

Contributions to Real-time Visual Tracking and Live Environment Modeling for Augmented Reality Applications

Design and Implementation of New Methods and
Evaluations on Real Image-based Ground Truth Data

Sebastian Lieberknecht

TECHNISCHE UNIVERSITÄT MÜNCHEN

Fakultät für Informatik

Computer Aided Medical Procedures & Augmented Reality / I16

Contributions to Real-time Visual Tracking
and Live Environment Modeling
for Augmented Reality Applications.

Design and Implementation of New Methods and
Evaluations on Real Image-based Ground Truth Data

Sebastian Thomas Lieberknecht

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität
München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Hon.-Prof. Dr. C. Steger
Prüfer der Dissertation: 1. Univ.-Prof. Dr. N. Navab
2. Prof. Dr. G. Reitmayr,
Technische Universität Graz / Österreich

Die Dissertation wurde am 09.11.2012 bei der Technischen Universität München eingereicht
und durch die Fakultät für Informatik am 27.12.2012 angenommen.

ABSTRACT

Real-time visual tracking and live environment modeling are key problems of Augmented Reality (AR) applications. A multitude of methods tackling these fundamental problems have been proposed and evaluated in the literature in the past few decades, while in general no common datasets were used. This thesis presents a novel methodology for benchmarking visual tracking algorithms. One of the first outcomes of the benchmarking approach was the creation of the first dataset specifically designed for planar template and/or feature-based visual tracking methods. It was created using a camera mounted on a highly accurate mechanical measurement arm.

All steps necessary to reproduce such a dataset are detailed including calibration, registration and synchronization. The publication of the dataset in 2009 was accompanied by the evaluation of four tracking methods. In the meantime the dataset was also used by several research institutes and universities and is an integral part of the Trakmark benchmark. Later, other outcomes followed such as the creation of datasets for evaluating 2.5 D and 3 D visual tracking methods and their usage for designing and evaluating new methods.

The thesis also presents two novel methods for tracking with a hand-held camera. Both approaches aim to reconstruct parts of the environment online so that augmentations can be better integrated into the environment by realistically modeling their occlusions.

The first approach, dubbed *dense deformable tracking*, is an extension of frame-to-frame template tracking to 2.5 D. A reference image of an object is given to the algorithm. It uses every pixel of the template to perform a free-form deformation of this reference image towards the true shape of the imaged object; simultaneously, the pose of the camera is estimated.

The advent of the Microsoft Kinect in Nov. 2010, and thus the general availability of a low-cost high resolution depth camera with registered optical image, reduced the complexity of the reconstruction problem. The thesis presents a live tracking and meshing system that can be used to track and reconstruct arbitrarily shaped objects for which the Kinect provides depth information. The system is initialized using only one frame. It instantly creates an environment mesh that allows to correctly handle occlusions.

The dense deformable tracking and the method based on the Kinect were evaluated on challenging objects using the methodology presented above to create ground truth data. Both methods were shown to be more accurate than related state-of-the-art methods. Several further improvements are outlined.

ZUSAMMENFASSUNG

Das Tracking der Kamerapose in Echtzeit sowie die gleichzeitige Rekonstruktion der durch die Kamera erfassten Umgebung sind Schlüsselemente hochqualitativer Augmented Reality-Anwendungen. Eine Fülle von Methoden wurde in diesen Gebieten veröffentlicht. Aufgrund unterschiedlicher Datenbasis bei den jeweiligen Evaluierungen sind diese Methoden untereinander jedoch zumeist nicht quantitativ vergleichbar.

Diese Arbeit stellt eine Methodik vor, gemäß derer optische Trackingverfahren objektiv und anhand realer Kamerabilder verglichen werden können. Ein erster Datensatz wurde entworfen speziell für die Evaluierung von planaren Trackingverfahren. Sämtliche hierzu nötigen Schritte werden detailliert beschrieben, beginnend bei der verwendeten Hardware, einem hochgenauen mechanischen Messarm mit daran montierter Kamera, bis hin zur gleichzeitigen Synchronisierung und Registrierung der Kamerabilder. Neben den vier bei der Veröffentlichung evaluierten Trackingverfahren wurde dieser Datensatz inzwischen von weiteren Universitäten und Forschungseinrichtungen angenommen und ist Kernbestandteil von Trakmark.

Diese Arbeit stellt außerdem zwei neuartige Trackingverfahren vor, welche auch Teile der Umgebung rekonstruieren für eine realistischere Darstellung der Augmentierungen einschließlich ihrer Verdeckungen.

Das erste Verfahren ist eine Erweiterung klassischer 2D Template-Tracking-Ansätze zu 2,5D. Jeder Pixel eines Referenzbilds wird dem aktuellen Kamerabild zugeordnet, indem innerhalb derselben Kostenfunktion sowohl Kamerapose als auch Deformation der Oberfläche des Templates geschätzt wird.

Das zweite Verfahren verwendet die im Laufe der Arbeit verfügbar gewordene RGB-D Kamera Kinect von Microsoft, um zur Laufzeit beliebig geformte Objekte zu rekonstruieren und tracken. Das System wird durch nur ein RGB-D Bild initialisiert, stellt dank des metrischen Tiefenbildes der Kinect Augmentierungen sofort im richtigen Maßstab und korrekt von der Umgebung verdeckt dar. Gegenüber dem ersten Verfahren hat sich durch den Einsatz von Punktmerkmalen zum Tracking die Robustheit deutlich gesteigert.

Beide Verfahren wurden anhand herausfordernder Objekte auch quantitativ evaluiert, wobei über die eingangs erwähnte Methodik eine hochgenaue Kamerapose zu jedem Bild bekannt war. Die entwickelten Verfahren sind genauer als verwandte aktuell veröffentlichte Methoden.

Abschließend wird Verbesserungspotential aufgezeigt für die Methodik zur Gewinnung von *ground truth* Daten und für beide Trackingverfahren.

ACKNOWLEDGEMENTS

This part of the journey is coming to an end. It started in August 2008, when I had just finished my studies. My wife Laura was expecting our first born child, Milosch. It was a time of major changes.

Fast forward 38 months: Selim, Daniel, Thomas and I are driving back home from Basel, Switzerland. I just presented our RGB-D tracking and meshing method at ISMAR and we won the tracking contest on a mobile phone. My son Max is three months old. This thesis consisted of 102 pages and many todos.

All this would not have been possible without the support from many people. First of all, I would like to thank Peter Meier and Thomas Alt for their trust in me, for creating the PhD position in their company and for continuously supporting me throughout my 8+ years at metaio. I would also like to thank Nassir Navab for accepting me as PhD student at his chair.

I am grateful to my supervisors Selim Benhimane and Slobodan Ilic. Selim's genetic curiosity, spirit and modesty are inspiring. He supported me on numerous occasions and provided precious feedback, also for the thesis. I am thankful to Slobodan for supporting me at CAMP. His views and competence were extremely helpful. I am greatly indebted to both of my supervisors. They taught me how to properly conduct and present academic work.

I would also like to thank the members of my committee, Prof. Dr. Carsten Steger and Prof. Dr. Gerhard Reitmayr, for taking the time to review the thesis and participating in my PhD defense.

Thanks goes to my colleagues at metaio for their friendship, support and countless funny moments – and incredible amounts of trash talk. Especially I would like to thank my long-term roommates Daniel Ulbricht, Quintus Stierstorfer, Georg Kuschik and Marion Langer; it was also a privilege to work with Daniel Kurz, Thomas Olszamowski, Andrea Huber and Marion Koelle.

Thanks also goes to my fellow PhD students at CAMP: Cédric Cagniard, Benoît Diotte, Bertram Drost, Stefan Hinterstoisser, Stefan Holzer and Christian Unger. It was a pleasure, you are great researchers.

I would like to thank my parents for their unconditional support. Thank you Laura, you are the sunshine of my life. Finally, to my sons Milosch and Max: Thank you for showing me day-to-day what really matters.

CONTENTS

1	Introduction	1
1.1	What is Augmented Reality?	1
1.2	Contributions of the thesis	3
1.3	Structure of the thesis	4
1.4	Publications related to the thesis	4
2	Ground Truth Generation for Evaluating AR Tracking Methods	7
2.1	Motivation	7
2.2	Related work	11
2.3	Hardware setup	14
2.4	Recording datasets for template-based methods	20
2.4.1	Determining ${}^{base}\mathbf{T}_{target}$	20
2.4.2	Capturing and synchronization	21
2.5	Recording datasets for SLAM-like methods	25
2.6	Designing tracking datasets	27
2.6.1	Planar targets	27
2.6.2	Non-planar convex targets	33
2.6.3	Non-planar targets of arbitrary shape	34
2.7	Evaluation methodologies	36
2.7.1	Evaluating template-based methods	36
2.7.2	Evaluating SLAM-like methods	39
2.8	Evaluation of four template-based methods	41
3	Dense Deformable Template Tracking	45
3.1	Motivation	45
3.2	Related work	47
3.3	Method	49
3.3.1	Parameterization of the mesh updates	51

CONTENTS

3.3.2	Minimizing the cost function	53
3.3.3	Regularization	54
3.4	Evaluation	58
3.4.1	Quantitative evaluation	58
3.4.2	Qualitative evaluation	62
4	RGB-D based Tracking and Meshing	65
4.1	Motivation	65
4.2	Related work	68
4.3	Real-time camera motion estimation	70
4.3.1	Inter-frame tracking	70
4.3.2	Sparse feature mapping	73
4.3.3	Relocalization	74
4.4	Online environment mesh creation	76
4.4.1	Creating local textured meshes	76
4.4.2	Integration of local meshes	77
4.5	Ground Truth-based evaluation	80
4.5.1	Quantitative evaluation of the tracking	80
4.5.2	Quantitative evaluation of the shape	81
4.6	Application on different AR scenarios	83
4.6.1	AR-based virtual furniture trial	84
4.6.2	Visual discrepancy check	85
4.6.3	Maintenance scenario	85
5	Future Work	87
6	Conclusion	91
	Appendices	97
A	Transformations and Lie algebra of $SE(3)$	97
B	Interpolation	101
C	Physical Imaging Process	103
D	Image Credits	107
E	Supplementary material	109
E.1	Ground Truth for AR tracking methods	109
E.2	Dense deformable template tracking	110
E.3	RGB-D based tracking and meshing	111

References

115

1.1 What is Augmented Reality?

Augmented Reality (AR) is the concept of superimposing virtual data onto the real world to enhance the user's perception of the world in order to assist a user in accomplishing a certain task. Unlike *Virtual Reality*, where the assistance is done in a purely virtual world, the goal of AR is to support the user by adding context-aware information that is registered to the *real* environment.

The range of applications for AR is immense; it includes navigation, training, industrial scenarios like maintenance and prototyping, educational scenarios like magic books, medical scenarios, advertisement and entertainment. In this thesis, we consider visual AR where 3D computer generated graphics are overlaid on real-time acquired camera images. Here, from a technical perspective, AR consists of continuously tracking the camera manipulated by a user with respect to an object of interest and presenting him information *e.g.* in visual form on a display. The thesis deals with methods for AR using a hand-held or head-mounted camera, a setting termed *visual inside-out tracking*; it is potentially the most employed setting today due to the prevalence of smartphones and tablets which are equipped with cameras and reasonably powerful processors. For a broader introduction to AR which sheds some light also on other tracking technologies, please refer to Azuma's Survey of AR [Azu97, ABB⁺01].

The realm of visual tracking can be characterized in a number of ways. One important characteristic is the degree of knowledge about a scene before the

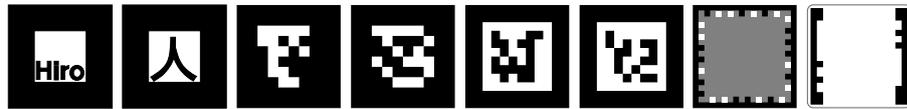


Figure 1.1: *From left:* Markers from ARToolkit [KB99], ARTag [Fia10], metaio [PMK06] and Studierstube [WRM⁺08] (two markers each).

tracking is started: Is it equipped with artificial landmarks, are parts of the “natural” environment known *e.g.* in form of (textured) models, or is there nothing known about the structure of the scene at all? In the last case, the tracking system has to localize the camera and create a map of the environment at the same time. This is known from robotics as simultaneous localization and mapping (SLAM). As SLAM systems usually cannot infer knowledge about a scene, they are more likely used for less context-aware applications such as games. Context can be gradually added by the user if necessary. For example, by specifying the size of a real-world object, augmentations can be shown in metric scale which is useful *e.g.* for virtual furniture trial.

When AR should be used in an environment which is known or which can be modified to some extent, the tracking problem is simplified a lot. One of the most popular early software development kits for AR has been the ARToolkit [KB99]. It relies on the user to specify *markers*, *i.e.* images with a square thick black frame and white margin (see figure 1.1). Later methods for detecting markers improved the robustness by employing codes within the frame [Fia10, PMK06] and also changed the appearance of the frame [WRM⁺08] to reduce the visual clutter for the user.

Another class of tracking methods has less strict constraints on the appearance of an object that should be tracked. This *marker-less* class typically employs sparse feature points, densely textured models or wire-frame models. Usually, there is only a requirement on the gradient of the texture, but there is no necessity for a specific shape being present for these methods to work opposed to the marker-based case. The natural appearance of the object suffices to provide enough information for performing the tracking task. A good introduction on marker-less tracking technologies for AR can be found in [Kle06].

Tracking methods may further be separated into those that do not require any knowledge about the previous pose of the camera (*detection* methods) and those that require an estimate of the pose in the current frame (*inter-frame* methods). Detection methods are typically either computationally expensive, as

e.g. SIFT [Low04], require relatively large amounts of memory per target for precomputed data, as *e.g.* Randomized Trees [LF06], or deliver a coarse relatively estimate of the camera pose. Inter-frame methods typically require less computational efforts and produce a fine estimate of the camera pose, *e.g.* the edge-based tracking system presented in [DC02] or the template-based tracking method ESM [BM07]. A full tracking system consists of methods from both classes, typically using an inter-frame method to refine the result of a detection method.

1.2 Contributions of the thesis

While big differences in the accuracy of tracking algorithms can be shown effortlessly using ad-hoc created video sequences or a live camera feed, these perceived differences are usually hard to quantify. From some moment on, every research field demands for common datasets to objectively compare different methods.

In this thesis, we present a novel methodology to create datasets with highly accurate ground truth for evaluating visual tracking methods based on *real* camera images. A first dataset was created for evaluating methods that track planar targets. This dataset has been used by several others. Furthermore, the dataset has been integrated into the development process of the commercial marker-less tracking systems of metaio GmbH¹. Over the course of the thesis, the nightly evaluations showed up several unintended changes very early. They were also valuable when components of the tracking systems were changed on purpose.

Furthermore, a method for simultaneously tracking and reconstruction a 2.5D template was developed. The method targets low-textured near-convex objects. Regular planar template tracking has been extended by allowing vertices of an overlaid mesh to move along their projection ray in the template image. The method was evaluated on several low textured objects and compared both to planar template tracking and a SLAM method, with known absolute ground truth based on the methodology mentioned in the last paragraph.

