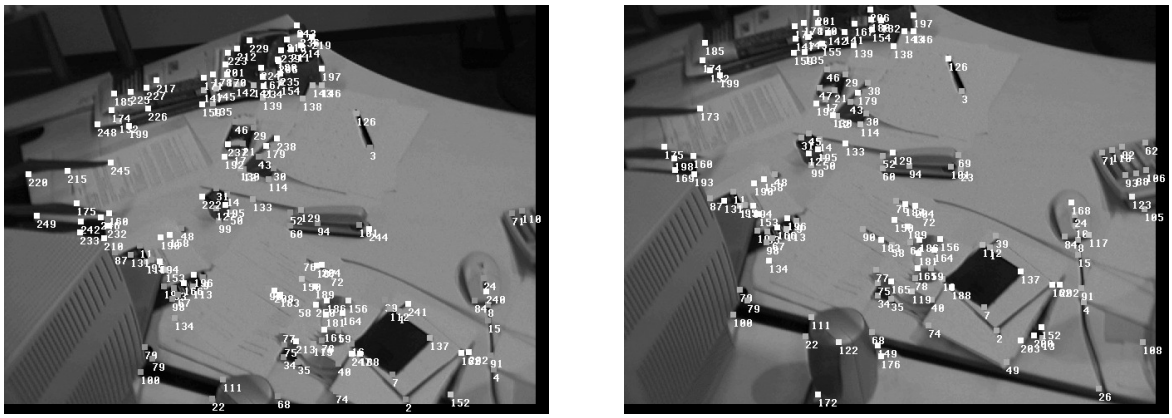


Figure 7.3: This figure depicts two exemplary input images of the approach for generating mosaics. The tracked feature points are visualized as small bright squares. For every feature point, an additional number identifies the corresponding trail.

7.1.2 Pictorial Scene Representation

In the VAMPIRE cognitive vision system, mosaics are used as a compact and non-redundant representation of image sequences. In order to avoid parallax artifacts, which occur when a moving camera records a non-planar scene, the approach described in [Gor04] partitions the scene into approximately planar subscenes and generates one mosaic for every subscene. This approach requires the identification of a large number of feature point correspondences for the detection of the planar subscenes and the recovery of the camera motion.

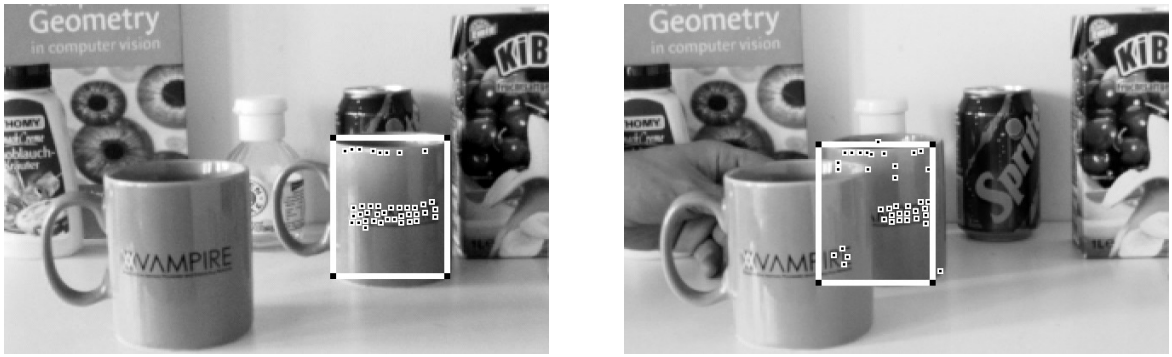


Figure 7.4: The two images in this figure illustrate the tracking of a moving cup with the combined object tracking approach. The tracked image region is marked with a large rectangle. The feature points are identified by small squares.

In the course of the VAMPIRE project, our standard feature point tracking system, which is detailed in Section 3.3, was successfully integrated into the described approach for generating mosaics. Figure 7.3 depicts two exemplary input images and visualizes the tracked feature points. In this application, the ability to efficiently track a large number of feature points was an important asset of our feature point tracking system. Furthermore, the hierarchical translation estimation described in Subsection 3.3.5 enabled our tracking system to cope with large feature displacements.

7.1.3 Data-Driven Object Tracking

Object tracking is an important prerequisite for several high-level components of the VAMPIRE cognitive vision system, like action recognition. Thus, many different algorithms for object tracking were evaluated in the VAMPIRE project. In particular, data-driven object tracking is concerned with tracking unknown objects, for which no object model is available. Several different data-driven object tracking algorithms are compared in [Deu05]. The presented experimental results indicate that the color histogram based CONDENSATION (conditional density propagation) approach is especially robust to partial occlusions and appearance changes. In order to improve the substandard accuracy of this approach with respect to the estimated 2-D translation, we combined the color histogram based tracking with our feature point tracking.

The basic task of our feature point tracking system in this application is to select feature points inside the tracked image region and to independently track the selected feature points. The final result of the object tracking is determined by a probabilistic fusion of the distance between the color histograms and the number of feature points inside the prospective image region with the CONDENSATION approach. The performed experiments demonstrate that the combined object tracking approach increases the accuracy of the color histogram based tracking and reduces

its propensity to confuse similarly colored objects. This property is illustrated in Figure 7.4, where the combined object tracking approach correctly tracks the rearmost cup, even though it is partially occluded by the identically colored cup in the front. As the original CONDENSATION approach requires a computation time of about 80 milliseconds per input image, the application of our feature point tracking system increases the computation time of the combined object tracking approach only slightly.

7.1.4 Model-Based Object Tracking

In contrast to data-driven object tracking, model-based object tracking requires the generation of an object model and the recognition of the object to be tracked. In return, the object tracking approach in this subsection allows the estimation of the 6-D pose of the object. The presented object tracking approach is closely related to the approach described in [Grä05]. Both approaches use SIFT features to generate a 3-D object model from a training video sequence. The object tracking is initialized by automatically matching the SIFT features of the input image and the object model before estimating the pose of the object with the POSIT algorithm. The presented object tracking approach independently tracks the feature points at the positions of the SIFT features with our feature point tracking system. For every input image, the pose of the object is robustly determined with the least median of squares technique by applying the three-point algorithm in the random sampling phase and refining the set of inliers with the POSIT algorithm as described in Subsection 4.3.4.

Due to the computational efficiency of the applied algorithms, the presented object tracking approach achieves a computation rate of more than 30 frames per second. The integrated intensity equalization of our feature point tracking system, which is detailed in Subsection 3.3.4, increases the robustness of the object tracking approach to intensity changes. Furthermore, the application of the least median of squares technique improves its robustness to partial occlusions of the object. As the presented approach has to perform a 3-D reconstruction for the generation of the 3-D object model, it would be an ideal application of our structure and motion estimation system, which is detailed in Section 4.3. Finally, the presented object tracking approach could benefit from the increased accuracy of our POSIT algorithm with virtual reference point, which has been proposed in Subsection 4.2.2.

7.2 Other Applications

7.2.1 Online Structure and Motion Estimation

A new algorithm for globally convergent structure and motion estimation in real-time is presented in [Sch08b]. It is mainly intended for mobile augmented reality applications, working both with and without artificial markers in the scene. As a

consequence, it is evaluated on a mobile demonstration platform, which is based on a tablet computer and two calibrated cameras. The proposed structure and motion estimation algorithm requires the independent identification of feature point correspondences in the two image sequences captured by the stereo camera setup. Feature points that relate to the same 3-D point are linked together in a separate operation afterwards.

The authors of [Sch08b] identify the required feature point correspondences with our adapted feature point tracking system, which is described in Subsection 3.4.3. In this application, the most important property of our feature point tracking system is its high computational efficiency, because the limited processing speed of the mobile tablet computer has to suffice for the feature point tracking, the structure and motion estimation, and the visualization of the augmentation in real-time. Furthermore, this application requires the estimated feature positions to be very accurate and as reliable as possible, because the proposed structure and motion estimation algorithm is not designed to robustly deal with outliers.

7.2.2 Visualization with Image-Based Models

As an alternative to the manual geometric modeling of an object with textured polygons, image-based rendering techniques promise the automatic generation of photorealistic models of a real object from an image sequence of this object. [Nie05] provides a short introduction to light fields, which are a prominent representative of image-based models. The generation of light fields is an application area for the algorithms of all three work areas of this thesis, i. e., feature point tracking, structure and motion estimation, and point set registration. However, although our structure and motion estimation system is perfectly capable of performing the required reconstruction of the camera motion and the scene structure, it was not completed in time for a use in the following applications.

The application of our feature point tracking system for the generation of static and dynamic light fields is described in [Sch05] and, in more detail, in [Sch08a]. As the captured image sequences typically comprise more than 100 images, this application especially benefits from the feature drift prevention approach proposed in Subsection 3.3.3 and the intensity equalization approach described in Subsection 3.3.4. Furthermore, our feature point tracking system is also used for the generation of light fields during endoscopic surgery in [Vog04]. In this application, the high computational efficiency of our feature point tracking system reduces the waiting time for the generation of the light field.