During the thesis, the Microsoft Kinect was launched, *i.e.* a low-cost camera delivering per pixel color and depth information (RGB-D). Thanks to its

¹<http://metaio.com>

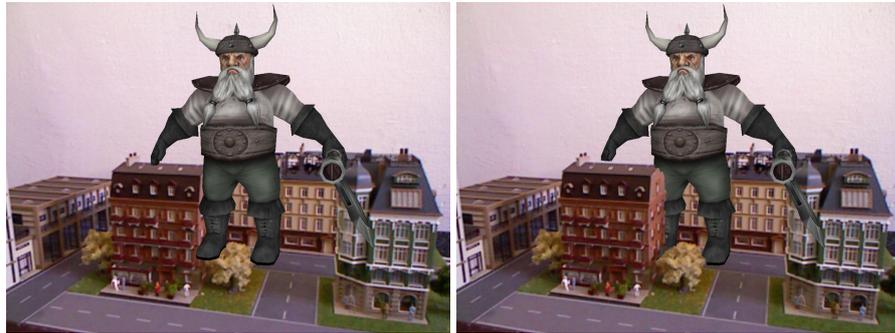


Figure 1.2: Effect of occlusion on human depth perception. A virtual dwarf is augmented in a miniature city. *Left*: Standard augmentation. *Right*: Augmentation with occlusion geometry.

relatively high resolution and update rate, the complexity of the dense reconstruction problem became manageable for many scenarios. This thesis presents a SLAM system that uses the Kinect. The system tracks the pose of the camera while a dense mesh of the environment is created. The mesh is used to occlude augmentations by real objects which greatly improves the integration of the augmentation as shown in figure 1.2.

1.3 Structure of the thesis

The thesis is structured as follows: After the introduction, the three main contributions of the thesis are presented, namely a methodology for ground truth generation for evaluating AR tracking methods (chapter 2), a method for dense deformable template tracking (chapter 3) and RGB-D based tracking and meshing (chapter 4). Each of these chapters starts by explaining the motivation of the specific contribution, followed by representative related work. Then, the method is presented and evaluated in details. After sketching a few areas of future work, the thesis concludes with a synthesis of the different contributions and their impact on state-of-the-art methods.

1.4 Publications related to the thesis

Most of the work presented in this thesis has been peer-reviewed and presented at conferences, published in conference proceedings, or published in journal proceedings. A book chapter has been published about a point-based 3D model-

based tracking system which formed the basis of the work on RGB-D tracking system. Further three related patent applications were filed. A list of the publications follows.

1. S. LIEBERKNECHT AND P. MEIER, *Methods and systems for analysing an image generated by at least one camera*, patent application filed 2008-11-10, WO 2010-051825 A1 (2010-05-15), EP 2356583 A1 (2011-08-17), US 2011-0280445 (2011-11-17).
2. S. LIEBERKNECHT, S. BENHIMANE, P. MEIER, AND N. NAVAB, *A dataset and evaluation methodology for template-based tracking algorithms*, in ISMAR, 2009.
3. S. LIEBERKNECHT, S. BENHIMANE, P. MEIER, AND N. NAVAB, *Benchmarking template-based tracking algorithms*, International Journal of Virtual Reality, Special Issue on Augmented Reality, 15 (2011), pp. 99–108.
4. S. LIEBERKNECHT, Q. STIERSTORFER, G. KUSCHK, D. ULBRICHT, M. LANGER, AND S. BENHIMANE, *Handbook of Augmented Reality*, Springer, 2011, ch. Evolution of a Tracking System, pp. 355–378.
5. B. BECKER, S. CUI, M. HUBER, P. KEITLER, G. KLINKER, S. LIEBERKNECHT, P. LÜCKEWILLE, D. PUSTKA, R. SCHEIBE, L.A. SCHWARZ, B. SCHWERDTFEGER, C. WÄCHTER, A. WEISS, AND K. ZÜRL, *Virtuelle Techniken im industriellen Umfeld: Das AVILUS-Projekt – Technologien und Anwendungen*, Springer, 2011, ch. Lokalisation und Tracking, pp. 83–109
6. S. LIEBERKNECHT AND S. BENHIMANE, *Method for determining a parameter set designed for determining the pose of a camera and/or for determining a three-dimensional structure of the at least one real object*, patent application filed 2010-12-21, PCT/EP2010/007831 (2012-06-28)
7. S. LIEBERKNECHT, S. BENHIMANE, AND S. ILIC, *Simultaneous reconstruction and tracking of non-planar templates*, in DAGM, 2011.
8. S. LIEBERKNECHT, S. BENHIMANE, AND A. HUBER, *Method for estimating a camera motion and for determining a three-dimensional model of a real environment*, patent application filed 2011-08-31, PCT/EP2011/065007 (to appear 2013-02)
9. S. LIEBERKNECHT, A. HUBER, S. ILIC, AND S. BENHIMANE, *RGB-D camera-based parallel tracking and mapping*, in ISMAR, 2011.

10. D. KURZ, S. LIEBERKNECHT, AND S. BENHIMANE, *Benchmarking inertial sensor-aided localization and tracking methods*, in Trakmark Workshop at ISMAR, 2011.

GROUND TRUTH GENERATION FOR EVALUATING AR TRACKING METHODS

This chapter describes a method to create ground truth data for the quantitative evaluation of tracking algorithms suited to AR applications, *i.e.* image sequences captured from a hand-held camera with a precisely known camera pose for every image of the sequences.

2.1 Motivation

In the last few years, marker-less visual tracking reached the level where a large variety of algorithms could be successfully used in a wide range of AR applications [KM07, WRM⁺08, CM11]. Only accurate, robust and stable tracking can empower a natural interaction with AR. Until recently, the performance of state-of-the-art algorithms was either evaluated quantitatively using synthetically rendered images or evaluated qualitatively using *ad hoc* recorded videos demoing a new method.

Using a synthetic approach has the advantage that the pose of the virtual camera can be set perfectly with respect to the synthetic scene, an example is shown in figure 2.1. A pose estimate obtained from a given tracking algorithm can be compared to this ground truth pose in order to compute an estimation error. However, the author of the synthetic scene has to carefully model the motion and the imaging properties of the camera.

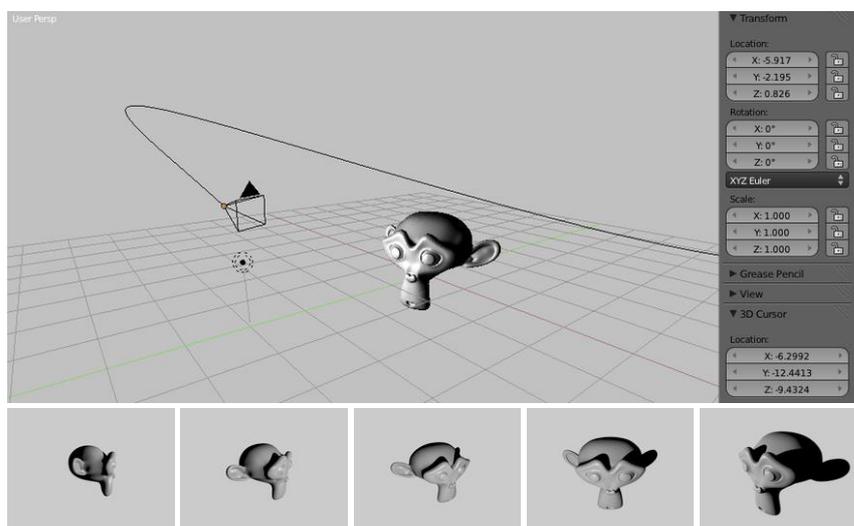


Figure 2.1: Synthetic scene with camera moving on a predefined path. *Top*: Scene is shown in 3D modelling software Blender [Ble], camera highlighted in orange. *Bottom*: Excerpt of the synthetic sequence.

The creation of realistic camera motion can be simplified by using motion capture hardware. However, the creation of realistic images remains a challenging task as there are numerous effects involved into the physical imaging process as described in appendix C. Figure 2.3 illustrates some of the major influence factors of virtual and real imaging process and camera motion. Some of these effects are observable in figure 2.2. In general, the imaging effects of a real camera can only be approximated which makes the evaluations intrinsically biased towards these approximations.

Synthetic sequences can be used during the design of new algorithms. They are suitable for early stages of algorithmic design to get a first impression of the behavior of the tracking. For example, the author of a method can first work on perfect data to simplify the development, then add more and more imperfections as the method matures. But synthetic data cannot guarantee the performance of a method when it is used with the real world data it was ultimately designed for.

The performance of a given algorithm can be easily shown on recorded or live videos of real scenes – but usually, the evaluation of the camera pose in this case remains qualitative. For example, it is possible to visually observe whether the camera pose was correctly estimated from the pose of the virtual augmentations or from the movement of the camera given an additional birds-eye view of the scene.



Figure 2.2: Digital image with various artifacts (*contrast and brightness adjusted for clarity*). High amounts of *sensor noise* are shown due to high signal gain in (originally) low-light condition (high ISO setting); *motion blur* due to hand-held camera; *blooming*, *i.e.* charge overflow from the “bright” cells to connected sensor cells (in lower right corner).

While this is also appropriate for early algorithmic development, the major drawback of such an evaluation is that it is subjective and that it needs a human observer. This can be a limiting factor both for small differences in performance and for automating the evaluation in general. Automated evaluation can be employed for *e.g.* assessing the precision and accuracy after changes to tracking algorithms on a daily basis in order to not accidentally degrade the quality of a method while refactoring its source code.

Since the research community is working on marker-less visual tracking very actively, the need of common objective datasets with ground truth is growing. We believe that image sequences acquired with a real camera where the pose of the camera for every frame of the sequences is accurately estimated give very strong benefits to the community. Such datasets allow a fast performance estimation in terms of speed and accuracy of a newly designed algorithm and its fair comparison with the state-of-the-art.

In the remainder of this chapter, we first present the related work. Next, we show the methodology used to record image sequences and generate ground truth data of the position and orientation of the camera in every frame. For

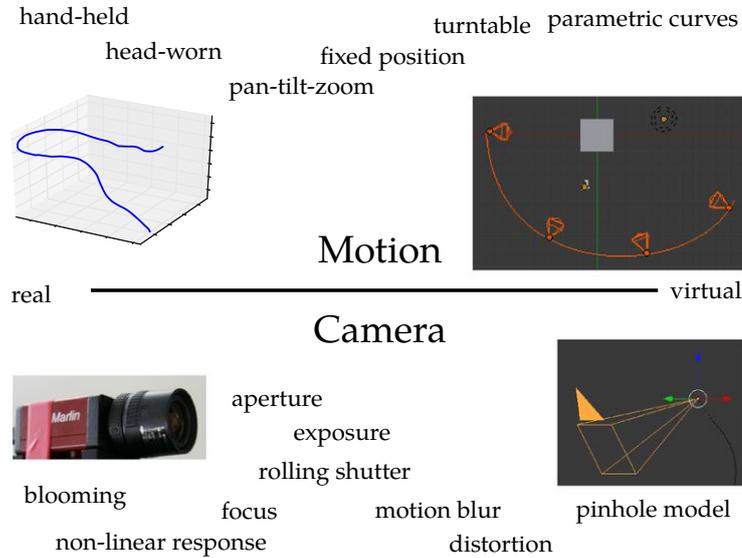


Figure 2.3: Overview of major influences on real imaging process and camera motion. The major steps of the imaging process are described in appendix C.

this, first the used hardware will be presented, then the calibration, acquisition and synchronization steps will be detailed for methods with prior knowledge about the scene (such as template tracking algorithms) as well as for SLAM-like methods.

Then, we present the datasets we created, starting with the template-tracking dataset that is now used by researchers of several universities and research institutes. Next, datasets featuring non-planar targets are shown and evaluation methodologies for planar and non-planar methods are presented. The datasets are used to evaluate four widely used template-based methods. The methods for non-planar and RGB-D tracking developed as part of this are evaluated in sections 3.4 and 4.5, respectively.

2.2 Related work

For a long time, Quam’s Yosemite sequence [Hee87] used to be the reference for evaluating optical flow algorithms. Quam created a model by mapping aerial photos onto a depth map of the Yosemite valley and generated a sequence by simulating a flight through the valley, shown in figure 2.4.



Figure 2.4: *From left:* Part of the “Yosemite” and “Urban” synthetic optical flow sequences; stereo dataset “Teddy” (all from the Middlebury datasets [BSL⁺07]).

Today, the Middlebury datasets [BSL⁺07] are the reference for optical flow. Besides having generated synthetic images, dense ground truth from real images was created by using hidden fluorescent texture. The same group additionally made ground truth datasets for dense stereo matching using structured light and datasets for multi-view stereo using a laser scanner [SCD⁺06, SZS⁺08].

In marker-less visual tracking, Baker and Matthews [BM04] used synthetic image warping to compare four different template-tracking algorithms [LK81, SS00, BM01, BM04], they varied the warping amplitude and the noise level of the image. Mikolajczyk and Schmid [MS04] used still images for comparing affine region detectors, their dataset was also used by many others (*e.g.* [DT05, BTG06, NS06]) to show the performance of several feature detection, description and matching schemes. Moreels and Perona [MP07] used a turn table together with a static stereo camera setup to evaluate the performance of feature detectors and descriptors on 3 D objects. They generated a database consisting of 100 objects with calibrated views, one image pair for each 5° rotation of the turntable.

None of the aforementioned datasets had been created specifically for evaluating tracking methods as used in AR applications. Arguably, they could also be used for this purpose, but the results of such an evaluation would not reflect a method’s performance in an inside-out AR scenario. This is because several typical but non-negligible effects are missing in the previous datasets; in particular the effects of live image acquisition, *i.e.* many images with typically very small inter-frame distance, the effects of a hand-held device motion, *i.e.* full



Figure 2.5: Targets used by Zimmermann, Matas and Svoboda [ZMS09] and one frame from a sequence. Ground truth was obtained by manually clicking on the corners.

6 DOF unconstrained motion instead of *e.g.* 1 DOF constrained motion for turntable sequences, and the effects of the physical imaging process of a real camera, *e.g.* motion blur.

Recently, Zimmermann, Matas and Svoboda [ZMS09] published a dataset consisting of gray-scale image sequences with transformations of the tracking targets that cover all six degrees of freedom and that thus could be used to evaluate frame-to-frame tracking algorithms. The ground truth location of the targets in 2D was manually obtained by clicking on either crosses that were attached to objects or by clicking directly on the corners of an object. The three image sequences consist of approximately 12,000 images in total and feature three different targets, namely a mouse pad, a towel and a desktop phone as shown in figure 2.5.

To our knowledge, at the moment of its publication, it was the first dataset that can be used for evaluating planar tracking methods in 2D quantitatively. The intrinsic calibration of the camera and the real-world dimension of each target are not given, such that an evaluation of a metric 6 DOF Euclidean camera pose is hard to obtain. The dataset considers a very limited number of objects and major factors that influence the tracking were fixed, *e.g.* the lighting conditions. Moreover, the ground truth data was based on information exclusively coming from the images. Following this approach, it is not possible to have reliable ground truth in the case of blurry or noisy images. It is also not possible to recover the camera position and orientation when the points used to determine the pose are not in the field of view of the camera. Consequently, the performance of the tested algorithms could not be evaluated in the presence of noise, motion blur or for some relative position between the camera and the tracked objects.

After the publication of the method and dataset described in this chapter, Gruber *et al.* [GGV⁺10] presented a set of 3D paper models that can be built from printouts. The authors included different image sequences with estimated

camera motion. For creating these sequences, cameras were attached to a mechanical robot arm, a mechanical measurement arm and an optical infrared tracking systems. The paper models and image sequences together with the calibration data are published online¹.

Gauglitz *et al.* [GHT11] recently published a dataset designed for planar targets that used six different patterns and focused on 16 different camera paths, including pure rotation, panning and several levels of motion blur. Four distinctive red spheres were put next to the patterns. They were used to compute a homography for each video frame. These homographies, the original image sequences as well as the intrinsic camera calibration and distortion coefficients are available for download².

¹http://studierstube.icg.tugraz.at/handheld_ar/cityofsights.php

²http://ilab.cs.ucsb.edu/tracking_dataset_ijcv/

2.3 Hardware setup

For the generation of ground truth data, we followed the principle of using proven hardware components that are typically used for measuring tasks and industrial quality assurance. This led us to choose a mechanical measurement arm as key component in conjunction with an industrial global shutter camera. The camera was mounted on the end effector of the measurement arm. Photogrammetric hardware was used for calibrating the system. An image of the setup is shown in figure 2.6. The measurement arm is mounted on a tripod which is clamped to a table. On the table, the target is fixed with adhesive tape.

The goal of the setup is to provide an alternative way for the camera pose ${}^{camera}\mathbf{T}_{target}$ provided by tracking algorithms. A pose is represented as a (4×4) matrix and can be decomposed in a (3×3) rotation \mathbf{R} and a (3×1) translation \mathbf{t} (c.f. appendix A). To evaluate the accuracy of the pose ${}^{camera}\mathbf{T}_{target}$ provided by a tracking algorithm, we indirectly compute the reference pose as

$${}^{camera}\mathbf{T}_{target} = {}^{camera}\mathbf{T}_{tip} \cdot {}^{tip}\mathbf{T}_{effector} \cdot {}^{effector}\mathbf{T}_{base} \cdot {}^{base}\mathbf{T}_{target} \quad (2.1)$$

where *base* refers to the base of the measurement arm located on the tripod, (end) *effector* refers to the last link of the arm moved by the user and *tip* refers to the exchangeable tip attached to the end effector which is used for measuring coordinates. As measurement arm we chose the FaroArm Platinum [FARa]. It has seven axes and an accuracy better than 0.013 mm within its reach of around 1.2 m from its base. The arm uses internal temperature sensors to compensate the effect of thermal variations. We assume that for our purpose the residual noise

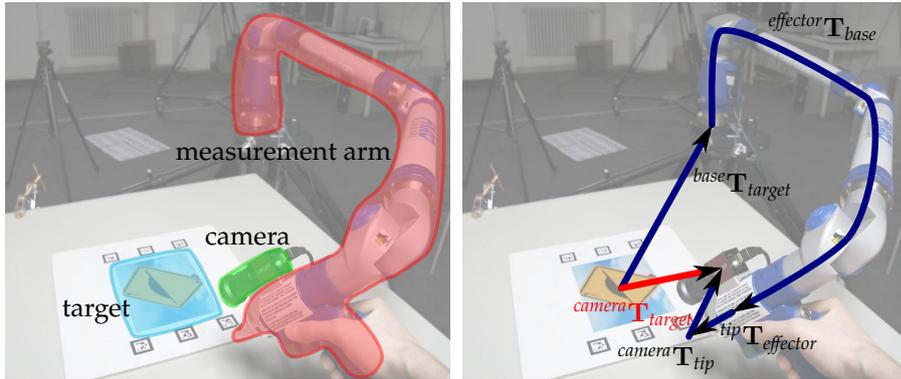


Figure 2.6: *Left*: The measurement arm, the camera and one of the targets. *Right*: The transformations needed to compute ${}^{camera}\mathbf{T}_{target}$ which is displayed in red.



Figure 2.7: The daylight panel mounted above the target. The panel was adjusted by switching on/off five independent fluorescent tubes.

of the measurement arm is neglectably small. An industrial AVT Marlin F080C camera [All] was rigidly mounted on the end effector using an aluminum adapter screwed between tip and end effector. The camera is able to progressively capture VGA frames with 40 Hz while the measurement arm provides absolute pose data with 250 Hz. As we planned to vary the lighting in case of the template-based tracking class, we used an adjustable panel light providing daylight-balanced 5600 Kelvin light (see figure 2.7).

To make the system operational, first the camera intrinsics and undistortion coefficients were estimated. The camera intrinsics consist of the focal lengths f_u and f_v , the principal point $\mathbf{u}_0 = [u_0 \ v_0 \ 1]^\top$ and the shear s . The intrinsic parameters form the matrix \mathbf{K}

$$\mathbf{K} = \begin{bmatrix} f_u & s & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.2)$$

which includes the distortion caused by non-rectangular pixels via s and non-square pixels via the ratio f_v and f_u . A 3D point $\mathbf{x} \in \mathbb{P}^3$ can be projected onto the image plane by first transforming them from the target to the camera coordinate frame via ${}^{camera}\mathbf{T}_{target}$. Next, the transformed point is multiplied by the intrinsics \mathbf{K} to take account of the specific lens and image sensor. Finally, we normalize the homogeneous coordinates by applying

$$\mathbf{d} \left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} \right) = \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix} \text{ for } z \neq 0. \quad (2.3)$$

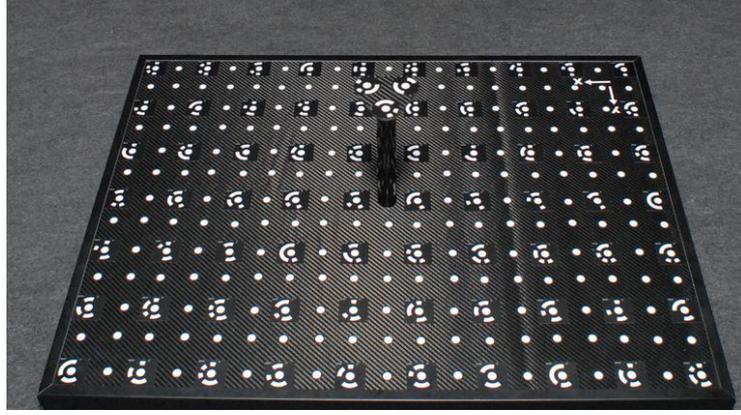


Figure 2.8: Calibration target used for camera intrinsics (\mathbf{K} and distortion coefficients) and hand-eye calibration ${}^{camera}\mathbf{T}_{tip}$.

The normalization can be done either before or after the multiplication with \mathbf{K} . The linear projection thus can be written as

$$\mathbf{u} = \mathbf{d} \left(\begin{bmatrix} \mathbf{K} & \mathbf{0} \end{bmatrix} {}^{camera}\mathbf{T}_{target} \mathbf{x} \right) \quad (2.4)$$

where $\mathbf{0}$ is a (3×1) null vector and $\mathbf{u} = [u \ v \ 1]^T \in \mathbb{P}^2$. To estimate \mathbf{K} and further non-linear radial and tangential image distortions, we used a professional photogrammetric calibration target shown in figure 2.8 and the accompanying software from AICON [AIC]. Next, the underlying model for image distortion [Luh03, p120] used by AICON is described.

A 2D point \mathbf{u} is distorted to the 2D point \mathbf{u}' as

$$\mathbf{u}' = \mathbf{distort}(\mathbf{u}) \quad (2.5)$$

$$= \mathbf{u} + \mathbf{d}_{radial}(\mathbf{u}) + \mathbf{d}_{tangential}(\mathbf{u}) + \mathbf{d}_{affine}(\mathbf{u}) \quad (2.6)$$

with each distortion being computed as

$$\mathbf{d}_{radial}(\mathbf{u}) = \sum_{i=1}^3 a_i \left(r(\mathbf{u})^{2i} - r_0^{2i} \right) [\mathbf{u} - \mathbf{u}_0] \quad (2.7)$$

$$\mathbf{d}_{tangential}(\mathbf{u}) = 2 \begin{bmatrix} b_1 (u - u_0)^2 \\ b_2 (v - v_0)^2 \\ 0 \end{bmatrix} + \begin{bmatrix} b_1 & b_2 \\ b_2 & b_1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} r(\mathbf{u})^2 \\ 2(u - u_0)(v - v_0) \end{bmatrix} \quad (2.8)$$

$$\mathbf{d}_{affine}(\mathbf{u}) = [c_1 (u - u_0) + c_2 (v - v_0) \quad 0 \quad 0]^T \quad (2.9)$$

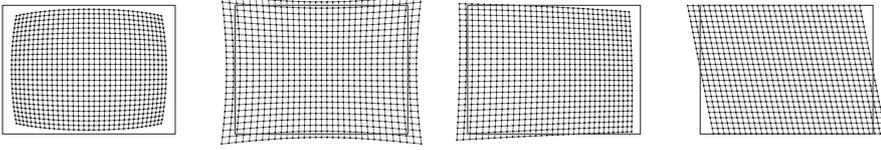


Figure 2.9: Sample distortions of a (640×480) image. The image border is indicated by the rectangular frame. *From left:* Radial distortions ($a_1 = -10^{-6}$ and $a_1 = 10^{-6}$), tangential ($b_1 = -10^{-4}$, $b_2 = 3 \times 10^{-5}$) and affine distortions ($c_2 = 2 \times 10^{-1}$). A centered principal point and $r_0 = 10^{-3}$ were used to create the examples.

where $r(\cdot)$ is the Euclidean distance to the principal point

$$r(\mathbf{u}) = \|\mathbf{u} - \mathbf{u}_0\|_2 \quad (2.10)$$

and (r_0, a_1, a_2, a_3) are coefficients for radial, (b_1, b_2) for tangential and (c_1, c_2) for affine distortion. Figure 2.9 shows exemplary distortion grids. Note that this distortion model also includes the linear distortions previously defined in \mathbf{K} . This can be seen from equations (2.9) and (2.2) by substituting $s = c_2 f_v$ and $f_u = c_1 f_v$.

For the presented methodology, we thus used only three degrees of freedom for \mathbf{K} , namely the principal point \mathbf{u}_0 and a common focal length $f(= f_u = f_v)$. The shear was ignored ($s = 0$). The result of the calibration then comprises three intrinsic coefficients (f, u_0, v_0) for \mathbf{K} and eight distortion coefficients. An undistorted $(U \times V)$ image \mathcal{I} is created from a distorted image $\tilde{\mathcal{I}}$ as

$$\mathcal{I}(\mathbf{u}) = \tilde{\mathcal{I}}(\text{distort}(\mathbf{u})) \quad \forall \mathbf{u} \in \mathbf{U} \quad (2.11)$$

where \mathbf{U} is the set of all integer pixel coordinates of the images

$$\mathbf{U} = \left\{ \begin{bmatrix} u & v & 1 \end{bmatrix}^\top \mid u \in \{1, \dots, U\}, v \in \{1, \dots, V\} \right\}. \quad (2.12)$$

We use bilinear interpolation for non-integer pixel coordinates (*c.f.* appendix B). The undistortion can be accelerated by caching the mapping between undistorted and distorted pixels $\mathbf{u} \leftrightarrow \mathbf{u}'$.

The calibration target used to estimate the intrinsic camera parameters and the distortion coefficients consists of a carbon fiber body onto which 73 distinctive and 215 non-distinctive circular markers were applied. The 3D position of the

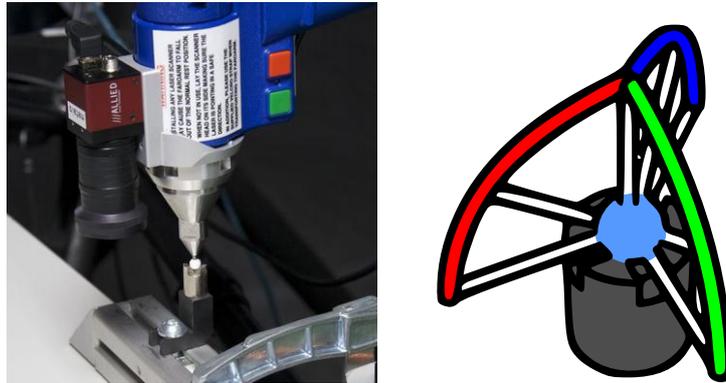


Figure 2.10: Calibration of the tip of the measurement arm (${}^{tip}\mathbf{T}_{effector}$). The tip should be located in the calibration target while the arm is moved on predefined paths [FARb].

center of each of these markers had been calibrated by AICON using a coordinate measurement machine. The camera calibration was conducted following the guidelines of the provider. Pictures were taken from recommended angles and distances such that the full sensor was covered by the images of the circular markers. The final reprojection error of the marker centroids was less than 0.065 pixels. This means that the calibration has provided accurate intrinsic parameter estimations.

Next, the intrinsics of the robot were calibrated, namely the seven joints and its tip. The calibration of the joints simply consisted of moving each joint within at least 90 % of its physically possible range. As can be seen from figure 2.6, this calibration allows us to get ${}^{effector}\mathbf{T}_{base}$ at 250 Hz. Obtaining the next transformation along the chain, *i.e.* the offset of the tip ${}^{tip}\mathbf{T}_{effector}$, is a slightly more tedious task. This transformation may vary because the tip can be exchanged, several tips of different diameter are available. A calibration cone as shown in figure 2.10 is mounted on a stable surface and the tip of the measurement arm is placed into the cone. Next, the end effector is moved while the center of the tip should not move; several hundreds measurements of such constrained motions are then used for computing the tip offset.

We observed that the accompanying software of the measurement arm gives a reliable calibration only when the cone is attached to a stable structure, *e.g.* a window board. While the arm itself was mounted on a tripod weighing around 20 kg, it was beneficial for the calibration to also rigidly connect the tripod to the same structure.

After that, the hand-eye-calibration was conducted for computing ${}^{camera}\mathbf{T}_{tip}$. For this, we again used the AICON calibration target and software, but this time mounted the end effector with already attached camera onto a second tripod in order to capture corresponding image-pose pairs. The calibration software was used this time only to obtain the highly accurate pose of the camera from the images. By moving the tripod, ten pose pairs were obtained and a hand-eye-calibration was computed with a method similar to the one proposed by Tsai and Lenz [TL89].

The next section describes how ground truth camera poses for known planar targets can be obtained. Besides computing the transformation ${}^{base}\mathbf{T}_{target}$ this also includes acquisition and synchronization of the images and poses of the measurement arm.

2.4 Recording datasets for template-based methods

Most of the hardware is now calibrated and ready to use. For evaluating methods tracking known targets, only the transformation ${}^{base}\mathbf{T}_{target}$ has still to be calibrated (per target). A sample target is shown in figure 2.11. We designed it to contain six fiducial markers next to the template image. While this section presents the steps needed to record ground truth sequences with any target and a wide range of motions, section 2.6 presents the specific targets and motions chosen for the proposed dataset.

2.4.1 Determining ${}^{base}\mathbf{T}_{target}$

The offset ${}^{base}\mathbf{T}_{target}$ is estimated via 3 D–3 D correspondences between the measured and designed positions of the outer fiducial corners. The correspondences for marker M_m are denoted as $\mathbf{x}_{m,i}$ and $\bar{\mathbf{x}}_{m,i}$, respectively. The plane π of the target is determined in a first step to enforce co-planarity among the measurements $\mathbf{x}_{m,i}$. The plane was measured by placing the tip of the arm on the target at several positions $\mathbf{p}_i \in \mathbb{P}^3$. A plane $\pi' = [\mathbf{n} \ d']^\top$ with normal $\mathbf{n} = [n_x \ n_y \ n_z]^\top$ was fit to these measurements by solving the system

$$\pi'^\top [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n] = \mathbf{0}. \quad (2.13)$$

The measurements \mathbf{p}_i specify the position of the *center* of the calibrated tip. The tip we used was a sphere with a radius $r = 3$ mm, thus the measurements \mathbf{p}_i used for determining the plane π' contain an translational offset of size r . The sign of the offset is determined by measuring an additional point \mathbf{p}_{off_plane} which is off the plane π' in the direction of the surface normal of the target. Then, the radius r of the measurement tip can be compensated by computing d as

$$d = d' + \text{sign}(\pi'^\top \mathbf{p}_{off_plane}) r \quad (2.14)$$

with

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases} \quad (2.15)$$

The derived plane $\pi = [\mathbf{n} \ d]^\top$ is on the surface of the target. Next, the tip is positioned on the four outer corners of each of the six fiducials M_m (marked with red crosses in figure 2.11). Each of these measurements $\mathbf{x}'_{m,i}$ is projected onto π as

$$\mathbf{x}_{m,i} = \mathbf{x}'_{m,i} - \pi^\top \mathbf{x}'_{m,i} \begin{bmatrix} \mathbf{n} \\ 0 \end{bmatrix}. \quad (2.16)$$

The 3D–3D correspondences $\mathbf{x}_{m,i} \leftrightarrow \bar{\mathbf{x}}_{m,i}$ are then used by the accompanying software of the measurement arm to estimate the transformation ${}^{base}\mathbf{T}_{target}$.