The approach for the reconstruction of the light fields in [Sch08a] has two use cases for the ability of our feature point tracking system to process the input images in an arbitrary order. The first use case complements the standard chronological processing of the input images with a second pass through the image sequence in reverse chronological order, i. e., from the last image to the first. This approach allows the extension of all feature trails that do not start in the first image of the

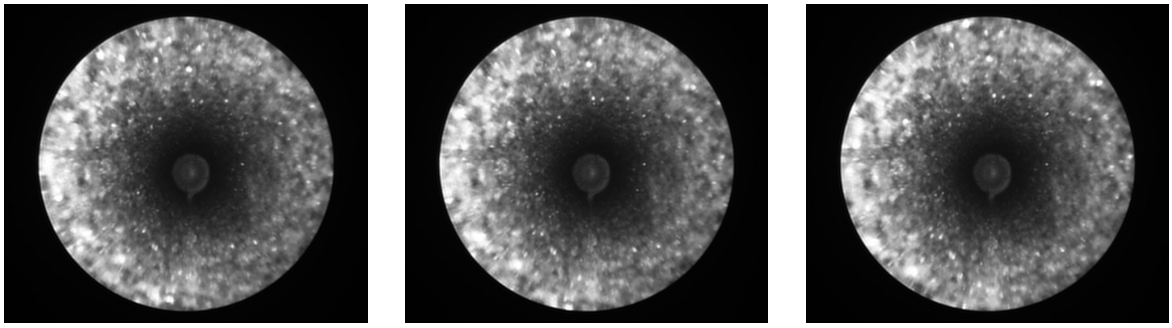


Figure 7.5: This figure illustrates the image quality of three preprocessed fiberscopic images of a cylindrical bore hole.

image sequence. The second use case, which performs non-sequential tracking, uses information about the camera poses to process image pairs that correspond to adjacent camera positions in the 3-D scene. This approach potentially prevents the problem that the chronological processing of the image sequence generates multiple feature trails for the same 3-D feature point when the 3-D feature point enters and leaves the field of view of the camera several times throughout the image sequence.

Our variant of the ICP algorithm with integrated scale estimation is used in [Sch08a] for evaluating the reconstruction of the camera motion. In particular, the algorithm is responsible for aligning the optical centers of the cameras in the reconstruction and the ground truth data in order to estimate the scale factor between the respective world coordinate systems. Another application of the ICP algorithm is the fusion of the reconstructed light field and the corresponding computed tomography data set in [Vog04]. As this operation requires the estimation of a scale factor between the two data sets, our variant of the ICP algorithm with integrated scale estimation is ideally suited for it.

7.2.3 Reconstruction from Fiberscopic Images

The work in [Win05] is concerned with the reconstruction of cylindrical bore holes from monocular fiberscopic images. The applied fiberscopes contain between 3000 and 50000 fibers, each of which transmits only one intensity value. In the captured images, these intensity values are represented by circular regions of pixels, which are surrounded by dark rings due to the physical properties of the glass fibers. As the resulting honeycomb pattern strongly degrades the image quality, it has to be eliminated in a preprocessing step. In addition to that, the only discernible textures in the bore holes are shadows created by the rough surface. Three exemplary images are depicted in Figure 7.5. Finally, the distortion of the feature windows, which is caused by the forward motion of the fiberscope into the cylindrical bore hole, substantially deviates from the affine distortion model used by our feature point tracking system.

Despite the high level of difficulty of the described application, the feature trails generated by our feature point tracking system enabled the reconstruction of the scene structure in [Win05]. This achievement corroborates the high versatility of our feature point tracking system. In our opinion, this application involves the most difficult conditions that our feature point tracking system is able to cope with, especially with respect to the poor quality of the available feature points and the strong distortion of the feature windows.

Chapter 8

Summary and Outlook

8.1 Summary

This work is dedicated to the development, enhancement, and evaluation of algorithms for sparse 3-D reconstruction from image sequences. In this work, we use the term 3-D reconstruction to denote all aspects of the reconstruction and the processing of the scene structure and the camera motion of an input image sequence. In contrast to approaches that generate or process dense samplings of the scene structure, like range images, algorithms for sparse 3-D reconstruction represent the scene structure with a limited number of feature points.

We categorize the algorithms for sparse 3-D reconstruction that are discussed in this thesis into three main work areas. These three work areas are concerned with feature point tracking, structure and motion estimation, and point set registration, respectively. In short, feature point tracking estimates the 2-D motion of distinct 3-D scene elements throughout the image sequence. Structure and motion estimation algorithms use the estimated 2-D feature positions to reconstruct the 3-D scene structure and the camera motion. Finally, our point set registration algorithms align two point sets that represent the same object. Due to our proposed extension for integrated scale estimation, they can be used to combine the 3-D scene structures of two independent reconstructions of the same scene. Interestingly, only the feature point tracking algorithms of our first work area operate directly on the input image sequences.

The algorithms of our three main work areas share three performance criteria, which are computational efficiency, accuracy, and robustness. It is important to note that these performance criteria conflict with each other in all three work areas. In order to allow the use of the developed algorithms in a wide range of applications, we attached great importance to their versatility, which is illustrated by their large number of user-specified parameters. As a consequence, the experimental evaluation of the algorithms forms a very important part of this work, because it is not only responsible for testing them with respect to the identified performance criteria, but also provides valuable information for setting the user-specified parameters to values that suit prospective applications. In the following, we examine the three main work areas one after another.

Our feature point tracking system is designed to be fully data-driven as well as highly versatile. It is derived from the Kanade-Lucas-Tomasi tracker, which uses a specialized Gauss-Newton gradient descent optimization algorithm as its motion estimation algorithm. The standard Kanade-Lucas-Tomasi tracker is very customizable and offers a high level of computational efficiency and accuracy. In addition to the motion estimation algorithm, it comprises algorithms for feature detection and outlier rejection.

During feature detection, the quality of the prospective feature windows is computed for every pixel of the input image and stored in an interest image. Our experimental evaluation of two different quality measures, the Harris detector and the Tomasi-Kanade detector, revealed only negligible differences with respect to the performance of the tracking system. After feature detection, the actual feature positions are selected with the help of the computed interest images. Our proposed feature selection strategy allows the definition of a minimum distance between the selected feature points. It also uses a hierarchical search structure to reduce the computation time of the feature selection.

Compared to the original motion estimation algorithm, the use of the inverse compositional algorithm considerably increases the computational efficiency of the motion estimation in our feature point tracking system. Thus, we are able to complement the translation estimation with the estimation of the affine motion for every feature window in every frame at low computational cost. As we estimate the affine motion between the first frame and the current frame, this approach virtually eliminates the feature drift problem and strongly increases the accuracy of the motion estimation in long image sequences. In addition to that, we are able to use the estimated affine distortion matrix for rejecting highly distorted feature windows as outliers in every frame.

The basin of convergence of the translation estimation approximately equals half the size of the feature windows. In order to cope with larger feature point displacements, our feature point tracking system performs a hierarchical translation estimation on a Gaussian image pyramid. This approach increases the basin of convergence of the translation estimation to approximately 20 pixels for an image pyramid with four levels and feature windows with a size of 11×11 pixels. We also devised a robustified result propagation strategy, which protects our tracking system against estimation errors on the higher levels of the image pyramid. In addition to that, this strategy supports the use of multiple starting positions for the translation estimation, which are predicted by assuming a constant position or a constant motion of the feature points. In order to allow the utilization of application-specific information, our feature point tracking system also accepts externally predicted feature positions.

One severe shortcoming of the standard Kanade-Lucas-Tomasi tracker is its low robustness to intensity changes in the input images. Our solution to this problem is the integration of an affine linear model for intensity equalization into the inverse compositional algorithm for motion estimation. Our experiments prove that

this approach is more efficient than existing approaches that estimate the motion of the feature points and the intensity equalization in alternation. As the intensity changes in consecutive images of an image sequence are quite small in most applications, it is usually sufficient to activate the integrated intensity equalization for the estimation of the affine motion.

In order to improve the efficiency of our feature point tracking system for small numbers of feature points, we replaced the gradient descent translation estimation algorithm with a block matching algorithm. This approach greatly reduces the computational overhead per image, because derivative images only have to be computed for input images that are used for feature detection. In our experiments, using the block matching algorithm for tracking 20 feature points reduced the computation time from approximately 4.5 ms per frame to less than 3 ms per frame. As expected, the gradient descent translation estimation algorithm is more efficient when a large number of feature points have to be tracked. It requires less than 20 ms per frame for tracking 400 feature points. Even when lost feature points are replaced in every fourth frame, the computation rate of our feature point tracking system exceeds 35 frames per second for tracking 300 feature points and 60 frames per second for tracking 150 feature points.

The versatility of our tracking system is illustrated by the large number of successful applications, which use both monocular and stereo image sequences captured by hand-held, helmet-mounted, and stationary digital cameras, as well as endoscopes and fiberscopes. An adapted version of our feature point tracking system supports the self-localization of the VAMPIRE augmented reality gear. Processing small subimages, this version tracks ten feature points at a rate of 200 frames per second on a relatively slow mobile computer.

In the second work area, we designed a versatile structure and motion estimation system by combining efficient algorithms for the initial reconstruction of the scene with several optional applications of an efficient bundle adjustment approach. Furthermore, the robustness of our estimation system to outliers is ensured by the comprehensive application of M-estimators and the least median of squares technique. In addition to the feature trails generated by our feature point tracking system, our structure and motion estimation system requires the intrinsic camera parameters as input data.