2.4.2 Capturing and synchronization

Next, we captured the image and pose streams. Our goal was to capture a live camera stream that can be played back for evaluations without loss of any detail. To accomplish this goal, the streams were recorded directly to pre-allocated main memory. Every sequence consists of $N = 1,200$ RGB images \mathcal{I}_i with a resolution of (640×480) pixels at 24 bit per pixel. The images were acquired from the camera at 40 Hz. We obtained absolute poses ${}^{effector}\mathbf{T}_{base,k}$ from the arm at only 75 Hz for the dataset due to a flaw of the threading of the FaroArm capture code at that time – for the evaluation of the RGB-D SLAM of chapter 4 this issue was fixed and we recorded at native 250 Hz.

As soon as the final image \mathcal{I}_N of a sequence was captured, we stopped capturing poses from the arm. Timestamps $t_{cam,i}$ and $t_{arm,k}$ were attached to the N images and $K \approx \frac{75}{40}N$ poses as soon as the capturing program had access to



Figure 2.11: A target used for evaluation with six fiducial markers $M_1 \dots M_6$ around it. The outer corners $\bar{\mathbf{x}}_{m,i}$ of the markers M_m , highlighted with red crosses, are used for the refinement of ${}^{base}\mathbf{T}_{target}$ and for synchronization.

them. The timestamps were used to synchronize the data, *i.e.* to associate a pose to each image \mathcal{I}_i . Using the same sequence for synchronization should provide a better accuracy for fast and sudden motions within this sequence compared to an a-priori synchronization on another sequence.

The arm supplies the poses ${}^{effector}\mathbf{T}_{base,k}$ as pairs of translations $\mathbf{t}_k \in \mathbb{R}^3$ and Euler angles for the rotation. As the synchronization requires a smooth interpolation of the poses, we converted the Euler angles to axis-times-angle $\mathbf{r}_k \in \mathfrak{so}(3)$. The captured data of about 1.1 GiB for 30 seconds were recorded directly into the main memory of the attached computer. This was done to minimize the influence of slowing down the recording at arbitrary moments because of hard disk access. After each sequence, the images, poses and their respective timestamps were written to disk as batch job.

Both the synchronization and the computation of the residual error of a sequence are based on the reprojection error of the marker corners. The images were undistorted with bilinear interpolation before the marker detection (*c.f.* sections 2.3 and appendix B). We assume that there is a constant offset t_{offset} between the timestamp of an image \mathcal{I}_i and its corresponding (interpolated) pose ${}^{camera}\mathbf{T}_{target}(t_{cam,i} + t_{offset})$ which minimizes the residual. The offset represents the time needed to capture the image, to transfer the image data via the various busses to the main memory and finally to invoke all hard- and software interrupts until the image is available to our software.

A pose ${}^{camera}\mathbf{T}_{target}(t)$ is computed as

$${}^{camera}\mathbf{T}_{target}(t) = {}^{camera}\mathbf{T}_{tip} {}^{tip}\mathbf{T}_{effector} {}^{effector}\mathbf{T}_{base}(t) {}^{base}\mathbf{T}_{target} \quad (2.17)$$

where the measurement arm pose ${}^{effector}\mathbf{T}_{base}(\cdot)$ is defined as

$${}^{effector}\mathbf{T}_{base}(t) = \begin{bmatrix} \mathbf{R}(\mathbf{r}(t)) & \mathbf{t}(t) \\ \mathbf{0} & 1 \end{bmatrix} \quad (2.18)$$

with $\mathbf{R}(\cdot)$ from equation (A.10). We interpolated $\mathbf{r}(\cdot)$ and $\mathbf{t}(\cdot)$ from the measurements as

$$\mathbf{r}(t) = \begin{bmatrix} \mathbf{r}_{k^*(t)} \\ \mathbf{r}_{k^*(t)+1} \end{bmatrix}^\top \begin{bmatrix} t - t_{arm,k^*(t)} \\ t_{arm,k^*(t)+1} - t \end{bmatrix} \quad (2.19)$$

$$\mathbf{t}(t) = \begin{bmatrix} \mathbf{t}_{k^*(t)} \\ \mathbf{t}_{k^*(t)+1} \end{bmatrix}^\top \begin{bmatrix} t - t_{arm,k^*(t)} \\ t_{arm,k^*(t)+1} - t \end{bmatrix} \quad (2.20)$$

where $k^*(t)$ returns the index of the measured timestamp closest to t as

$$k^*(t) = \underset{k \in \{1, \dots, K\}}{\operatorname{argmin}} (t - t_{arm,k} \mid t_{arm,k} \leq t). \quad (2.21)$$

The residual ϕ_i is computed for each image \mathcal{I}_i as

$$\phi_i(t_{offset}) = \sum_{M_m \in D_i} \sum_{c=1}^4 \|\mathbf{u}_{m,c} - \mathbf{u}(t_{cam,i} + t_{offset}, \bar{\mathbf{x}}_{m,c})\|_2 \quad (2.22)$$

with

$$\mathbf{u}(t, \bar{\mathbf{x}}) = \mathbf{d} \left(\begin{bmatrix} \mathbf{K} & \mathbf{0} \end{bmatrix}^{camera} \mathbf{T}_{target}(t) \bar{\mathbf{x}} \right) \quad (2.23)$$

and where $\mathbf{0}$ is a (3×1) null vector, D_i is the set of detected markers in image \mathcal{I}_i and $\mathbf{u}_{m,c} \in \mathbb{P}^2$ and $\bar{\mathbf{x}}_{m,c} \in \mathbb{P}^3$ are corresponding normalized 2D and 3D positions of corners of marker M_m .

The residual of a sequence was computed as the root of the mean of the squares (RMS) of the 90% smallest residuals $\phi_i(t_{offset})$ of all images with detected markers, *i.e.* where $D_i \neq \{\}$. We did not use all residuals in order to be robust to a small amount of outliers, specifically misdetections due to motion blur. The global residual was minimized with Nelder-Mead simplex [PCB02] to obtain the optimal offset t_{offset}^* as

$$t_{offset}^* = \underset{t_{offset}}{\operatorname{argmin}} \sqrt{\frac{1}{|K_{90\%}|} \sum_{k \in K_{90\%}} \phi_k(t_{offset})^2}. \quad (2.24)$$

Surprisingly, the initial residual was above one pixel, which made the dataset so far inadequate for ground truth evaluations. We analyzed each influence on the residual, starting from the fiducial detection. Although this component does not have a formally certified accuracy, it is highly accurate [PMK06] and thus appropriate for establishing a residual. We next rechecked the assumptions

about the single transformations of equation (2.1). The internal pose ${}^{effector}\mathbf{T}_{base}$ of the arm is known to be very precise, as it is a certified measurement system according to the ASME's B89.4.22 - 2004³. The guaranteed accuracy of better than 0.013 mm also includes the calibration of the tip (${}^{tip}\mathbf{T}_{effector}$). The 3D coordinates of the AICON calibration target were also obtained from a certified mechanical measurement device. The low residual error of below 0.065 pixel of the intrinsic calibration suggests that this part of the calibration (\mathbf{K} and distortion coefficients) is also accurate. For the estimation of the hand-eye calibration ${}^{camera}\mathbf{T}_{tip}$ only highly accurate poses were used, thus we assumed that it also is of high accuracy.

Finally, we decided to also include a refinement of ${}^{base}\mathbf{T}_{target}$ into the synchronization. We initially assumed that the 3D reference coordinates provided to the 3D-3D registration should represent the center of the tip. Instead, the provided 3D coordinates turned out to should have been the projections on the target plane. This means that we had an offset of about 3 mm per 3D-3D correspondences $\mathbf{x}_{m,i} \leftrightarrow \bar{\mathbf{x}}_{m,i}$. The refinement of ${}^{base}\mathbf{T}_{target}$ thus lead to a substantial improvement of the residual. For the proposed dataset, it is now below one pixel on average.

³see <http://www.asme.org/products/codes---standards/methods-for-performance-evaluation-of-articulated->

2.5 Recording datasets for SLAM-like methods

In contrast to methods that have prior knowledge about the object they track, SLAM-like tracking and reconstruction methods introduce an arbitrary scale by assumptions about certain algorithmically necessary distances. For example, PTAM [KM07] assumes a distance of 10 cm between the camera centers associated to the first pair of images used to create the map of the environment.

The same image sequence given to two different SLAM algorithms thus usually results in differently scaled reconstructions of the environment which may also be associated to two different coordinate systems. It is only possible to evaluate SLAM methods by aligning their trajectories to a common coordinate frame. Compared to the workflow used in the last chapter, we now chose to only use the transform ${}^{effector}\mathbf{T}_{base}$ and let the alignment of the trajectories take care of the three explicitly calibrated transformations used before (see figure 2.6). This removes most of the calibration steps, as now only *intrinsic*s of camera and measurement arm have to be dealt with.

The proposed ground truth acquisition records unmodified real-world objects as *e.g.* the industrial object shown in figure 2.12. For obtaining a realistic performance of the algorithms, we chose not to introduce any fiducials as they typically affect *e.g.* feature-based SLAM-like methods because of their high-contrast appearance. We previously used fiducials both for establishing a pre-defined coordinate system and for synchronization. The former step has been merged into the alignment of the poses. We still assume that the offset t_{offset} between poses and images is constant. One synchronization strategy could be conducted by (re-)adding fiducials to a part of the scene that is not visible when recording the object of interest. Then, the aforementioned approach for synchronization can be used whenever the markers are in the field of view. To minimize interference with the SLAM method, separate sections of the recorded sequences, *e.g.* at the beginning and in the end, can be dedicated to contain the fiducials.

The drawback of this approach would be that there again is either the full extrinsic calibration of the system necessary or that the coordinate system of the marker also has to be obtained in a separate step. Further a part of each sequence has to be reserved for recording images of fiducials. However, the advantage of this method is that only seven degrees of freedom remain during the evaluation (6 for pose and 1 for scale).

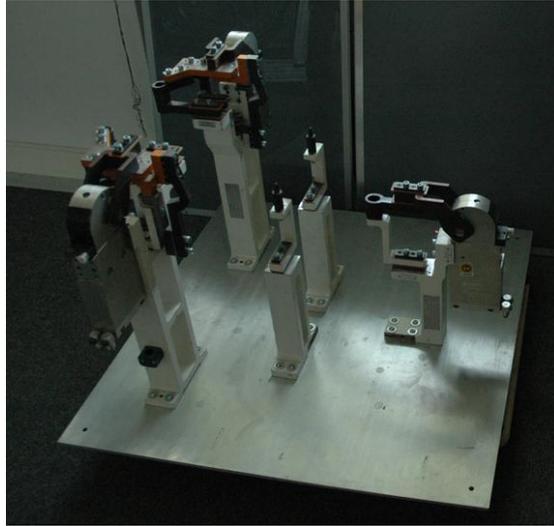


Figure 2.12: Clamping fixture used for evaluating SLAM-like methods on typical industrial objects.

For the evaluation of our work on RGB-D images (see chapter 4), we chose to record the data streams at their native speeds, *i.e.* images at 30 Hz and poses at 250 Hz. As detailed in section 2.7.2, we use a combined alignment/evaluation optimization approach that consists of a 7-DOF initial linear alignment with a 8-DOF non-linear robust refinement which then additionally included the timestamp offset t_{offset} .

2.6 Designing tracking datasets

After the acquisition and synchronization of sequences with ground truth data in general has been shown, we now present the datasets we created for our work on evaluating planar template-based methods [LBMN09, LBMN11], deformable template-based tracking [LBI11] and live tracking and meshing using a RGB-D camera [LHIB11].

2.6.1 Planar targets

The motivation for creating the first dataset was to allow a fair comparison between a wide range of template-based tracking algorithms. There are algorithms that use corners, edges and whole regions of an image. We tried to make the dataset balanced and focused on real world usage. In the following, we describe how we tried to achieve these goals.

One of the most basic and essential tasks for optical tracking is the tracking of planar structures. In fact, in order to be able to determine the relative position and orientation of the camera with respect to different objects of the environment in real-time, most of the model-free detection and tracking algorithms have to assume that the objects are locally or piecewise planar. That is why we used planar targets in the first dataset. The targets were chosen to represent a broad overview of all types of possible tracking targets. We use a classification of four groups, namely “Low Texture”, “High Texture”, “Repetitive Texture” and “Normal Texture”, meaning somewhere in between. Each class is represented in the dataset by two targets, shown in figure 2.13. The targets were downloaded from an online image database, see appendix D. We did not track any of these targets in advance in order not to make the selection of the targets biased to any of the evaluated methods.

The “Low Texture” group consists of images of road signs which are composed of two distinct colors, large uniform areas and thus large edges. In the “Repetitive” group there are images of electronic boards, one image with mainly large, one with mainly small components. An image of a car and of a cityscape are in the group of the “Normal” reference targets. Finally, the group of “Highly Textured” images is composed of an image of a wall made of different sized stones and of an image with English lawn which features many small structures.



Figure 2.13: The reference targets used in our dataset. *From left:* Low, Repetitive, Normal, High Texturedness. Image credits are in appendix D.

Each target image was resampled to a resolution of (800×600) before printing. The algorithms later were provided with the inner (640×480) area of these target images which occurred to be an appropriate resolution as VGA cameras are widespread. The targets were printed with a color laser printer. Both the printer and the camera were not specifically color calibrated such that the captured images of the printed targets match the reference targets exactly. These steps were skipped on purpose as they are typically skipped in real world scenarios as well and as they are hard to emulate with synthetic images. The printed pages were glued onto foam board which was later fixed on a table rigidly connected with the base of the measurement arm as shown in figure 2.6.

Next, we designed the dynamic part of the evaluation, *i.e.* the general types of motion of the camera. We chose to include five different types for each target into the dataset: “Angle”, “Range”, “Fast Far”, “Fast Close” and “Illumination”. In the sequences of type “Angle”, we focus on varying the angle between the normal of the reference target and the optical axis of the camera between 0° and approximately 80° while trying to keep the distance to the target constant. The target covers around 10–30% of the image.

The “Range” sequences focus on the size of the reference image in the camera image. The maximum distance of the camera to the target resulted in a visible area of the reference target of about (130×100) pixels or 4% of the original area. The maximum of the visible area is around 100%, *i.e.* the reference template occupied the whole camera image. The reference template is always facing the camera near fronto-parallel in these sequences, there is mainly rotation around the normal of the target.