The basic approach of our structure and motion estimation system for the initial reconstruction of the scene consists of three distinct steps. First, the input image sequence is partitioned into segments by our proposed key frame selection algorithm. Its main idea is to consider the median of the sines of the view ray angles of the feature points in the prospective key frames as a quality measure for the segmentation. Our experiments indicate that this approach improves the accuracy of the reconstruction, as long as the necessary estimation of the relative camera poses with the five-point algorithm works reliably.

In the second step, the segments are independently reconstructed with efficient algorithms like the midpoint triangulation algorithm, the three-point algorithm and

the POSIT algorithm for absolute camera pose estimation, and the five-point algorithm for relative camera pose estimation. We enhanced the POSIT algorithm to work with a virtual reference point, which results in a considerably more accurate absolute camera pose estimation when the number of input feature points is high. The third step comprises the merging of the reconstructed segments, which requires the estimation of their relative scale. In all three steps, the robustness of the algorithms to outliers is improved with the least median of squares technique. All in all, the initial reconstruction exhibits a computation rate of more than 40 frames per second in our experiments.

In order to improve the accuracy of the initial reconstruction, we added four optional applications of an efficient bundle adjustment approach to our structure and motion estimation system. Furthermore, we tested the effects of three M-estimators on the performance of the bundle adjustment approach. Our extensive experimental evaluation indicates that the best overall performance is achieved with the bundle adjustment optimization of the merged segments in combination with the Cauchy estimator. Our experiments with the three image sequences of real-world scenes illustrate the outstanding accuracy of this configuration, which results in a computation rate of approximately ten frames per second. These experiments also show that the computational efficiency of this configuration can be dramatically increased by reducing the number of tracked feature points and limiting the number of iterations of the bundle adjustment. These changes slightly reduce the accuracy of the reconstruction, but increase the computation rate to 50 frames per second.

Our third work area is concerned with 3-D point set registration. In this work, we focussed on the ICP algorithm, which is the most popular locally convergent algorithm for aligning two 3-D point sets. Its basic approach is to determine the set of corresponding point pairs for the current alignment and to optimally align the current point pairs in alternation. Our experimental evaluation of the ICP algorithm showed that both the accuracy and the basin of convergence of the ICP algorithm strongly depend on the shape of the input point sets.

In addition to the outliers caused by estimation errors during the generation of the input point sets, point set registration algorithms have to cope with systematic outliers commonly resulting from partially overlapping point sets. Consequently, we analyzed and evaluated three different extensions of the ICP algorithm for robust correspondence estimation. In our experiments, the recently proposed least fractional squares extension provided the best balance between robustness to outliers, basin of convergence, and computational efficiency.

Finally, we presented an extension of the ICP algorithm for the integrated estimation of the relative scale of the two input point sets. Although the estimation of the additional parameter value reduces the basin of convergence with respect to the rotation angle and the translation vector length of the initial motion, the accuracy of the estimated scale is very high. Interestingly, our experiments indicate that the basin of convergence with respect to the scale factor is much larger when the scale of the data point set is larger than that of the model point set.

8.2 Outlook

The computational efficiency of the presented algorithms is an important aspect in all three work areas of this thesis. In recent years, there has been a paradigm shift in the computer hardware industry from single-core central processing units with ever increasing clock rates to multi-core central processing units with moderate clock rates. In addition to that, the field of general-purpose computation on graphics processing units is starting to offer unprecedented processing speed for a wide range of scientific applications. However, taking advantage of these developments requires the parallelization of the implemented algorithms. In this thesis, both the independent motion estimation of the feature points and the independent reconstruction of the segments facilitate a parallel implementation. Furthermore, preliminary experiments with a parallelized version of the nearest neighbor search in the ICP algorithm yielded promising results for large point sets in our third work area.

The impressive performance of our purely data-driven feature point tracking system does not leave much room for further improvements. The accuracy of the reconstruction from fiberscopic images in Subsection 7.2.3 could arguably be improved by estimating the perspective distortions of the feature windows. However, the associated increase in the number of estimated parameters also decreases the basin of convergence of the motion estimation, especially when our integrated intensity equalization is used. A standard way of enhancing data-driven algorithms is to take advantage of additional model knowledge. On the one hand, a close coupling between our feature point tracking system and a structure and motion estimation algorithm allows the prediction of the feature positions in the input images. This idea can either be realized by estimating the camera pose after tracking a small number of feature points in every input image or by applying a two-pass approach that processes the complete image sequence twice, as proposed in [Sch08a, pages 58ff.]. On the other hand, in order to improve the detection of outliers, a combination of our feature point tracking system with an object recognition algorithm could identify groups of feature points that are bound to stay in close proximity to each other. Furthermore, image understanding algorithms could help to detect and avoid depth discontinuities, which often lead to erroneous feature trails.

The main building block of our structure and motion estimation system is the five-point algorithm presented in [Nis04b]. Although it is theoretically capable of dealing with planar scenes, as long as a third view is used for the disambiguation of its tentative solutions, our evaluation strongly indicates that inaccurate input data can lead to the selection of the wrong solution. This behavior calls for further investigation, because it is especially detrimental to our proposed key frame selection algorithm. Due to the large number of possible applications of our structure and motion estimation system, there are many opportunities for further improvements and adaptations. For example, the elimination of input data from images with little camera motion could increase the efficiency of the reconstruction of the scene

structure. Finally, violations of the basic assumptions of our structure and motion estimation system, like independently moving objects in the captured scene, often lead to completely incorrect reconstructions. Although approaches for solving some of these problems exist in the literature, they lie far outside the scope of this work. However, the ability to automatically evaluate the quality of the performed reconstructions could significantly improve the reliability of our structure and motion estimation system.

In the area of point set registration with the ICP algorithm, we deliberately focussed our attention on the evaluation of different techniques for robust correspondence estimation and the proposal of our extension for integrated scale estimation. As a consequence, we explicitly encourage the examination of the interoperability of the discussed extensions and the large number of other existing extensions in the context of an actual application. In particular, extensions that increase the basin of convergence of the ICP algorithm could considerably improve the applicability of our extension for integrated scale estimation.

Appendix A

German Translation

A.1 Titel

Effiziente und robuste Algorithmen
zur punktbasierten 3-D Rekonstruktion
aus Bildsequenzen

A.2 Kurzbeschreibung

Die vorliegende Arbeit befasst sich mit Algorithmen zur Rekonstruktion der Struktur der Szene und der Kamerabewegung aus einer Eingabebildsequenz. Insbesondere beschränken wir uns auf Algorithmen zur punktbasierten 3-D Rekonstruktion, welche die Struktur der Szene mit einer begrenzten Anzahl von Merkmalspunkten repräsentieren. Unser Hauptaugenmerk liegt dabei auf der Effizienz und der Robustheit der entwickelten Algorithmen, um den Einsatz der Algorithmen in der Augmented-Reality-Ausrüstung eines kognitiven Systems zu ermöglichen. Weiterhin wird die Vielseitigkeit der entwickelten Algorithmen durch die große Auswahl an zusätzlichen Anwendungsbereichen, wie zum Beispiel der Objektverfolgung, der Revisualisierung von Objekten und der Rekonstruktion von Objekten aus fiberskopischen Bildern, demonstriert. In dieser Arbeit teilen wir die entwickelten Algorithmen in drei Arbeitsbereiche ein: Punktverfolgung, Schätzung von Struktur und Bewegung, sowie Registrierung von Punktmengen.

Unser Punktverfolgungssystem basiert auf dem Kanade-Lucas-Tomasi Punktverfolger. Wir stellen mehrere Verbesserungen vor, die seine Effizienz und Robustheit erhöhen, wie zum Beispiel eine effiziente hierarchische Auswahlstrategie für die Merkmalspunkte, eine neue Strategie zur Weitergabe der Schätzergebnisse bei der hierarchischen Translationsschätzung, und die effiziente Integration eines affin linearen Modells zum Ausgleich von Helligkeitsschwankungen. Unsere Experimente beweisen die Robustheit des entwickelten Systems gegenüber starken Helligkeitsschwankungen und schwerer Bewegungsunschärfe. Gleichzeitig erreicht das System beim Verfolgen von 150 Merkmalspunkten eine Verarbeitungsgeschwindigkeit von 60 Bildern pro Sekunde. Außerdem präsentieren wir ein speziell angepasstes

Hochgeschwindigkeits-Punktverfolgungssystem für die Selbstlokalisierung einer Augmented-Reality-Ausrüstung. Dieses System verfolgt auf einem relativ langsamen tragbaren Rechner zehn Merkmalspunkte mit einer Verarbeitungsgeschwindigkeit von 200 Bildern pro Sekunde.

Zusätzlich zu den ermittelten 2-D Positionen der Merkmalspunkte benötigt unser vorgestelltes System zur Schätzung von Struktur und Bewegung die intrinsischen Kameraparameter als Eingabedaten. Das System erreicht seine große Vielseitigkeit durch die Kombination mehrerer effizienter Algorithmen, welche eine erste Rekonstruktion der Szene durchführen, mit einem Bündelausgleichsverfahren zur Optimierung der so erhaltenen Ergebnisse. Seine Robustheit gegenüber Ausreißern wird durch die umfassende Anwendung des "Least Median of Squares"-Ansatzes und den Einsatz von M-Estimatoren sichergestellt. In diesem Arbeitsbereich schlagen wir sowohl die Erweiterung des POSIT Algorithmus mit einem virtuellen Referenzpunkt als auch einen neuen Algorithmus zur Auswahl von Schlüsselbildern vor. In unseren Experimenten bewältigt das vorgeschlagene System Eingabedaten mit einem Ausreißeranteil von bis zu 40%. Dabei ist es auch in der Lage, eine Verarbeitungsgeschwindigkeit von 50 Bildern pro Sekunde zu erreichen.

Unsere Arbeit auf dem Gebiet der Registrierung von Punktmengen basiert auf dem lokal konvergenten ICP Algorithmus. Wir vergleichen drei unterschiedliche Ansätze zur Erhöhung der Robustheit dieses Algorithmus gegenüber Ausreißern. Unsere Experimente zeigen, dass der "Least Fractional Squares"-Ansatz das beste Gesamtergebnis liefert. Weiterhin schlagen wir eine Erweiterung zur integrierten Schätzung des Skalierungsfaktors zwischen zwei Punktmengen vor. Wenn sich die initiale Bewegung zwischen den beiden Punktmengen innerhalb des Konvergenzradius des ICP Algorithmus befindet, erlaubt diese Erweiterung eine genaue Registrierung zweier unterschiedlich skalierter Punktmengen.

Bibliography

[Aan02] H. Aanaes, R. Fisker, K. Astrom, and J. M. Carstensen. Robust Factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1215–1225, September 2002.

[Agu03] P. M. Q. Aguiar and J. M. F. Moura. Rank 1 Weighted Factorization for 3D Structure Recovery: Algorithms and Performance Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1134–1149, 2003.

[Ana02] P. Anandan and M. Irani. Factorization with Uncertainty. *International Journal of Computer Vision*, 49(2-3):101–116, 2002.

[Ans03] A. Ansar and K. Daniilidis. Linear Pose Estimation from Points or Lines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):578–589, 2003.

[Bak98] S. Baker, S. K. Nayar, and H. Murase. Parametric feature detection. *International Journal of Computer Vision*, 27(1):27–50, 1998.

[Bak01] S. Baker and I. Matthews. Equivalence and Efficiency of Image Alignment Algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1090–1097, December 2001.

[Bak04] S. Baker and I. Matthews. Lucas-Kanade 20 Years On: A Unifying Framework. *International Journal of Computer Vision*, 56(3):221–255, 2004.

[Bar72] D. I. Barnea and H. F. Silverman. A Class of Algorithms for Fast Digital Image Registration. *IEEE Transactions on Computers*, 21(2):179–186, 1972.

[Bar94] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of Optical Flow Techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[Bar03] A. Bartoli. Towards Gauge Invariant Bundle Adjustment: A Solution Based on Gauge Dependent Damping. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 760–765, Nice, France, October 2003.

[Bar05] A. Bartoli and P. Sturm. Structure-from-Motion Using Lines: Representation, Triangulation, and Bundle Adjustment. *Computer Vision and Image Understanding*, 100(3):416–441, December 2005.

[Bat07] D. Batra, B. Nabbe, and M. Hebert. An Alternative Formulation for Five Point Relative Pose Problem. In *Proceedings of the IEEE Workshop on Motion and Video Computing*, Austin, USA, February 2007.

[Ben04] G. H. Bendels, P. Degener, R. Wahl, M. KÃűrtgen, and R. Klein. Image-Based Registration of 3D-Range Data Using Feature Surface Elements. In *International Symposium on Virtual Reality, Archaeology and Cultural Heritage*, pages 115–124, Brussels, Belgium, December 2004.

[Bes92] P. J. Besl and N. D. McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[Bis96] E. M. Bispo and R. B. Fisher. Free-Form Surface Matching for Surface Inspection. In *The Mathematics of Surfaces VI*, pages 119–136. Clarendon Press, 1996.

[Bla95] G. Blais and M. D. Levine. Registering Multiview Range Data to Create 3D Computer Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):820–824, 1995.

[Bla98] M. J. Black and A. D. Jepson. EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

[Bou95] J.-Y. Bouget and P. Perona. Visual Navigation Using a Single Camera. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 645–652, Boston, USA, June 1995.

[Bou00] J.-Y. Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker. Technical report, Intel Corporation, Microprocessor Research Labs, 2000.

[Bre98] L. Bretzner and T. Lindeberg. Feature Tracking with Automatic Selection of Spatial Scales. *Computer Vision and Image Understanding*, 71(3):385–392, 1998.

[Bru05] A. Bruhn, J. Weikert, and C. Schnörr. Lucas/Kanade Meets Horn/Schunck: Combining Local and Global Optic Flow Methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.

[Buc06] A. M. Buchanan and A. W. Fitzgibbon. Interactive Feature Tracking Using K-D Trees and Dynamic Programming. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 626–633, New York, USA, June 2006.

[Bue04] J. M. Buenaposada, E. Munoz, and L. Baumela. Efficient Appearance-Based Tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition - Workshop on Articulated and Nonrigid Motion*, volume 1, 2004.

[Bur05] D. Burschka, M. Li, R. Taylor, G. D. Hager, and M. Ishii. Scale-Invariant Registration of Monocular Endoscopic Images to CT-Scans for Sinus Surgery. *Medical Image Analysis*, 9(5):413–439, 2005.

[Car02] R. L. Carceroni and K. N. Kutulakos. Multi-View Scene Capture by Surfel Sampling: From Video Streams to Non-Rigid 3D Motion, Shape and Reflectance. *International Journal of Computer Vision*, 49(2-3):175–214, 2002.

[Cha02] P. Chang and M. Hebert. Robust Tracking and Structure from Motion through Sampling Based Uncertainty Representation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 3030–3037, Washington, USA, May 2002.

[Che92] Y. Chen and G. Medioni. Object Modelling by Registration of Multiple Range Images. *Image and Vision Computing*, 10(3):145–155, 1992.

[Che99a] C.-S. Chen, Y.-P. Hung, and J.-B. Cheng. RANSAC-Based DARCES: A New Approach to Fast Automatic Registration of Partially Overlapping Range Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1229–1234, 1999.

[Che99b] D. Chetverikov and J. Verestóy. Feature Point Tracking for Incomplete Trajectories. *Computing*, 62(4):321–328, 1999.

[Che05] D. Chetverikov, D. Stepanov, and P. Krsek. Robust Euclidean Alignment of 3D Point Sets: The Trimmed Iterative Closest Point Algorithm. *Image and Vision Computing*, 23(3):299–309, 2005.

[Cho03] W. Chojnacki, M. J. Brooks, A. van den Hengel, and D. Gawley. Revisiting Hartley's Normalised Eight-Point Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1172–1177, 2003.

[Chr96] S. Christy and R. Horaud. Euclidean shape and motion from multiple perspective views by affine iterations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(11):1098–1104, 1996.

[Chr11] H. I. Christensen. CogVis: Cognitive Vision Systems. `http://cogvis.nada.kth.se`, January 2011. Information Society Technologies IST-2000-29375.

[Chu98] D. H. Chung, I. D. Yun, and S. U. Lee. Registration of Multiple-Range Views Using the Reverse-Calibration Technique. *Pattern Recognition*, 31(4):457–464, 1998.

[Cox95] I. J. Cox, S. Roy, and S. L. Hingorani. Dynamic Histogram Warping of Image Pairs for Constant Image Brightness. In *Proceedings of the IEEE International Conference on Image Processing*, 1995.

[Csu11] G. Csurka. LAVA: Learning for Adaptable Visual Assistants. `http://web.archive.org/web/20061102041838/http://www.l-a-v-a.org/`, January 2011. Information Society Technologies IST-2001-34405.

[Dav95] C. Q. Davis, Z. Z. Karu, and D. M. Freeman. Equivalence of Subpixel Motion Estimators Based on Optical Flow and Block Matching. In *Proceedings of the IEEE International Symposium on Computer Vision*, pages 7–12, Coral Gables, USA, November 1995.

[Dav07] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, June 2007.

[DeM95] D. F. DeMenthon and L. S. Davis. Model-Based Object Pose in 25 Lines of Code. *International Journal of Computer Vision*, 15(1-2):123–141, June 1995.

[Deu05] B. Deutsch, C. Gräßl, F. Bajramovic, and J. Denzler. A Comparative Evaluation of Template and Histogram Based 2D Tracking Algorithms. In *Pattern Recognition, 27th DAGM Symposium*, pages 269–276, Wien, Austria, 2005.

[Dou06] L. Douadi, M.-J. Aldon, and A. Crosnier. Pair-Wise Registration of 3D/Color Data Sets with ICP. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 663–668, Beijing, China, October 2006.

[Du07] S. Du, N. Zheng, S. Ying, Q. You, and Y. Wu. An Extension of the ICP Algorithm Considering Scale Factor. In *Proceedings of the IEEE International Conference on Image Processing*, volume 5, pages 193–196, San Antonio, USA, September 2007.

[Egg97] D. W. Eggert, A. Lorusso, and R. B. Fisher. Estimating 3-D Rigid Body Transformations: A Comparison of Four Major Algorithms. *Machine Vision and Applications*, 9:272–290, 1997.

[Eng06] C. Engels, H. Stewenius, and D. Niste. Bundle Adjustment Rules. In *Proceedings of the ISPRS Symposium of Commission III - Photogrammetric Computer Vision*, pages 266–271, Bonn, Germany, September 2006.