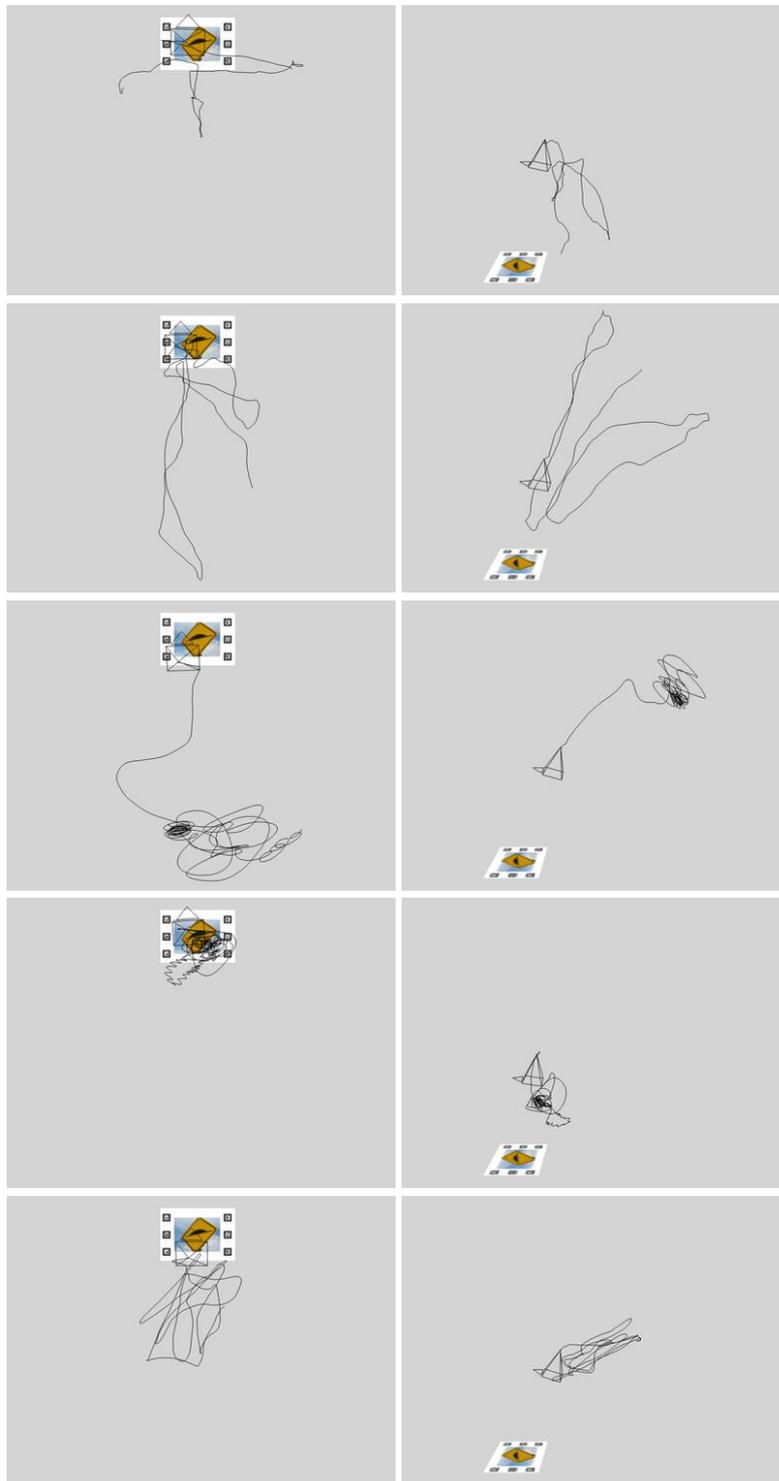


Figure 2.14: Camera trajectories from Target “Bump”. From top: “Angle”, “Range”, “Fast Far”, “Fast Close” and “Illumination”.

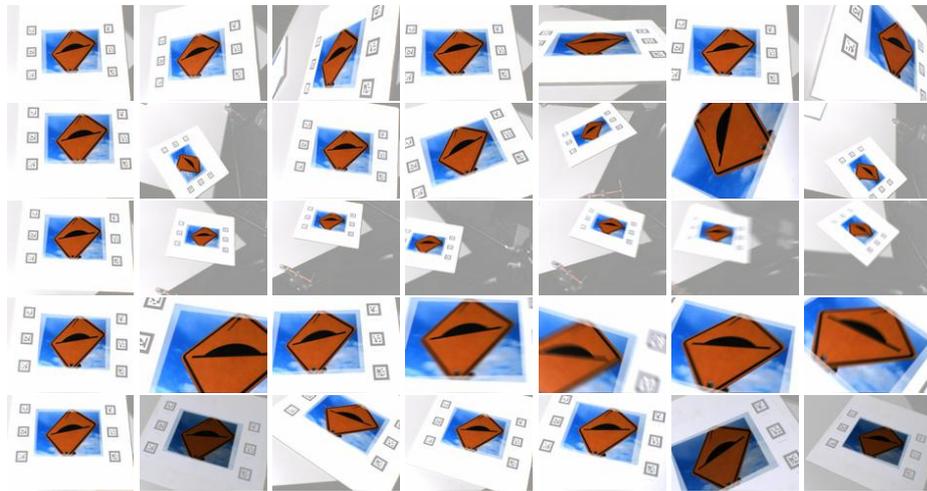


Figure 2.15: Sample frames for Target “Bump Sign”. From top: Sequence “Angle”, “Range”, “Fast Far”, “Fast Close” and “Lighting”. Background is dimmed.

The next type of sequences is “Fast Far”; here the camera is moved away from the target until it covers again an area of approximately 4% of the image. Then, we move the camera with increasing speed which results in big inter-frame motion. Towards the end of these sequences, the effect of motion blur shows strongly. These motions are also applied to the “Fast Close” sequences, the only difference here being that the reference image typically covers 60% or more of the image where parts of the targets go often outside the image.

The last type of sequences we recorded, “Illumination”, varies the lighting conditions of the scene while the camera is moving slowly. For this, we switch off and on again two sets of fluorescent tubes during the sequence. Additionally, a shadow is cast onto the reference target by a waving hand. This effectively changes the mean brightness of the camera image as the exposure time and aperture of the camera were held constant. In this scenario, the target is always covering more than 15% of the camera image. Selected frames from the sequences can be seen in figures 2.15 and 2.16, representative motion of the camera for each type of motion is shown in figure 2.14.

After the synchronization was completed, the average RMS reprojection error for all sequences was 0.86 pixels. This error also incorporates all errors from the internal and external calibrations. The “Fast Close” sequences typically have the highest residual error (mean 1.54 pixels) due to the motion blur and the large size

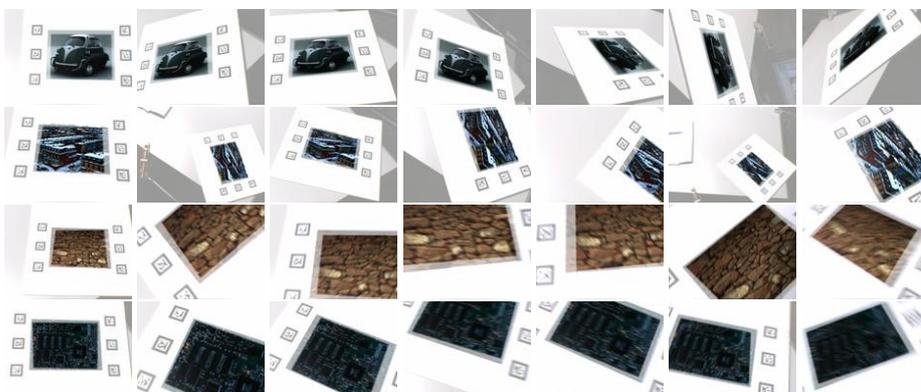


Figure 2.16: *From top:* Sample frames for Targets “Isetta”, “Philadelphia”, “Wall” and “Lucent”. Background is dimmed.



Figure 2.17: Post-processing. *From left:* captured image, undistorted image without background, final image with randomized borders.

of the reference target in the image. The lowest residual error (mean 0.54 pixels) was observed for the “Illumination” sequences.

To prevent algorithms from being distracted by a cluttered background, we chose to remove the background, *i.e.* any image area not covered by the template. In addition to that, we also removed the original borders of the reference target to prevent algorithms from simply using the image borders instead of the template image itself. The original borders of the (800×600) reference target were replaced by randomized borders, but at the same time we made sure that the (640×480) image the algorithms were given is not cut by the new randomized borders. These steps are visualized in figure 2.17.

While removing the background is clearly a simplification, we chose this approach with the current datasets to focus on the best performance of the algorithms possible with images of the tracked object coming from a real camera. In later datasets we could either add a cluttered background to the real scene

(but leaving the necessary area for the fiducials de-cluttered) or add a virtual cluttered background after recording.

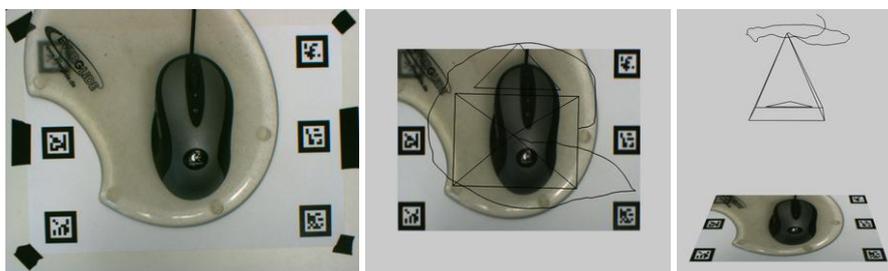


Figure 2.18: Target used for the evaluation of the dense deformable tracking approach of chapter 3. *Left*: The reference image given to the algorithms. *Middle and right*: Two views on the ground truth trajectory of the sequence.

2.6.2 Non-planar convex targets

For our work on deformable template tracking [LBI11], we applied the calibration steps described in section 2.4. This time, the target was not a planar print but a computer mouse on a mouse pad, featuring very low texture and a convex shape. The target and motion is shown in figure 2.18. As will be discussed in chapter 3, the algorithm was designed to simultaneously deform an initially planar template while tracking the pose of a moving camera.

As this approach is closer to reconstruction and SLAM rather than traditional non-reconstructing template-tracking, we chose the motion of the camera such that it is suitable for reconstruction purposes. In contrast to the motion of the previous section it does not include fast, sudden motion. We captured a single sequence that starts with an almost fronto-planar view of the target and slowly moves on a spiral-like track outwards, see figure 2.18. To create a template image of the non-planar target, we rectified the first image where the image plane is almost fronto-planar. We used bilinear interpolation where the pixel correspondences were established based on the ground truth pose.

We also removed the fiducials from the images since corner detectors employed by most feature-based tracking methods would mainly extract the feature points on the markers. To analyze the influence of blur, we additionally blurred the images artificially. The results of the evaluation will be shown next to the presentation of the algorithm in chapter 3.

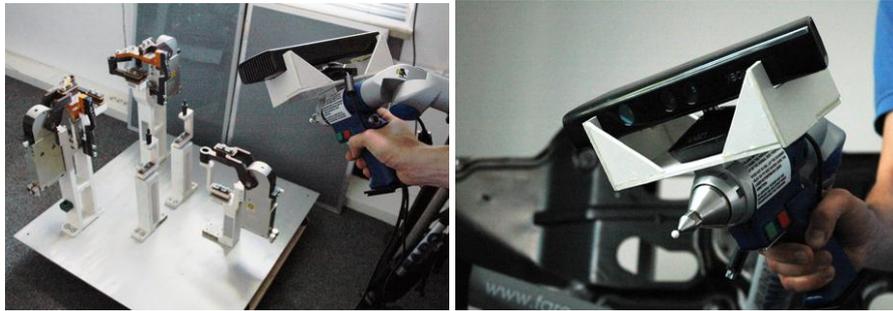


Figure 2.19: Physical setup for the 3D RGB-D ground truth data.

2.6.3 Non-planar targets of arbitrary shape

Given arbitrary objects of interest for tracking and reconstruction, there is a multitude of possible motions depending on the specific AR scenario. Some examples of motions include very fast, irregular motion that may be found when the user is engaged in an AR game. More gentle motions with long stalls may be of interest as well, as they are typical when AR is used for design concepts or discrepancy checks. A further typical type of motion includes a continuous ‘scanning’ which is used for obtaining 3D models of parts of the environment.

We created the 3D dataset for evaluating our method for RGB-D tracking and meshing (described in chapter 4) on a challenging industrial object. Figure 2.19 shows the hardware setup. The AVT camera mounted on the end effector was exchanged against a Microsoft Kinect which tries to associate a depth value to every pixel of the (640×480) color image acquired at 30 Hz and uses a rolling shutter.

Due to the size of the industrial object and the working volume of the measurement arm, we recorded four sequences with continuous slow movement around the object. Figure 2.20 shows excerpts of the sequences. As can be seen, the object covered usually more than 50% of the captured image.

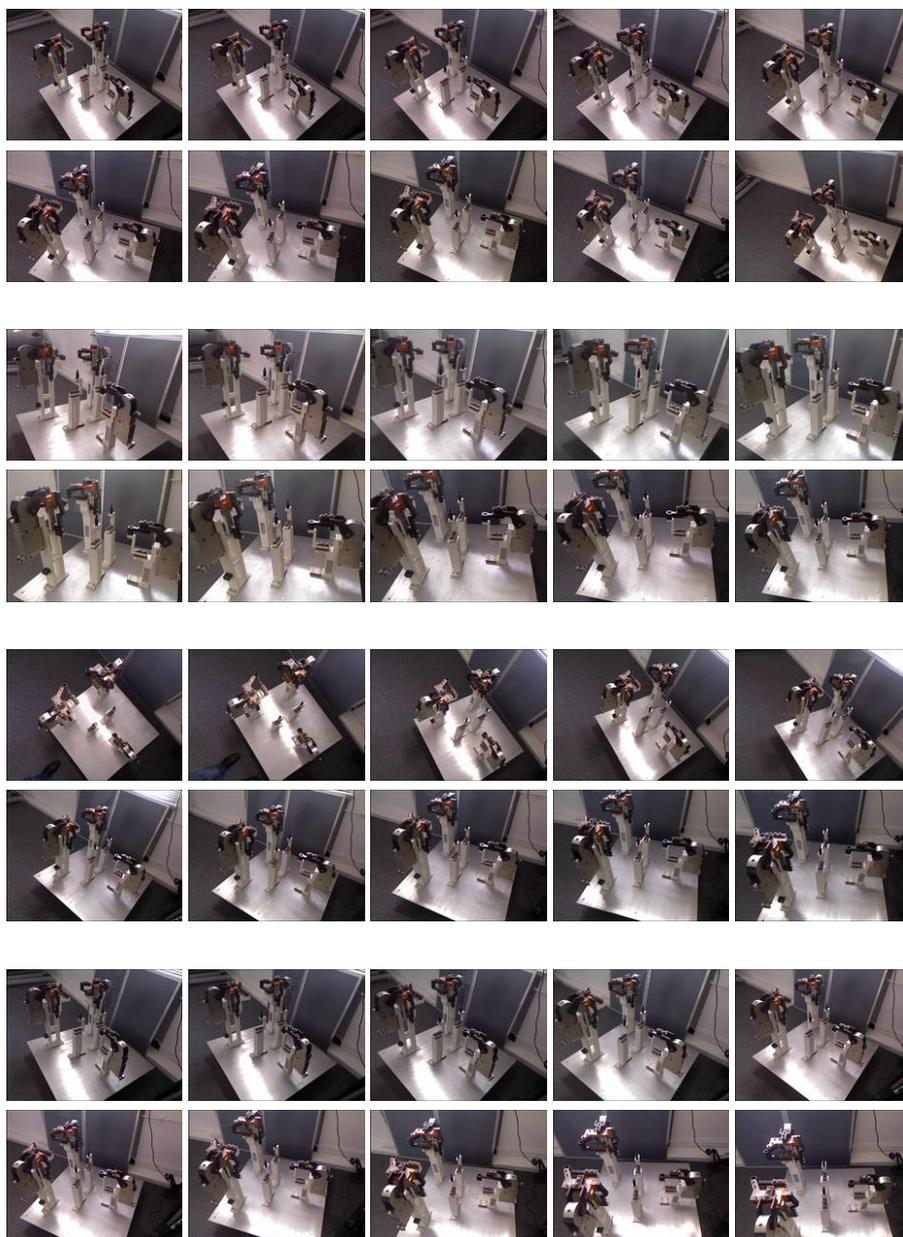


Figure 2.20: Excerpt of the four RGB-D sequences, starting from top with images from sequence 1. Each sequence consists of 300 images of which every 30th is shown.

2.7 Evaluation methodologies

In this section, we first describe which method we used to evaluate different template-based tracking algorithms, then how we propose to compare different SLAM-like tracking methods. The results of the template-based tracking algorithms are presented in the next section, while the results of the non-planar tracking methods are presented right next to the presentation of the algorithms in chapters 3 and 4.