[Fau92] O. D. Faugeras, Q.-T. Luong, and S. Maybank. Camera Self-Calibration: Theory and Experiments. In *Proceedings of the European Conference on Computer Vision*, pages 321–334, Santa Margherita Ligure, Italy, May 1992.

[Fau93] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. The MIT Press, Cambridge, USA, 1993.

[Fio01] P. D. Fiore. Efficient Linear Solution of Exterior Orientation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):140–148, February 2001.

[Fit98] A. W. Fitzgibbon and A. Zisserman. Automatic Camera Recovery for Closed or Open Image Sequences. In *Proceedings of the European Conference on Computer Vision*, pages 311–326, Freiburg, Germany, June 1998.

[Fit03] A. W. Fitzgibbon. Robust Registration of 2D and 3D Point Sets. *Image and Vision Computing*, 21(13-14):1145–1153, 2003.

[För86] W. Förstner and A. Pertl. Photogrammetric Standard Methods and Digital Image Matching Techniques for High Precision Surface Measurements. *Pattern Recognition in Practice II*, pages 57–72, 1986.

[For95] J .A. Ford. Improved Algorithms of Illinois-type for the Numerical Solution of Nonlinear Equations. Technical Report CSM-257, University of Essex, 1995.

[For03] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Pearson Education, Upper Saddle River, USA, 2003.

[Fus99] A. Fusiello, E. Trucco, T. Tommasini, and V. Roberto. Improving Feature Tracking with Robust Statistics. *Pattern Analysis and Applications*, 2(4):312–320, 1999.

[Fus00] A. Fusiello. Uncalibrated Euclidean Reconstruction: A Review. *Image and Vision Computing*, 18:555–563, 2000.

[Gel05] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann. Robust Global Registration. In *Proceedings of the Eurographics Symposium on Geometry Processing*, pages 197–206, Vienna, Austria, July 2005.

[Geo04] B. Georgescu and P. Meer. Point Matching Under Large Image Deformations and Illumination Changes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:674–688, 2004.

[Gib02] S. Gibson, J. Cook, T. Howard, R. Hubbold, and D. Oram. Accurate Camera Calibration for Off-line, Video-Based Augmented Reality. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, pages 37–46, Darmstadt, Germany, 2002.

[Gon01] R. C. Gonzales and R. E. Woods. *Digital Image Processing*. Prentice-Hall, Upper Saddle River, USA, 2nd edition, 2001.

[Gor04] N. Gorges, M. Hanheide, W. Christmas, C. Bauckhage, G. Sagerer, and J. Kittler. Mosaics from Arbitrary Stereo Video Sequences. In *Pattern Recognition, 26th DAGM Symposium*, Lecture Notes in Computer Science, pages 342–349, Tübingen, Germany, 2004. Springer-Verlag.

[Grä05] C. Gräßl, T. Zinßer, I. Scholz, and H. Niemann. 3-D Object Tracking with the Adaptive Hyperplane Approach Using SIFT Models for Initialization. In *Proceedings of the IAPR Conference on Machine Vision Applications*, pages 5–8, Tsukuba Science City, Japan, 2005.

[Gue02] R. F. C. Guerreiro and P. M. Q. Aguiar. 3D Structure from Video Streams with Partially Overlapping Images. In *Proceedings of the IEEE International Conference on Image Processing*, volume 3, pages 897–900, Rochester, USA, 2002.

[Gun05] B. K. Gunturk, J. Glotzbach, Y. Altunbasak, R. W. Schafer, and R. M. Mersereau. Demosaicking: Color Filter Array Interpolation in Single Chip Digital Cameras. *IEEE Signal Processing Magazine (Special Issue on Color Image Processing)*, 22(1):44–54, January 2005.

[Gup97] N. Gupta and L. Kanal. Gradient Based Image Motion Estimation Without Computing Gradients. *International Journal of Computer Vision*, 22(1):81–101, 1997.

[Hag98] G. D. Hager and P. N. Belhumeur. Efficient Region Tracking with Parametric Models of Geometry and Illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.

[Han03] M. Han and T. Kanade. Multiple Motion Scene Reconstruction from Uncalibrated Views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):884–894, July 2003.

[Har88] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Proceedings of the Fourth Alvey Vision Conference*, pages 147–151, Manchester, UK, 1988.

[Har92] R. Hartley. Estimation of Relative Camera Positions for Uncalibrated Cameras. In *Proceedings of the European Conference on Computer Vision*, pages 579–587, Santa Margherita Ligure, Italy, May 1992.

[Har93] R. Hartley. Euclidean Reconstruction from Uncalibrated Views. In *Proceedings of the Joint European - US Workshop on Applications of Invariance in Computer Vision*, volume 825 of *Lecture Notes In Computer Science*, pages 237–256, Azores, Portugal, October 1993.

[Har94a] R. M. Haralick, C.-N. Lee, K. Ottenberg, and M. Nölle. Review and Analysis of Solutions of the Three Point Perspective Pose Estimation Problem. *International Journal of Computer Vision*, 13(3):331–356, 1994.

[Har94b] R. Hartley. Projective Reconstruction and Invariants from Multiple Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(10):1036–1041, October 1994.

[Har97a] R. Hartley. In Defense of the Eight-Point Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, June 1997.

[Har97b] R. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, November 1997.

[Har00] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, 2000.

[Har05] R. Hartley and S. B. Kang. Parameter-free Radial Distortion Correction with Centre of Distortion Estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 1834–1841, Beijing, China, October 2005.

[Hei99] B. Heigl, D. Paulus, and H. Niemann. Tracking Points in Sequences of Color Images. In *Pattern Recognition and Image Understanding*, pages 70–77, Herrsching, Germany, 1999. Infix, Sankt Augustin.

[Hey97] A. Heyden and K. Astrom. Euclidean Reconstruction from Image Sequences with Varying and Unknown Focal Lenght and Principal Point. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 438–443, Puerto Rico, 1997.

[Hey99] A. Heyden, R. Berthilsson, and G. Sparr. An Iterative Factorization Method for Projective Structure and Motion from Image Sequences. *Image and Vision Computing*, 17(13):981–991, 1999.

[Hir05] K. Hirakawa and T. W. Parks. Adaptive Homogeneity-Directed Demosaicing Algorithm. *IEEE Transactions on Image Processing*, 14(3):360–369, March 2005.

[Hor81] B. K. P. Horn and B. G. Schunck. Determining Optical Flow. *Artificial Intelligence*, 17:185–203, 1981.

[Hor99] J. Hornegger and C. Tomasi. Representation Issues in the ML Estimation of Camera Motion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 914–919, Corfu, Greece, 1999.

[Hub03] D. Huber and M. Hebert. Fully Automatic Registration of Multiple 3-D Data Sets. *Image and Vision Computing*, 21(7):637–650, 2003.

[Huy03] D. Q. Huynh, R. Hartley, and A. Heyden. Outlier Correction in Image Sequences for the Affine Camera. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 585–590, Nice, France, October 2003.

[Jin01] H. Jin, P. Favaro, and S. Soatto. Real-Time Feature Tracking and Outlier Rejection with Changes in Illumination. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 684–689, Vancouver, Canada, July 2001.

[Jur02] F. Jurie and M. Dhome. Hyperplane Approximation for Template Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):996–1000, 2002.

[Kan94] T. Kanade and M. Okutomi. A Stereo Matching Algorithm With an Adaptive Window: Theory and Experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):920–932, 1994.

[Kan97] S. B. Kang, R. Szeliski, and H.-Y. Shum. A Parallel Feature Tracker for Extended Image Sequences. *Computer Vision and Image Understanding*, 67(3):296–310, September 1997.

[Kan98] T. Kanade and D. D. Morris. Factorization Methods for Structure from Motion. *Philosophical Transactions of the Royal Society*, 356(1740):1153–1173, 1998.

[Ke05] Q. Ke and T. Kanade. Quasiconvex Optimization for Robust Geometric Reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 986–993, Boston, USA, June 2005.

[Ke06] Q. Ke and T. Kanade. Uncertainty Models in Quasiconvex Optimization for Geometric Reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1199–1205, New York, USA, June 2006.

[Ken03] C. Kenney, B. Manjunath, M. Zuliani, G. Hewer, and A. Van Nevel. A Condition Number for Point Matching with Application to Registration and Postregistration Error Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11):1437–1454, November 2003.

[Koc99] R. Koch, M. Pollefeys, B. Heigl, L. Van Gool, and H. Niemann. Calibration of Hand-Held Camera Sequences for Plenoptic Modeling. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 585–591, Corfu, Greece, September 1999.

[Las98] J. Lasenby, W. J. Fitzgerald, A. N. Lasenby, and C. J. L. Doran. New Geometric Methods for Computer Vision: An Application to Structure and Motion Estimation. *International Journal of Computer Vision*, 26(3):191–213, 1998.

[LH81] H. C. Longuet-Higgins. A Computer Algorithm for Reconstructing a Scene From Two Projections. *Nature*, 293:133–135, 1981.

[Li06] H. Li and R. Hartley. Five-Point Motion Estimation Made Easy. In *Proceedings of the IEEE International Conference on Pattern Recognition*, pages 630–633, Hong Kong, 2006.