2.7.1 Evaluating template-based methods

Within the template-based methods, we differentiate between *inter-frame* and *detection-based* tracking algorithms: Inter-frame tracking algorithms require an initial pose or homography, but then should be able to track the reference target in the images with high precision and over many consecutive frames. Detection-based methods on the other hand typically try to establish a relatively coarse initial pose or homography.

Similar to the Middlebury datasets [BSL⁺07], we do not publish the ground truth for every frame. To support inter-frame tracking algorithms, we provide 2D–2D correspondences of the XGA boundaries $[\pm 512 \pm 384]^\top$ between the current image and the reference frame for the first and every 250th frame in the dataset. Thus it is possible to initialize an inter-frame frame algorithm five times per sequence in case it lost tracking.

Detection-based tracking algorithms were provided with the reference images only. They had to locate the target in the images of the sequences without any further prior knowledge about the pose or homography. The missing need of an approximate initial pose is a benefit over the inter-frame tracking algorithms, they are usually initialized from detection algorithms. The total number of successfully detected frames is more important for this type of algorithms than the number of consecutively tracked frames.

We evaluated four tracking algorithms: FERNS [OFL07], ESM [BM07] and two algorithms that use SIFT [Low04] and respectively SURF [BTG06] to obtain 2D–3D correspondences for estimating the camera pose. The (640×480) uncompressed intensity images of the centered inner areas of the reference targets were used as template (marked white in figure 2.21). Next, we setup each algorithm

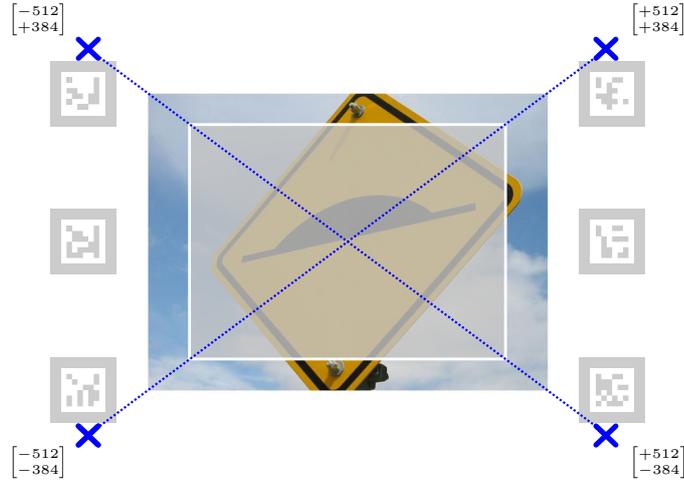


Figure 2.21: A reference target with the six fiducials. The inner area of each template (marked with white dimmed frame) is provided as (640×480) reference image to the algorithms. The evaluation is based on the reprojection error of four points on the XGA boundary (marked with blue crosses).

with the templates: For SIFT and SURF, features were detected in the reference image and descriptors were extracted at these positions. For FERNS, classifiers were trained. For ESM, the intensity image was cast to floating point values.

After that, the tracking methods were run on the sequences. The evaluation is based on four reference points which are placed on the diagonal lines of the reference images (marked with blue crosses in figure 2.21). The reference points are at the XGA resolution boundaries, *i.e.* at $[\pm 512 \pm 384]^T$. Their corresponding 3D positions in our setting are

$$\bar{\mathbf{x}}_1 = \begin{bmatrix} \bar{x}_0 \\ \bar{y}_0 \\ 0 \\ 1 \end{bmatrix} \quad \bar{\mathbf{x}}_2 = \begin{bmatrix} -\bar{x}_0 \\ \bar{y}_0 \\ 0 \\ 1 \end{bmatrix} \quad \bar{\mathbf{x}}_3 = \begin{bmatrix} -\bar{x}_0 \\ -\bar{y}_0 \\ 0 \\ 1 \end{bmatrix} \quad \bar{\mathbf{x}}_4 = \begin{bmatrix} \bar{x}_0 \\ -\bar{y}_0 \\ 0 \\ 1 \end{bmatrix} \quad (2.25)$$

with $\bar{x}_0 = 117.12$ mm and $\bar{y}_0 = 87.84$ mm. Since the effect of the perceived errors varies depending on the position and orientation of the tracked object, using errors in the image domain, *i.e.* in pixel, is beneficial for AR over using metric errors, *i.e.* meters for the translation and radians or degrees for the rotation.

Errors in the image domain better reflect whether the augmentation would be correctly perceived or not and whether it would allow a seamless integration, *i.e.* whether the augmentation would appear as close as possible to the desired

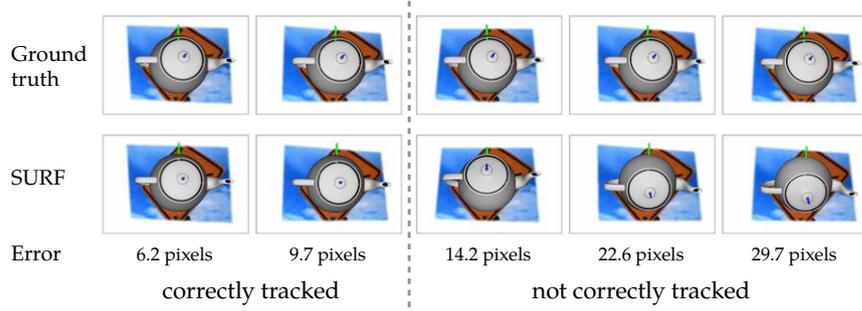


Figure 2.22: Visualization of ${}^{camera}\mathbf{T}_{target}$ from the ground truth dataset and SURF. The images are augmented with a teapot. A threshold of 10 pixels is used for classification of “correctly” tracked images.

position *in the image*. For every image \mathcal{I}_i per sequence, the RMS distance y_i of each imaged reference point $\mathbf{u}_{i,j}$ to the ground truth point $\bar{\mathbf{u}}_{i,j}$ was computed as

$$y_i = \sqrt{\frac{1}{4} \sum_{j=1}^4 \|\mathbf{u}_{i,j} - \bar{\mathbf{u}}_{i,j}\|_2^2} \quad (2.26)$$

with the ground truth 2D coordinate $\bar{\mathbf{u}}_{i,j}$ as

$$\bar{\mathbf{u}}_{i,j} = \mathbf{u}(t_i + t_{offset}, \bar{\mathbf{x}}_j) \quad (2.27)$$

where $\mathbf{u}(\cdot)$ is defined in equation (2.23). After computing these errors for each image of a sequence, all frames with $y_i \geq 10$ pixels are removed as we regard a higher RMS error as sign that the tracking algorithm lost the target. Based on the filtered results, we compute the ratio of tracked frames and analyze the distribution of the error.

Figure 2.22 shows a comparison of poses originating from matched SURF features to ground truth poses. The former are ordered by reprojection error, from 6.2 to 29.7 pixels. While the first two images with errors below 10 pixels are still very similar to their corresponding ground truth images in the upper row, with increasing reprojection error also the dissimilarity increases. We observed that until an error of 10 pixels, the tracking result looks reasonable when the template is visible at least in 15% of the image which is fulfilled in the majority of the sequences.

2.7.2 Evaluating SLAM-like methods

When evaluating methods that do not have any prior knowledge about the scene, we directly used the trajectory of the estimated camera pose and the movement of the measurement arm. As mentioned before, we record the image stream and pose stream at their native frequencies and perform a joint synchronization/evaluation at the same time using the results of an algorithm. We obtain an initial estimation of the scale and Euclidean transformation by assuming that there is no offset in the timestamps of the data from camera and the measurement arm. We create 3D–3D correspondences between the camera center and the end effector and use these as input to a closed-form method from Umeyama [Ume91] that computes the estimates of the initial transformation and scale. As before, the poses of the measurement arm were interpolated on the $\mathbb{SE}(3)$ manifold to match the timestamps of the images.

Umeyama’s method was originally designed to align point clouds, it is optimal when the assumption holds that the correspondences contain errors exclusively belonging to a normal distribution. But as we are using it to align trajectories where one should be evaluated to the other, and additionally still have to synchronize the measurements, we use the result only as first step in the alignment process.

In a second step, we take both the possibility of outliers into account and additionally also search for the offset of the timestamps. We use a Nelder-Mead simplex [PCB02] to minimize the distance of the corresponding 3D points. The cost function was parameterized by the six degree of freedom Euclidean transformation (using Lie algebra for parameterizing the rotation), the scale of the trajectory and the offset of the timestamps. Noise and outliers are handled by re-weighting the error with the Tukey M-Estimator [Ste99].

Thus, “ground truth” can be generated for every pair of algorithm and sequence individually. However, we observed that the offset of the timestamp was in the order of ± 50 microseconds after successfully aligning the results of our proposed method on RGB-D SLAM. The timestamp offset seemed to have no substantial influence in the optimization. Since we wanted to compare multiple algorithms on the same sequence, we generated common ground truth camera poses using a fixed time offset.

The error measure used for the evaluation consists of a RMS error of the translation in millimeters. For the rotation, we used an angular RMS error

by using the difference of rotations in axis-times-angle representation which represents an intuitive error in radians. Although we just argued in the previous section that an error measured in pixel would be preferable for AR in general, we chose to use error measures based directly on the trajectory as the recorded sequences of the 3D object were mostly at approximately the same distance to the object. The recorded sequences had roughly the same scale and gave rise to comparable results, *i.e.* the errors are on similar scales. The results of the evaluation can be found in chapter 4.

2.8 Evaluation of four template-based methods

We used the original implementations of ESM, FERNS and SURF for the evaluation. For SIFT we used the implementation from Vedaldi and Fulkerson [VF08]. The majority of the parameters were left at their authors' default settings. We only constrained SURF and FERNS to use the 800 strongest points, a number we had to provide to the implementations that seemed high enough to not degrade their performance.

The poses used in the evaluation were computed from 2 D–3 D correspondences using a non-linear Levenberg-Marquardt minimization of the reprojection error that was initialized by POSIT [DD95]. To obtain these correspondences from the algorithms based on SIFT and SURF, we used nearest-neighbor matching of the descriptors to generate initial 2 D–3 D correspondences, then removed outliers via a RANSAC [FB81]-scheme. As FERNS has integrated matching and outlier removal, we directly used the filtered correspondences. For ESM, we similarly used the output homography to project corners of the reference template into the current frame.

The targets were evaluated in the order shown in figure 2.13, *i.e.* “Low”, “Repetitive”, “Normal”, “High Texturedness”. Each target was evaluated following the discussed types of motion, we used the order “Angle”, “Range”, “Fast Far”, “Fast Close”, “Illumination”. The results of the evaluation indicated that SIFT is, most of the time, outperforming FERNS and SURF in terms of accuracy and percentage of tracked frames. However, it should be mentioned that the evaluation of SIFT took more than 2.5 days (approximately 3 s per frame) to compute whereas FERNS, SURF and ESM finished in less than 6 hours each.

The focus of this evaluation was primarily to see whether the targets and the chosen sequences per target were suitable for building a dataset that is both challenging and at the same time not undoable so that it can be useful to the computer vision community. Thus, we mainly focused on the accuracy of the algorithms in close-to-ideal situations. The timings were not considered since the best possible frame rates of the algorithms are generally achieved with extensive fine-tuning of parameters and this was of minor interest to us.

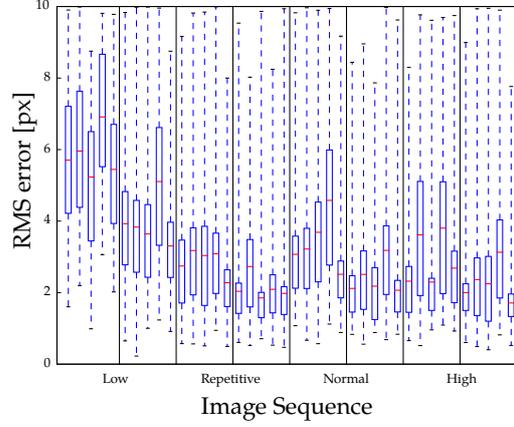
The percentage of correctly tracked frames is presented in figure 2.23 and figure 2.24. Feature point approaches typically select positions with high corneriness. The evaluation showed that ESM often depends on the selected area

to be tracked. In contrast to the selective feature points approaches, ESM gives the same weight to every pixel in its area. This can severely degrade the accuracy if *e.g.* the border of the image is approximately uniform. Thus, to make the comparison fair, we manually selected patches in the low texture targets for ESM. After that, it tracked the extremely low textured yellow road sign for 100% of the “Angle” sequence and surpassed the other algorithms in terms of accuracy. Concerning low texturedness, both FERNS and SURF showed a better performance on the slightly more textured stop sign target.

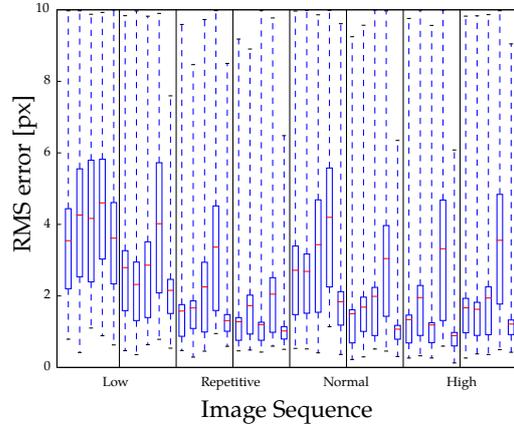
The reason for the performance of SURF for the first target is that SURF does not find sufficient feature points on the yellow traffic sign, the same again applies to the grass target which for ESM also turned out to be an extremely difficult target. FERNS was in general very well adapted to the “Angle” sequences which might come from the explicit training phase that warps the reference targets numerous times.

The “Fast Close” sequences with large amounts of motion blur were the most difficult to detect for all four algorithms, whereas “Range” and “Illumination” sequences were often correctly detected. Figures 2.23 and 2.24 also show the RMS errors for all targets, sequences and algorithm as box-and-whiskers-like diagram. The whiskers mark the minimum and maximum error while the box spans from the first to the third quartile, the mean is given via the red horizontal line. The targets per target group are separated by a vertical black line. In general, the “Fast Close” sequences were detected with the largest error per target while “Illumination” yielded, most of the time, the lowest error.

2.8 EVALUATION OF FOUR TEMPLATE-BASED METHODS

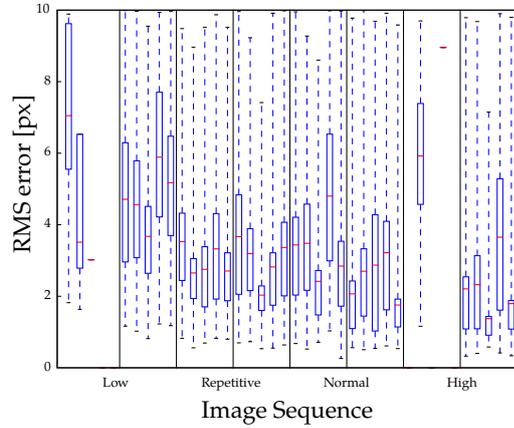


<i>FERNS</i>	Angle	Range	Fast Far	Fast Close	Illumination
Low	17.08 %	8.08 %	1.58 %	3.75 %	6.58 %
	47.33 %	71.00 %	22.92 %	40.50 %	76.08 %
Repetitive	36.42 %	65.17 %	15.42 %	48.50 %	91.83 %
	42.50 %	45.17 %	6.25 %	50.00 %	81.33 %
Normal	69.50 %	80.58 %	24.92 %	68.00 %	95.92 %
	38.75 %	53.08 %	9.00 %	64.67 %	81.67 %
High	34.92 %	38.17 %	5.92 %	16.00 %	31.58 %
	71.75 %	61.50 %	13.42 %	63.00 %	96.92 %

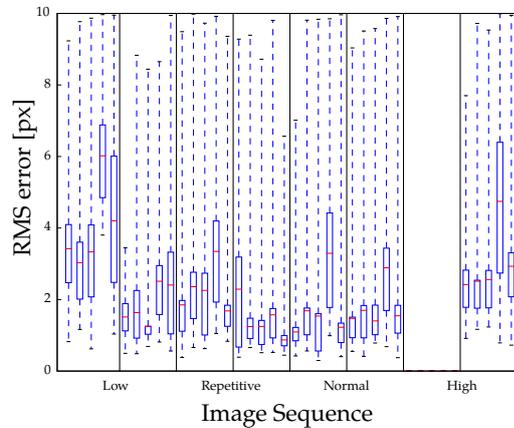


<i>SIFT</i>	Angle	Range	Fast Far	Fast Close	Illumination
Low	47.25 %	49.08 %	10.33 %	19.58 %	59.17 %
	36.08 %	95.42 %	25.50 %	55.58 %	99.75 %
Repetitive	59.00 %	99.33 %	43.33 %	71.92 %	100.00 %
	69.50 %	95.67 %	15.17 %	62.83 %	98.17 %
Normal	63.50 %	84.25 %	21.75 %	55.17 %	96.08 %
	53.00 %	96.08 %	31.67 %	77.67 %	99.58 %
High	66.83 %	85.08 %	18.33 %	37.42 %	97.00 %
	79.50 %	94.75 %	31.42 %	72.75 %	99.50 %

Figure 2.23: *Diagrams*: Distribution of the RMS error for each sequence, only successfully tracked frames were taken into account. The whiskers denote minimum and maximum, the box spans from first to third quartile; a red line segment shows the mean RMS error. *Tables*: Ratio of successfully tracked images (with $y_i < 10$ pixel).



<i>SURF</i>	Angle	Range	Fast Far	Fast Close	Illumination
Low	0.50 %	0.33 %	0.08 %	0.00 %	0.00 %
	27.17 %	67.00 %	11.83 %	33.50 %	55.17 %
Repetitive	16.50 %	38.42 %	5.25 %	47.33 %	41.00 %
	25.58 %	50.00 %	6.08 %	54.17 %	49.50 %
Normal	37.92 %	50.17 %	6.17 %	50.33 %	67.75 %
	45.33 %	70.75 %	14.25 %	69.67 %	89.58 %
High	0.00 %	7.75 %	0.00 %	0.08 %	0.00 %
	64.00 %	44.42 %	6.50 %	51.50 %	72.33 %



<i>ESM</i>	Angle	Range	Fast Far	Fast Close	Illumination
Low	100.00 %	92.33 %	35.00 %	21.58 %	71.08 %
	100.00 %	64.17 %	10.58 %	26.83 %	56.25 %
Repetitive	61.92 %	50.42 %	22.50 %	50.17 %	34.50 %
	2.92 %	11.33 %	6.83 %	35.83 %	11.33 %
Normal	95.42 %	77.75 %	7.50 %	67.08 %	76.75 %
	99.58 %	99.00 %	15.67 %	86.75 %	90.67 %
High	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
	100.00 %	61.42 %	22.83 %	45.50 %	79.67 %

Figure 2.24: Continuation of figure 2.23

DENSE DEFORMABLE TEMPLATE TRACKING

This chapter proposes and evaluates a method for simultaneously tracking the pose of a moving camera and reconstructing a non-planar, convex object in real-time given only a *template* image of the object.

3.1 Motivation

Template tracking is one of the fundamental problems in computer vision. A multitude of techniques have been proposed in the literature. Since the seminal work of Lucas and Kanade [LK81] and multiple subsequent works [JD02, BM04, BM06], researchers mainly concentrate on planar templates and estimate camera motion by minimizing the energy defined by the difference of the image intensities in subsequent frames. In general, the image intensity constancy assumption is made, *i.e.* the same physical point is assumed to have the same intensity value across different frames.

The applications of template tracking are wide and include, but are not limited to, vision-based control, human-computer interfaces, augmented reality (as shown in figure 3.1), robotics, surveillance, medical imaging and visual reconstruction. In many applications, the planarity assumption is good enough, but in general that is not the case. For that reason Bartoli and Zisserman [BZ04] and Silveira and Malis [SM10] considered computing 2D warpings of the reference templates while tracking them. The real depth and camera motion are then obtained by decomposing the estimated warpings.



Figure 3.1: Template tracking used for Augmented Reality: By holding a LEGO box in front of a camera, a virtual animation of the fully assembled LEGO model appears on top of it.

Motivated by the fact that the world is not planar and driven by the emerging needs of simultaneous recovery of the structures and motion of the camera, this chapter addresses the problem of simultaneous tracking and reconstruction of a non-planar template in real-time. The model of the template is represented as a triangular mesh. We start with a planar shape and simultaneously recover 6 DOF camera motion and deform the shape such that the underlying 3D structure is approximately recovered. As we use all pixels of the template, the object does not necessarily have to be well textured and contain many feature points. This is different from classical Structure from Motion (SfM) and SLAM techniques that primarily rely on sparse feature points, such as *e.g.* Klein and Murray's PTAM [KM07]. While sparse methods perform very well in terms of runtime and precision, they generally depend on the amount of the observed features and tend to be quite sensitive to blur.

Unlike methods which rely on prior deformation models [SUF08, SF09] and assume fixed camera position, we solve for camera motion and do not impose any constraints on the model deformation. Therefore, we can equally reconstruct and track templates that are smooth or have creases. However, since the problem is ill-posed, we have made certain assumptions: we use templates of a predefined size, assume that in its reference position the entire template is visible and is not self-occluded, and we finally restrict mesh vertices to only move along the camera rays, thus having one degree of freedom per vertex.

In the remainder of the chapter, we first discuss related works in more detail, then describe our method and present experimental results.

3.2 Related work

Template tracking has always been assuming the planarity of the object of interest to be tracked. Since Lucas and Kanade's original work [LK81], the real-time constraint was enforced and became standard in recent works such as Baker and Matthews' Inverse Compositional (IC [BM04]) or Benhimane and Malis' Efficient Second-Order Minimization (ESM [BM07]) methods. Improvements in convergence speed and robustness in the calibrated camera setting were especially achieved by the method of Benhimane and Malis [BM06]. For those reasons, we in part relied on their method.

Other researchers also proposed to find deformations of an object in a sequence of acquired images. These methods generally consist of estimation of the parameters of the warping function that registers the reference image, in which the object is mainly planar, to the input image where the object is deformed. Pilet *et al.* [PLF08] and Gay-Bellile *et al.* [GBBS09] relied on feature points. While the former can deal with a huge amount of outliers, the latter is relatively sensitive to them.

Datta *et al.* [DSK08] used affine warps and integrated the idea of articulated points as hard constraints into the minimization, *i.e.* they force patches to move according to their connectivity. Hilsmann *et al.* [HSE10] re-textured the surface of a deforming object realistically by estimating both the changes in geometry and photometry. They also explicitly model external occlusions to further improve the quality of the augmentation.

Silveira and Malis [SM10] use 2D warps and present a generic framework for template tracking which can undergo deformations. In all of these cases, the warping is done in image space and therefore does not provide a 3D shape, but instead 2D warpings of the images as in deformable registration. To recover the 3D shape, the recovered 2D warpings are decomposed into a rigid motion and according depths.

On a separate track, deformable surface tracking from monocular videos has been developed. Because of the inherent ambiguity, deformation models have been introduced to constrain deformations of particular objects like *e.g.* paper and clothes [SUF08, SF09, VSTF09]. These approaches generally output the 3D surface meshes of the deformed object. However, they do not provide the relative

camera/object motion in the image sequence, require heavily textured objects and generally do not work in real-time.

Simultaneous recovery of the camera motion and the 3D shape is also related to SfM methods based either on bundle adjustment [TMHF00] or filtering, such as Klein and Murray's PTAM [KM07] or Davison's MonoSLAM [DRMS07] respectively. A theoretical comparison of both directions was conducted by Strasdat [SMD10]. Both PTAM and MonoSLAM strictly rely on point-based image features and incremental (if real-time) reconstruction of an observed scene, while neither of them operates on the dense pixel level. The system proposed by Newcombe and Davidson [ND10] indeed produces a dense reconstruction using a movable camera. It relies on PTAM to precisely recover the motion of the camera, furthermore also PTAM's sparse feature map is used to initialize a GPU-based dense optical flow method [WPZ⁺09].

Most of the previously mentioned methods are using feature points and/or define constraints on the possible model deformations. Relying on features usually implies that the observed object has to be well textured. Instead of using a set of extracted feature points in the image, we share the belief that a higher accuracy can be achieved when using all available pixels of the template – which in turn enables tracking of low textured templates.

Recently, Newcombe *et al.* [NLD11] presented a system for densely tracking and mapping that no longer needs PTAM's sparse feature tracking as basis. The dense reconstructed model of the environment is now directly tracked, and thus unprecedented results are achieved also in presence of strong motion blur and low textured objects. However, due to the computational complexity of this approach, it has to be run on a modern GPU for real-time performance. In this chapter, we present a method that requires rather low computational efforts and thus potentially also could be run on less powerful mobile devices.

3.3 Method

Our method is based on the idea of tracking the camera pose relative to a reference image \mathcal{I}^* of an object. Simultaneously, we estimate the shape of the object as seen in \mathcal{I}^* , using a triangular mesh M as model for the shape and initially assuming it to be planar. As the camera moves, the mesh deforms towards the true shape of the object.

The pose of the camera that was used to take the reference image is used as origin of the coordinate system, *i.e.* the pose assigned to \mathcal{I}^* is a (4×4) identity matrix. When a new image \mathcal{I} from a sequence or live camera feed becomes available, the pose of the camera where \mathcal{I} was captured should be estimated and the mesh updated. For simplicity, we denote this pose as \mathbf{T} throughout this section. As we typically deal with an image sequence, we assume that we have the estimates for pose and mesh of the last processed frame available as $\widehat{\mathbf{T}}$ and \widehat{M} .

We further assume that, ignoring occlusion and drastic lighting changes, the reference image \mathcal{I}^* can be constructed from \mathcal{I} by back-warping each face f given the true camera pose and the recovered mesh. As we only know the approximations $\widehat{\mathbf{T}}$ and \widehat{M} , we produce an estimated image $\widehat{\mathcal{I}}^*$ by applying a homography \mathbf{G} to each face of the mesh. We then compute the photometric error y as the difference between \mathcal{I}^* and $\widehat{\mathcal{I}}^*$ and minimize it. An overview of the method is shown in figure 3.2.

We use a piece-wise planar mesh where each face f has a corresponding normal $\mathbf{n}_f^* = \frac{1}{d_f^*} [n_x^* \ n_y^* \ n_z^*]^\top$. Please note that the normal \mathbf{n}_f^* is scaled by the inverse of the distance d_f^* of the face to the camera center \mathbf{c}^* in the reference frame. The pixels associated with the face are interpreted to lie on the plane defined by

$$\begin{bmatrix} \mathbf{n}_f^* \\ -1 \end{bmatrix}^\top \mathbf{x} = 0 \quad (3.1)$$

where \mathbf{x} is a 3D point in normalized homogeneous coordinates. Each face can be warped from the image \mathcal{I} to the frame of the reference image \mathcal{I}^* using the homography

$$\mathbf{G}(\mathbf{T}, \mathbf{n}_f^*) = \mathbf{K} [\mathbf{R} + \mathbf{t}\mathbf{n}_f^{*\top}] \mathbf{K}^{-1} \mathbf{G}_f \quad (3.2)$$

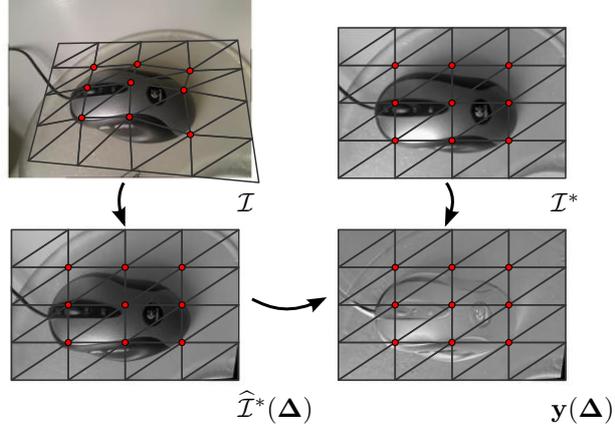


Figure 3.2: Overview of the method: The mesh is overlaid onto the object. Out of the camera image \mathcal{I} the estimate of the reference image $\hat{\mathcal{I}}^*(\Delta)$ is unwarped. The error $\mathbf{y}(\Delta) = \hat{\mathcal{I}}^*(\Delta) - \mathcal{I}^*$ is iteratively minimized to obtain object shape and camera pose.

where \mathbf{K} is a (3×3) matrix with the known camera intrinsics (*c.f.* equation (2.2)) and \mathbf{R} and \mathbf{t} are contained in \mathbf{T} (*c.f.* equation (A.2)). To use multiple faces in the optimization, the homography \mathbf{G}_f is used to apply a 2D translation within the reference image \mathcal{I}^* as

$$\mathbf{G}_f = \begin{bmatrix} 1 & 0 & u_f^* \\ 0 & 1 & v_f^* \\ 0 & 0 & 1 \end{bmatrix}. \quad (3.3)$$

At the beginning of each iteration, we warp each face from the current frame to the reference frame to compute the photometric error. We then compute and eventually apply an update Δ to decrease the error. In the following, each step of the minimization will be explained.

For the sake of simplicity, we consider only a single face consisting of m pixels. The homogeneous 2D coordinates of the pixels are denoted as $\mathbf{p}_i^* \in \{\mathbf{p}_1^*, \mathbf{p}_2^*, \dots, \mathbf{p}_m^*\}$. They are in the coordinate frame of the reference image \mathcal{I}^* . We now define the $(m \times 1)$ error vector $\mathbf{y}(\Delta)$ as row by row concatenation of the per-pixel error measures

$$y_i(\Delta) = \mathcal{I} \left(\mathbf{d} \left(\mathbf{G} \left(\hat{\mathbf{T}}\mathbf{T}(\Delta), \hat{\mathbf{n}}_f^* + \mathbf{n}_f^*(\Delta) \right) \mathbf{p}_i^* \right) \right) - \mathcal{I}^*(\mathbf{p}_i^*) \quad (3.4)$$

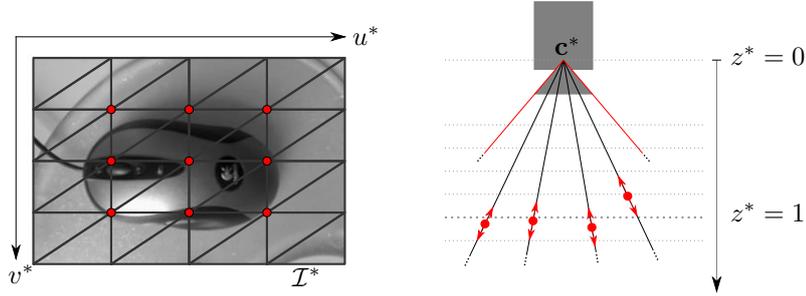


Figure 3.3: Parameterization of the mesh deformation. The vertices of the mesh are free to move along their respective projection ray, *i.e.* their 2D coordinates $[u_i^* \ v_i^* \ 1]^\top$ in the reference image \mathcal{I}^* are fixed but their depth z_i^* may change.

where $\mathbf{d}(\cdot)$ is the perspective division by the last coordinate (*c.f.* equation (2.3)). Intensity values at non-integer coordinates in \mathcal{I} are bilinearly interpolated as described in appendix B.