[Li07] H. Li and R. Hartley. The 3D-3D Registration Problem Revisited. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–8, Rio de Janeiro, Brazil, 2007.

[Liu05] B. Liu, M. Yu, D. Maier, and R. Männer. An Efficient and Accurate Method for 3D-Point Reconstruction from Multiple Views. *International Journal of Computer Vision*, 65(3):175–188, 2005.

[Lom06] E. Lomonosov, D. Chetverikov, and A. Ekart. Pre-registration of Arbitrarily Oriented 3D Surfaces Using a Genetic Algorithm. *Pattern Recognition Letters*, 27(11):1201–1208, 2006.

[Lou05a] M. I. A. Lourakis and A. A. Argyros. Efficient, Causal Camera Tracking in Unprepared Environments. *Computer Vision and Image Understanding*, 99(2):259–290, 2005.

[Lou05b] M. I. A. Lourakis and A. A. Argyros. Is Levenberg-Marquardt the Most Efficient Optimization Algorithm for Implementing Bundle Adjustment? In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 1526–1531, Beijing, China, 2005.

[Low04] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[Lu02] C.-P. Lu, G. D. Hager, and E. Mjolsness. Fast and Globally Convergent Pose Estimation from Video Images,. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):610–622, 2002.

[Lu04] X. Lu, D. Colbry, and A. K. Jain. Three-Dimensional Model Based Face Recognition. In *Proceedings of the IEEE International Conference on Pattern Recognition*, pages 362–366, Cambridge, UK, August 2004.

[Luc81] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 674–679, Vancouver, Canada, April 1981.

[Mah01] S. Mahamud, M. Hebert, Y. Omori, and J. Ponce. Provably-Convergent Iterative Methods for Projective Structure from Motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1018–1025, Kauai, USA, December 2001.

[Mal04] H. S. Malvar, L. He, and R. Cutler. High-quality Linear Interpolation for Demosaicing of Bayer-patterned Color Images. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 485–488, Montreal, Canada, May 2004.

[Mas95] T. Masuda and N. Yokoya. A Robust Method for Registration and Segmentation of Multiple Range Images. *Computer Vision and Image Understanding*, 10(3):295–307, 1995.

[Mat04] I. Matthews, T. Ishikawa, and S. Baker. The Template Update Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):810–815, 2004.

[McC02] B. McCane, B. Gavin, and K. Novins. Algorithmic Fusion for More Robust Feature Tracking. *International Journal of Computer Vision*, 49(1):79–89, 2002.

[McL95] P. McLauchlan and D. Murray. A Unifying Framework for Structure and Motion Recovery from Image Sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 314–320, Cambridge, USA, June 1995.

[McL00] P. McLauchlan. A Batch/Recursive Algorithm for 3d Scene Reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 738–743, Hilton Head, USA, 2000.

[Mei01] E. H. W. Meijering, W. J. Niessen, and M. A. Viergever. Quantitative Evaluation of Convolution-Based Methods for Medical Image Interpolation. *Medical Image Analysis*, 5(2):111–126, June 2001.

[Mik04] K. Mikolajczyk and C. Schmid. Scale & Affine Invariant Interest Point Detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[Mik05a] K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, October 2005.

[Mik05b] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A Comparison of Affine Region Detectors. *International Journal of Computer Vision*, 65:43–72, 2005.

[Mit96] A. Mitiche and P. Bouthemy. Computation and Analysis of Image Motion: A Synopsis of Current Problems and Methods. *International Journal of Computer Vision*, 19(1):29–55, 1996.

[MN07] F. Moreno-Noguer, P. Fua, and V. Lepetit. Accurate Non-Iterative O(n) Solution to the PnP Problem. In *Proceedings of the IEEE International Conference on Computer Vision*, Rio de Janeiro, Brazil, October 2007.

[Mor80] H. P. Moravec. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. Ph. d. thesis, Stanford University, Stanford, USA, 1980.

[Mor97] T. Morita and T. Kanade. A Sequential Factorization Method for Recovering Shape and Motion From Image Streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(8):858–867, August 1997.

[Mue04] U. Muehlmann, M. Ribo, P. Lang, and A. Pinz. A New High Speed CMOS Camera for Real-Time Tracking Applications. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 5195–5200, New Orleans, USA, April 2004.

[Mur01] V. Murino, A. Fusiello, U. Castellani, and L. Ronchetti. Reconstruction of Complex Environments by Robust Pre-aligned ICP. In *Proceedings of the International Conference on 3-D Digital Imaging and Modeling*, pages 187–194, Quebec City, Canada, May 2001.

[Nag04] H.-H. Nagel. Steps Towards a Cognitive Vision System. *AI Magazine*, 25(2):31–50, 2004.

[Neu97] P. J. Neugebauer. Geometrical Cloning of 3D Objects via Simultaneous Registration of Multiple Range Images. In *Proceedings of the International Conference on Shape Modeling and Applications*, Aizu-Wakamatsu, Japan, March 1997.

[Nic02] K. Nickels and S. Hutchinson. Estimating Uncertainty in SSD-Based Feature Tracking. *Image and Vision Computing*, 20:47–58, 2002.

[Nie05] H. Niemann and I. Scholz. Evaluating the Quality of Light Fields Computed from Hand-held Camera Images. *Pattern Recognition Letters*, 26:239–249, 2005.

[Nis00] D. Nister. Reconstruction from Uncalibrated Sequences with a Hierarchy of Trifocal Tensors. In *Proceedings of the European Conference on Computer Vision*, pages 649–663, June 2000.

[Nis04a] D. Nister. A Minimal Solution to the Generalised 3-point Pose Problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 560–567, Washington, USA, June 2004.

[Nis04b] D. Nister. An Efficient Solution to the Five-Point Relative Pose Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, June 2004.

[Nis04c] D. Nister, O. Naroditsky, and J. Bergen. Visual Odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 652–659, Washington, USA, June 2004.

[Nis06] D. Nister, O. Naroditsky, and J. Bergen. Visual Odometry for Ground Vehicle Applications. *Journal of Field Robotics*, 23(1):3–20, January 2006.

[Oli99a] J. Oliensis. A Multi-Frame Structure-from-Motion Algorithm under Perspective Projection. *International Journal of Computer Vision*, 34(2-3):163–192, 1999.

[Oli99b] J. Oliensis. Fast and Accurate Self-Calibration. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 745–752, Corfu, Greece, September 1999.

[Oli00] J. Oliensis. A Critique of Structure-from-Motion Algorithms. *Computer Vision and Image Understanding*, 80(2):172–214, November 2000.

[Oli01] J. Oliensis and Y. Genc. Fast and Accurate Algorithms for Projective Multi-Image Structure from Motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):546–559, 2001.

[Oli02] J. Oliensis. Exact Two-Image Structure from Motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1618–1633, 2002.

[Oli07] J. Oliensis and R. Hartley. Iterative Extensions of the Sturm/Triggs Algorithm: Convergence and Nonconvergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2217–2233, 2007.

[Phi07] J. M. Phillips, R. Liu, and C. Tomasi. Outlier Robust ICP for Minimizing Fractional RMSD. In *Proceedings of the International Conference on 3-D Digital Imaging and Modeling*, pages 427–434, Montreal, Canada, August 2007.

[Poe94] C. Poelman and T. Kanade. A Paraperspective Factorization Method for Shape and Motion Recovery. In *Proceedings of the European Conference on Computer Vision*, pages 97–108, Stockholm, Sweden, 1994.

[Pol99a] M. Pollefeys and L. Van Gool. Stratified Self-Calibration with the Modulus Constraint. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):707–724, 1999.

[Pol99b] M. Pollefeys, R. Koch, and L. Van Gool. Self-Calibration and Metric Reconstruction in Spite of Varying and Unknown Internal Camera Parameters. *International Journal of Computer Vision*, 32(1):7–25, 1999.

[Pol02] M. Pollefeys, F. Verbiest, and L. Van Gool. Surviving Dominant Planes in Uncalibrated Structure and Motion Recovery. In *Proceedings of the European Conference on Computer Vision*, pages 837–851, Copenhagen, Denmark, May 2002.

[Pol04] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual Modeling with a Hand-Held Camera. *International Journal of Computer Vision*, 59(3):207–232, 2004.

[Pot04] H. Pottmann, S. Leopoldseder, and M. Hofer. Registration without ICP. *Computer Vision and Image Understanding*, 95(1):54–71, 2004.

[Poy98] C. Poynton. The Rehabilitation of Gamma. In *Proceedings of the SPIE Conference on Human Vision and Electronic Imaging*, volume 3299, pages 232–249, San Jose, USA, January 1998.

[Pul99] K. Pulli. Multiview Registration for Large Data Sets. In *Proceedings of the International Conference on 3D Digital Imaging and Modeling*, pages 160–168, Ottawa, Canada, October 1999.

[Qua99] L. Quan and Z. Lan. Linear N-Point Camera Pose Determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):774–780, August 1999.

[Ram06] S. Ramalingam, S. K. Lodha, and P. Sturm. A Generic Structure-from-Motion Framework. *Computer Vision and Image Understanding*, 103(3):218–228, September 2006.

[Rep05] J. Repko and M. Pollefeys. 3D Models from Extended Uncalibrated Video Sequences: Addressing Key-Frame Selection and Projective Drift. In *Proceedings of the International Conference on 3-D Digital Imaging and Modeling*, pages 150–157, Ottawa, Canada, June 2005.

[Rib02] M. Ribo, H. Ganster, M. Brandner, P. Lang, C. Stock, and A. Pinz. Hybrid Tracking for Outdoor AR Applications. *IEEE Computer Graphics and Applications Magazine*, 22(6):54–63, 2002.

[Rob04] D. Robinson and P. Milanfar. Fundamental Performance Limits in Image Registration. *IEEE Transactions on Image Processing*, 13(9):1185–1199, 2004.

[Ros06] E. Rosten and T. Drummond. Machine Learning for High-Speed Corner Detection. In *Proceedings of the European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

[Rou87] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. Wiley, New York, 1987.

[Rus01] S. Rusinkiewicz and M. Levoy. Efficient Variants of the ICP Algorithm. In *Proceedings of the International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, Quebec City, Canada, May 2001.

[Sag11] G. Sagerer, S. Wachsmuth, M. Hanheide, and S. Wrede. VAMPIRE: Visual Active Memory Processes and Interactive Retrieval. `http://www.vampire-project.org`, January 2011. Information Society Technologies IST-2001-34401.

[Sai03] M. Sainz, A. Susin, and N. Bagherzadeh. Camera Calibration of Long Image Sequences with the Presence of Occlusions. In *Proceedings of the IEEE International Conference on Image Processing*, volume 1, pages 317–320, Barcelona, Spain, 2003.

[Sal07] J. Salvi, C. Matabosch, D. Fofi, and J. Forest. A Review of Recent Range Image Registration Methods with Accuracy Evaluation. *Image and Vision Computing*, 25(5):578–596, May 2007.

[Sax07] A. Saxena, M. Sun, and A. Y. Ng. 3-D Reconstruction from Sparse Views using Monocular Vision. In *ICCV Workshop on Virtual Representations and Modeling of Large-Scale Environments*, pages 1–8, Rio de Janeiro, Brazil, 2007.

[Sch00] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of Interest Point Detector. *International Journal of Computer Vision*, 37(2):151–172, 2000.

[Sch04] I. Scholz, J. Denzler, and H. Niemann. Calibration of Real Scenes for the Reconstruction of Dynamic Light Fields. *IEICE Transactions on Information & Systems*, E87-D(1):42–49, 2004.

[Sch05] I. Scholz, C. Vogelgsang, J. Denzler, and H. Niemann. Dynamic Light Field Reconstruction and Rendering for Multiple Moving Objects. In *Proceedings of the Ninth IAPR Conference on Machine Vision Applications*, pages 184–188, Tokyo, Japan, 2005.

[Sch08a] I. Scholz. *Reconstruction and Modeling of Static and Dynamic Light Fields*, volume 26 of *Studien zur Mustererkennung*. Logos Verlag, Berlin, 2008.

[Sch08b] G. Schweighofer, S. Segvic, and A. Pinz. Online/Realtime Structure and Motion for General Camera Models. In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, pages 1–6, Copper Mountain, USA, January 2008.

[Scl98] S. Sclaroff and J. Isidoro. Active Blobs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1146–1153, Bombay, India, 1998.

[Seg07] S. Segvic, G. Schweighofer, and A. Pinz. Performance Evaluation of the Five-Point Relative Pose with Emphasis on Planar Scenes. In *Proceedings of the Workshop of the Austrian Association for Pattern Recognition*, pages 33–40, Schloss Krumbach, Austria, May 2007.

[Sha02] G. C. Sharp, S. W. Lee, and D. K. Wehe. ICP Registration using Invariant Features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):90–102, 2002.

[Sha05] K. Shafique and M. Shah. A Non-Iterative Greedy Algorithm for Multi-Frame Point Correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):51–65, 2005.

[Sha08] G. C. Sharp, S. W. Lee, and D. K. Wehe. Maximum-Likelihood Registration of Range Images with Missing Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1):120–130, January 2008.

[Shi94] J. Shi and C. Tomasi. Good Features to Track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, Seattle, USA, 1994.

[Shu99] H. Shum, Q. Ke, and Z. Zhang. Efficient Bundle Adjustment with Virtual Key Frames: A Hierarchical Approach to Multi-Frame Structure from Motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2538–2543, Ft. Collins, USA, June 1999.

[Sil05] L. Silva, O. R. P. Bellon, and K. L. Boyer. Precision Range Image Registration Using a Robust Surface Interpenetration Measure and Enhanced Genetic Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):762–776, May 2005.

[Siv06] J. Sivic, F. Schaffalitzky, and A. Zisserman. Object Level Grouping for Video Shots. *International Journal of Computer Vision*, 67(2):189–210, April 2006.

[Smi97] S. M. Smith and J. M. Brady. SUSAN - A New Approach to Low Level Image Processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[Soa98a] S. Soatto and P. Perona. Reducing "Structure From Motion": A General Framework for Dynamic Vision Part 1: Modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(9):933–942, 1998.

[Soa98b] S. Soatto and P. Perona. Reducing "Structure From Motion": A General Framework for Dynamic Vision Part 2: Implementation and Experimental Assessment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(9):943–960, 1998.

[Sta11] The Stanford 3D Scanning Repository. `http://graphics.stanford.edu/data/3Dscanrep`, January 2011.

[Ste99] C. V. Stewart. Robust Parameter Estimation in Computer Vision. *SIAM Review*, 41(3):513–537, 1999.

[Ste03] C. V. Stewart, C.-L. Tsai, and B. Roysam. The Dual-Bootstrap Iterative Closest Point Algorithm with Application to Retinal Image Registration. *IEEE Transactions on Medical Imaging*, 22(11):1379–1394, 2003.

[Ste05] H. Stewenius, F. Schaffalitzky, and D. Nister. How Hard is Three-View Triangulation Really? In *Proceedings of the IEEE International Conference on Computer Vision*, pages 686–693, Beijing, China, October 2005.

[Ste06] H. Stewenius, C. Engels, and D. Nister. Recent Developments on Direct Relative Orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(4):284–294, 2006.

[Stu96] P. Sturm and B. Triggs. A Factorization Based Algorithm for Multi-Image Projective Structure and Motion. In *Proceedings of the European Conference on Computer Vision*, pages 709–720, Cambridge, UK, April 1996.

[Sug04] Y. Sugaya and K. Kanatani. Extending Interrupted Feature Point Tracking for 3-D Affine Reconstruction. In *Proceedings of the European Conference on Computer Vision*, volume 1, pages 310–321, Prague, Czech Republic, May 2004.

[Süh02] M. Sühling, M. Arigovindan, P. Hunziker, and M. Unser. Motion Analysis of Echocardiograms Using a Local-Affine, Spatio-Temporal Model. In *Proceedings of the IEEE International Symposium on Biomedical Imaging: Macro to Nano*, volume 2, pages 573–576, Washington, USA, July 2002.

[Sze98] R. Szeliski and P. H. S. Torr. Geometrically Constrained Structure from Motion: Points on Planes. In *Proceedings of the European Workshop on 3D Structure from Multiple Images of Large-Scale Environments*, pages 171–186, Freiburg, Germany, June 1998.

[Tho04] T. Thormaehlen, H. Broszio, and A. Weissenfeld. Keyframe Selection for Camera Motion and Structure Estimation from Multiple Views. In *Proceedings of the European Conference on Computer Vision*, volume 1, pages 523–535, Prague, Czech Republic, May 2004.

[Tis04] P. Tissainayagam and D. Suter. Assessing the Performance of Corner Detectors for Point Feature Tracking Applications. *Image and Vision Computing*, 22:663–679, 2004.

[Toi02] T. Toivonen, J. Heikkilä, and O. Silven. A New Algorithm for Fast Full Search Block Motion Estimation Based on Number Theoretic Transforms. In *Proceedings of the International Workshop on Systems, Signals and Image Processing*, pages 90–94, Manchester, United Kingdom, November 2002.

[Tom91] C. Tomasi and T. Kanade. Detection and Tracking of Point Features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[Tom92] C. Tomasi and T. Kanade. Shape and Motion from Image Streams Under Orthography: A Factorization Method. *International Journal of Computer Vision*, 9(2):137–154, November 1992.

[Tön05] K. D. Tönnies. *Grundlagen der Bildverarbeitung*. Pearson Education, München, Germany, 2005.

[Tor97] P. H. S. Torr and D. W. Murray. The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix. *International Journal of Computer Vision*, 24(3):271–300, September 1997.

[Tor04] L. Torresani and A. Hertzmann. Automatic Non-rigid 3D Modeling from Video. In *Proceedings of the European Conference on Computer Vision*, volume 2, pages 299–312, Prague, Czech Republic, May 2004.

[Toy99] K. Toyama and G. Hager. Incremental Focus of Attention for Robust Vision-Based Tracking. *International Journal of Computer Vision*, 35(1):45–63, November 1999.

[Tri99] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment — A modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, pages 298–372, Corfu, Greece, September 1999.

[Tru98] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice-Hall, Upper Saddle River, USA, 1998.

[Tru99] E. Trucco, A. Fusiello, and V. Roberto. Robust Motion and Correspondence of Noisy 3-D Point Sets with Missing Data. *Pattern Recognition Letters*, 20(8):889–898, 1999.

[Tso98] J. K. Tsotsos, G. Verghese, S. Dickinson, M. Jenkin, A. Jepson, E. Milios, F. Nuflo, S. Stevenson, M. Black, D. Metaxas, S. Culhane, Y. Ye, and R. Mann. PLAYBOT: A Visually-Guided Robot to Assist Physically Disabled Children in Play. *Image and Vision Computing Journal, Special Issue on Vision for the Disabled*, 16:275–292, April 1998.

[Tur94] G. Turk and M. Levoy. Zippered Polygon Meshes from Range Images. In *Proceedings of the ACM SIGGRAPH*, pages 311–318, Orlando, USA, July 1994.

[Ved07] A. Vedaldi, G. Guidi, and S. Soatto. Moving Forward in Structure From Motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7, Minneapolis, USA, June 2007.

[Vee01] C. J. Veenman, M. J. T. Reinders, and E. Backer. Resolving Motion Correspondence for Densely Moving Points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):54–72, 2001.

[Vog04] F. Vogt, S. Krüger, T. Zinßer, T. Maier, H. Niemann, W. Hohenberger, and C. Schick. Fusion von Lichtfeldern und CT-Daten für minimal-invasive Operationen. In *8. Workshop Bildverarbeitung für die Medizin*, pages 309–313, Erlangen, Germany, 2004. Springer-Verlag.

[Vog06] F. Vogt. *Augmented Light Field Visualization and Real-Time Image Enhancement for Computer Assisted Endoscopic Surgery*. Der Andere Verlag, Tönning, Germany, 2006.

[Wai04] A. Waibel, H. Steusloff, R. Stiefelhagen, et al. CHIL - Computers in the Human Interaction Loop. In *5th International Workshop on Image Analysis for Multimedia Interactive Services*, Lisbon, Portugal, April 2004.

[Wei97] S. Weik. Registration of 3-D Partial Surface Models Using Luminance and Depth Information. In *Proceedings of the International Conference on 3-D Digital Imaging and Modeling*, pages 93–100, May 1997.

[Whi01] A. Whitehead and G. Roth. Salient Frame Extraction for Structure from Motion. In *Proceedings of the International Conference on Visualization, Imaging and Image Processing*, pages 658–663, Marbella, Spain, September 2001.

[Wie02] J. Wieghardt, R. P. Würtz, and C. von der Malsburg. Gabor-Based Feature Point Tracking with Automatically Learned Constraints. *Dynamic Perception*, pages 121–126, 2002.

[Win05] C. Winter, I. Scholz, S. Rupp, and T .Wittenberg. Reconstruction of Tubes from Monocular Fiberscopic Images – Application and First Results. In *Vision, Modeling, and Visualization*, pages 57–64, Erlangen, Germany, 2005.

[Xia06] J. Xiao, J. Chai, and T. Kanade. A Closed-Form Solution to Non-Rigid Shape and Motion Recovery. *International Journal of Computer Vision*, 67(2):233–246, 2006.

[Zha96] Z. Zhang. On the Epipolar Geometry Between Two Images with Lens Distortion. In *Proceedings of the IEEE International Conference on Pattern Recognition*, volume 1, pages 407–411, Vienna, Austria, August 1996.

[Zha97] Z. Zhang. Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting. *Image and Vision Computing*, 15:59–76, 1997.

[Zha98] Z. Zhang. Determining the Epipolar Geometry and its Uncertainty: A Review. *International Journal of Computer Vision*, 27(2):161–195, March 1998.

[Zha00] Z. Zhang. A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, November 2000.

[Zha02] Z. Zhang. On the Optimization Criteria Used in Two-view Motion Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7):717–729, 2002.

[Zhe95] Q. Zheng and R. Chellappa. Automatic Feature Point Extraction and Tracking in Image Sequences for Arbitrary Camera Motion. *International Journal of Computer Vision*, 15:31–76, 1995.

[Zin03a] T. Zinßer, J. Schmidt, and H. Niemann. A Refined ICP Algorithm for Robust 3-D Correspondence Estimation. In *Proceedings of the IEEE International Conference on Image Processing*, volume 2, pages 695–698, Barcelona, Spain, 2003.

[Zin03b] T. Zinßer, J. Schmidt, and H. Niemann. Performance Analysis of Nearest Neighbor Algorithms for ICP Registration of 3-D Point Sets. In *Vision, Modeling and Visualization*, pages 199–206, Munich, Germany, 2003.

[Zin04] T. Zinßer, C. Gräßl, and H. Niemann. Efficient Feature Tracking for Long Video Sequences. In *Pattern Recognition, 26th DAGM Symposium*, Lecture Notes in Computer Science, pages 326–333, Tübingen, Germany, 2004. Springer-Verlag.

[Zin05a] T. Zinßer, C. Gräßl, and H. Niemann. High-Speed Feature Point Tracking. In *Vision, Modeling and Visualization*, pages 49–56, Erlangen, Germany, 2005.

[Zin05b] T. Zinßer, J. Schmidt, and H. Niemann. Point Set Registration with Integrated Scale Estimation. In *Proceedings of the International Conference on Pattern Recognition and Image Processing*, pages 116–119, Minsk, Belarus, 2005.

[Ziv04] Z. Zivkovic and F. van der Heijden. Improving the Selection of Feature Points for Tracking. *Pattern Analysis and Applications*, 7(2):144–150, 2004.

Index

absolute camera pose estimation, 88
accuracy, 37, 81, 132
adapted tracking system, 73
affine distortion, 48
albedo, 23
aperture, 27
augmented normal equations, 103

back-projection, 98
baseline, 28
basin of convergence, 37, 132
Bayer pattern, 26
bilinear interpolation, 43
block matching, 39, 70
blooming, 26
breakdown point, 118
brightness, 24
bundle adjustment, 84, 99

camera calibration, 19, 78
camera coordinate system, 14
Cauchy estimator, 123
center of projection, 15
cheirality constraint, 98
cognitive vision system, 1
companion matrix, 90
computational efficiency, 37, 81, 132
coordinate systems, 14
correlation matrix, 91

damping parameter, 103
demosaicing algorithm, 26
depth of field, 20
derivative image, 51
digital image, 12
dog leg method, 104
dynamic histogram warping, 57
dynamic range, 26

eight-point algorithm, 83
endoscopic surgery, 220
epipolar constraint, 28
epipolar geometry, 27
epipolar line, 28
epipolar plane, 28
epipole, 28
essential matrix, 29
Euclidean reconstruction, 78
exposure, 26
extrinsic camera parameters, 18

factorization algorithm, 82
Fair estimator, 123
feature descriptor, 34
feature detection, 44
feature drift problem, 47, 56
feature point, 34
feature point tracking, 3, 34
feature selection, 45, 50
feature trail, 35
feature window, 34, 41
fiberscope, 221
five-point algorithm, 94
focal length, 18
forwards additive algorithm, 53
frame, 13
frame rate, 36
fundamental matrix, 29

gain, 27

gain ratio, 104
gamma correction, 24
gauge freedom, 106
Gaussian image pyramid, 61
geometric image formation, 14
golden section search, 141

hand-eye calibration, 201
Harris detector, 46
hierarchical search structure, 51
hierarchical translation estimation, 61
homogeneous coordinates, 19
Huber estimator, 124
hybrid tracking system, 69

ICP algorithm, 134
ideal motion problem, 129
image acquisition, 11
image center, 15
image coordinate system, 14
image gradient, 42
image intensity, 13, 24
image noise, 26
image plane, 16
image resolution, 13
image sampling, 13
intensity equalization, 57
interest image, 38, 44
intrinsic camera parameters, 18
invariant-based algorithm, 83
inverse compositional algorithm, 53
irradiance, 23

Jacobian matrix, 102

k-D tree algorithm, 136
Kanade-Lucas-Tomasi tracker, 40
key frame selection, 108

Lambertian surface, 23
least fractional squares, 142
least median of squares, 118, 139
least trimmed squares, 140
Levenberg-Marquardt method, 102
luminance, 24

M-estimator, 122
measurement matrix, 82
midpoint method, 86

nearest neighbor interpolation, 43
nearest neighbor search, 136
normal equations, 103
normalized correlation coefficient, 71
normalized cross correlation, 71

object tracking, 40
    data-driven, 218
    model-based, 219
optical axis, 16
optical center, 15
outlier, 117
outlier rejection, 47, 54, 64, 120

parameterization, 100
parameterized warp function, 48
paraperspective projection, 17
perspective projection, 16
pixel, 14
point set registration, 5, 129
POSIT algorithm, 91
primary aberrations, 20
principal point, 15

quantization, 13

radial distortion, 21
radiance, 23
radiometric image formation, 22
reference plane, 17
region tracking, 40
registration error, 135
relative camera pose estimation, 95
result propagation strategy, 63
RGB color space, 23
robustness, 37, 81, 132
Rodrigues' rotation formula, 101

sensor coordinate system, 14

seven-point algorithm, 84
shutter, 27
singular value decomposition, 91
Sobel operator, 42
sparse 3-D reconstruction, 3
stopping criterion, 43
structure and motion, 4, 78
surface reflectance model, 23

template matching, 40
thick lens model, 20
thin lens model, 20
three-point algorithm, 88
Tomasi-Kanade detector, 44
triangulation, 86

VAMPIRE
    augmented reality gear, 2, 69
    cognitive vision system, 1
    project, 1
virtual reference point, 93

weak-perspective projection, 17
world coordinate system, 14