The update is parametrized as $\Delta = [a_{1\dots 6} \ \psi_{1\dots n}^*]^\top$. The first six parameters a_i represent the update $\mathbf{T}(\cdot)$ of the pose of the camera. We use the Lie algebra of $\mathbb{SE}(3)$ for $\mathbf{T}(\cdot)$ as explained in appendix A. The remaining ψ_i^* of Δ are used to update the mesh, more specifically the normals of each face. The parametrization of the mesh updates and $\mathbf{n}_f^*(\cdot)$ are explained next.

3.3.1 Parameterization of the mesh updates

Deformations of the mesh M^* are modeled by moving vertices along their respective rays emanating from the camera center \mathbf{c}^* in the reference view, as shown in figure 3.3. Every vertex \mathbf{v}_i^* of the mesh is defined via its homogeneous 2D coordinates $\mathbf{v}_i^* = [u_i^* \ v_i^* \ 1]^\top$ in the frame of the reference image \mathcal{I}^* and its depth z_i^* with respect to the camera center \mathbf{c}^* .

In the following we express the normal $\mathbf{n}_f^*(\Delta)$ of a face f in terms of its vertices $\{\mathbf{v}_i^*, \mathbf{v}_j^*, \mathbf{v}_k^*\}$ and associated inverse depths $\{\psi_i^*, \psi_j^*, \psi_k^*\}$. A 3D point $[x \ y \ z]^\top$ is projected into the image as

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{d} \left(\mathbf{K} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \right) = \frac{1}{z} \mathbf{K} \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \quad (3.5)$$

This gives

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = z\mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \quad (3.6)$$

The co-planar 3D points of each face further satisfy the plane equation

$$\mathbf{n}^\top \begin{bmatrix} x \\ y \\ z \end{bmatrix} = d \quad (3.7)$$

with the plane normal $\mathbf{n} = [n_x \ n_y \ n_z]^\top$. Substituting $[x \ y \ z]^\top$ using (3.6) gives

$$\mathbf{n}^\top z\mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = d. \quad (3.8)$$

Multiplying by $\frac{1}{dz}$ (where $z \neq 0$ and $d \neq 0$) gives

$$\frac{\mathbf{n}^\top}{d}\mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{z} = \psi^*. \quad (3.9)$$

Transposing the equation finally ends in

$$\begin{bmatrix} u & v & 1 \end{bmatrix} \mathbf{K}^{-\top} \frac{\mathbf{n}}{d} = \psi^*. \quad (3.10)$$

Combining three points $\mathbf{v}_1^*, \mathbf{v}_2^*, \mathbf{v}_3^*$ with according inverse depths $\psi_1^*, \psi_2^*, \psi_3^*$ into this system, we obtain

$$\begin{bmatrix} \mathbf{v}_1^* & \mathbf{v}_2^* & \mathbf{v}_3^* \end{bmatrix}^\top \mathbf{K}^{-\top} \frac{\mathbf{n}}{d} = \begin{bmatrix} \psi_1^* \\ \psi_2^* \\ \psi_3^* \end{bmatrix}. \quad (3.11)$$

The matrix $[\mathbf{v}_1^* \ \mathbf{v}_2^* \ \mathbf{v}_3^*]^\top$ is invertible as the three points are non-collinear. By multiplying first with the inverse of this matrix and then with \mathbf{K}^\top , we finally obtain $\mathbf{n}_f^*(\Delta)$ as

$$\mathbf{n}_f^*(\Delta) = \frac{\mathbf{n}}{d} = \mathbf{K}^\top [\mathbf{v}_1^* \ \mathbf{v}_2^* \ \mathbf{v}_3^*]^{-\top} [\psi_1^* \ \psi_2^* \ \psi_3^*]^\top. \quad (3.12)$$

3.3.2 Minimizing the cost function

To increase the numerical stability of the minimization, we add a regularization term to the cost function via a function $\mathbf{r}(\Delta) : \mathbb{R}^{6+n} \rightarrow \mathbb{R}^{6+n}$ where n is the number of movable vertices in the mesh, also called control points. The regularization will be presented in section 3.3.3. We write cost function as

$$\phi(\Delta) = \frac{1}{2} \left(\|\mathbf{y}(\Delta)\|_2^2 + \lambda \|\mathbf{r}(\Delta)\|_2^2 \right) \quad (3.13)$$

where the scalar λ is used to balance the squared norms of $\mathbf{y}(\Delta)$ and $\mathbf{r}(\Delta)$. We assume that the updates $\mathbf{T}(\Delta), \mathbf{n}_f^*(\Delta)$ of the estimates $\hat{\mathbf{T}}, \hat{\mathbf{n}}_f^*$ are reasonably small such that a Taylor expansion can be used for minimization. We denote \mathbf{J}_y and \mathbf{J}_r as the Jacobians of the data and regularization terms evaluated at $\Delta = \mathbf{0}$. A first-order Taylor expansion of equation (3.13) gives

$$\phi(\Delta) \approx \tilde{\phi}(\Delta) = \frac{1}{2} \left(\|\mathbf{y}(\mathbf{0}) + \mathbf{J}_y \Delta\|_2^2 + \lambda \|\mathbf{r}(\mathbf{0}) + \mathbf{J}_r \Delta\|_2^2 \right). \quad (3.14)$$

The update Δ is then computed by derivation with respect to Δ and setting the resulting equation to zero as

$$\frac{\partial}{\partial \Delta} \tilde{\phi}(\Delta) = \mathbf{J}_y^\top (\mathbf{y}(\mathbf{0}) + \mathbf{J}_y \Delta) + \lambda \mathbf{J}_r^\top (\mathbf{r}(\mathbf{0}) + \mathbf{J}_r \Delta) \stackrel{!}{=} \mathbf{0}. \quad (3.15)$$

We collect the terms dependant on Δ on one side and obtain

$$(\mathbf{J}_y^\top \mathbf{J}_y + \lambda \mathbf{J}_r^\top \mathbf{J}_r) \Delta = -(\mathbf{J}_y^\top \mathbf{y}(\mathbf{0}) + \lambda \mathbf{J}_r^\top \mathbf{r}(\mathbf{0})). \quad (3.16)$$

This system is solved for Δ using *e.g.* Cholesky decomposition or the pseudo-inverse. The update Δ is applied and another iteration is run until either the norm of the update is below a certain threshold (we chose 10^{-3} in the experiments) or the maximal number of iterations is reached.

The Jacobian \mathbf{J}_y can be written as the product of gradient of the estimated reference image $\frac{\partial}{\partial \mathbf{u}} \widehat{\mathcal{I}}^*(\mathbf{u}) = \mathbf{J}_{\widehat{\mathcal{I}}^*}$, the Jacobian of the projection $\frac{\partial}{\partial \mathbf{a}} \mathbf{d}(\mathbf{a}) = \mathbf{J}_d$ and the Jacobian of the homography $\frac{\partial}{\partial \Delta} \mathbf{G}(\widehat{\mathbf{T}}\mathbf{T}(\Delta), \widehat{\mathbf{n}}_f^* + \mathbf{n}_f^*(\Delta)) \mathbf{p}_i^* = \mathbf{J}_G$ as

$$\mathbf{J}_y = \mathbf{J}_{\widehat{\mathcal{I}}^*} \mathbf{J}_d \mathbf{J}_G. \quad (3.17)$$

In the spirit of [BM06], this first order linearization can be approximated to second-order as

$$\mathbf{J}_y = \frac{1}{2} (\mathbf{J}_{\widehat{\mathcal{I}}^*} + \mathbf{J}_{\mathcal{I}^*}) \mathbf{J}_d \mathbf{J}_G \quad (3.18)$$

by including the gradient $\frac{\partial}{\partial \mathbf{u}} \mathcal{I}^*(\mathbf{u}) = \mathbf{J}_{\mathcal{I}^*}$ of the reference image \mathcal{I}^* . As shown in the evaluation, this in general increases the convergence frequency of the Gauss-Newton optimization with low additional costs. As usually done in template tracking, we try to increase the convergence radius and speed using a coarse-to-fine strategy. Specifically, we run the proposed method on multiple levels of a power-of-two image pyramid.

3.3.3 Regularization

In case the camera is close to the reference camera, the matrix $\mathbf{J}_y^\top \mathbf{J}_y$ becomes increasingly ill-conditioned, *i.e.* small changes in $\mathbf{y}(\mathbf{0})$ may provoke big changes in the estimated update Δ . This is because the projection rays of the current camera are approximately aligned with those of the reference camera (depicted in figure 3.3). In this degenerate configuration, arbitrary movements of the vertices, respectively their inverse depths ψ_i^* , result in almost identical unwarped reference images $\widehat{\mathcal{I}}^*$.

However, this configuration can be easily mitigated by adding a regularization term to the cost function that restrains the vertices in that case. We define $\mathbf{r}(\Delta) = [\mathbf{0}_{(1 \times 6)} \ r_1(\Delta) \ r_2(\Delta) \ \dots \ r_n(\Delta)]^\top$ such that it does not effect the update of the pose but only operates on the n movable vertices. We compute $r_i(\Delta)$ for all $i \in \{1, 2, \dots, n\}$ as

$$r_i(\Delta) = \left(1 + \lambda_s e^{-\lambda_r \|\widehat{\mathbf{t}}\|^2}\right) \left(\frac{1}{\widehat{\psi}_i^* + \psi_i^*} - \bar{z}_i\right). \quad (3.19)$$

The first part of the regularization term is a weighting factor based on the baseline $\widehat{\mathbf{t}}$ of the last iteration that penalizes the degenerate configuration just discussed.

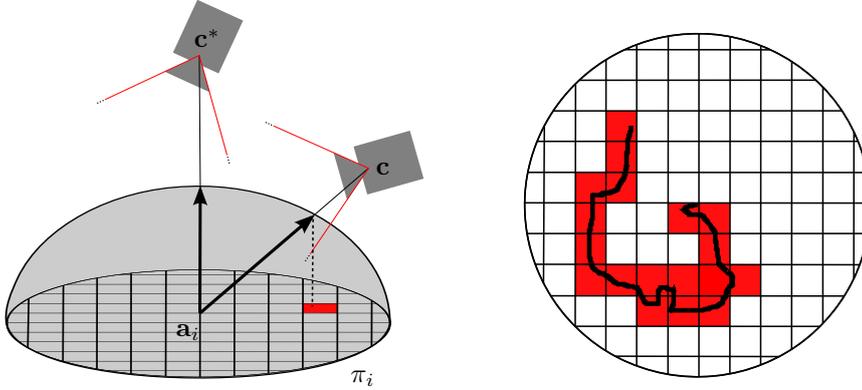


Figure 3.4: Computation of reference depth \bar{z}_i from all successful previous registrations. Left: The highlighted cell is currently used for storing the estimate of the height and the similarity measure based on adjacent faces of the vertex. Right: Example of cells containing data during typical run of the algorithm, with the ground track of the camera overlaid.

The scalars λ_s and λ_r determine the scale and range of the penalty concerning the baseline, empirically $\lambda_s = \lambda_r = 10$ gave good results. The second part of $r(\Delta)$ is responsible for damping the deformations and moving them towards their most likely true value. It penalizes changes of the depths with respect to a reference depth \bar{z}_i of the vertex.

A naïve way of determining \bar{z}_i may consist in computing it as running average, *e.g.* updated after every image as $\bar{z}_i \leftarrow 0.9\bar{z}_i + 0.1/\hat{\psi}_i^*$. This method is simple yet effective in case of a continuously moving camera. However, when the camera becomes stationary, \bar{z}_i will converge towards the value optimal for only this local configuration (which may be different from the globally optimal depth because of ambiguities). In other words, all information from previous successful registrations will be lost over time.

Therefore, our approach of computing \bar{z}_i tries to preserve previous knowledge about the camera motion. The idea is to spatially sample height estimates together with confidence values incorporating not only the height estimate of vertex i but also other estimates such as the pose of the camera and the height estimates of adjacent vertices. It is outlined in figure 3.4 and will now be explained in detail.

We use the ray to the reference camera center \mathbf{c}^* from the initially assumed 3D point \mathbf{a}_i corresponding to vertex \mathbf{v}_i as normal for a plane π_i . On this plane, a unit square centered on \mathbf{a}_i is subdivided into a regular grid. While the user moves the camera, we cast a ray from \mathbf{a}_i to the current camera center \mathbf{c} , intersect it with the unit half-sphere around \mathbf{a}_i and project the intersection onto π_i , as displayed on the left side of figure 3.4. The cell which contains the projection records the current height estimate of the algorithm and a similarity measure S_i consisting of the sum of the normalized cross correlations (NCC) between \mathcal{I}^* and $\hat{\mathcal{I}}^*$ for all adjacent faces of vertex i . The NCC is defined for two $(U \times V)$ images \mathcal{I}_a and \mathcal{I}_b as

$$\text{NCC}(\mathcal{I}_a, \mathcal{I}_b) = \frac{1}{N-1} \frac{1}{\sigma_a \sigma_b} \sum_{\mathbf{u} \in \mathbf{U}} (\mathcal{I}_a(\mathbf{u}) - \mu_a) (\mathcal{I}_b(\mathbf{u}) - \mu_b) \quad (3.20)$$

with $N = UV$ and where \mathbf{U} is the set of the integer pixel coordinates with measured intensities (*c.f.* equation (2.12)). Furthermore, the average intensity μ_i is defined as

$$\mu_i = \frac{1}{N} \sum_{\mathbf{u} \in \mathbf{U}} \mathcal{I}_i(\mathbf{u}) \quad (3.21)$$

and the deviation σ_i is defined as

$$\sigma_i = \sqrt{\frac{1}{N} \sum_{\mathbf{u} \in \mathbf{U}} (\mathcal{I}_i(\mathbf{u}) - \mu_i)^2}. \quad (3.22)$$

The similarity measure and the height estimates stored in a cell are updated when there was either no prior record or in case the stored similarity measure was smaller than the one currently obtained. The reference height \bar{z}_i is then computed from the recorded data as

$$\bar{z}_i = \frac{\sum_{\mathbf{p}} z_i(\mathbf{p}) S_i(\mathbf{p}) w(\mathbf{p}, \mathbf{a}_i)}{\sum_{\mathbf{p}} S_i(\mathbf{p}) w(\mathbf{p}, \mathbf{a}_i)} \quad (3.23)$$

where \mathbf{p} represents a point inside the unit square on plane π_i , $z_i(\mathbf{p})$ is the corresponding recorded estimated heights, $S_i(\mathbf{p})$ the according summed adjacent NCCs and $w(\mathbf{p}, \mathbf{a}_i)$ is a function that down-weights samples drawn close to \mathbf{a}_i as these are least discriminative. We used as weighting function

$$w(\mathbf{p}, \mathbf{a}_i) = 1 - e^{-\lambda_w \|\mathbf{p} - \mathbf{a}_i\|_2^2} \quad (3.24)$$

with $\lambda_w = 50$, on a grid of resolution (100×100) . Initially, the value of \bar{z}_i changes rapidly as the shape transforms from the initial estimate towards a more likely shape. The variance of \bar{z}_i increases, but at a certain point, when the user moved sufficiently, the variance begins to steadily decrease. The objective is that when the user has seen the object from various viewpoints so that all cells of the unit circle are covered, \bar{z}_i becomes constant. Consequently, the estimated shape of the template becomes very close to the running average and regularization practically cancels. In practice, the outer regions of the grid are in general rarely visited, so we used the grid only for the inner 70 % of the unit circle and stored the data of the outer 30 % in its outmost cells to better utilize the grid.

3.4 Evaluation

The proposed method was quantitatively evaluated both on synthetic and real video sequences for which ground truth of the camera pose was available. In case of the synthetic sequence we also evaluated the estimate of the shape. Further, we evaluated the method qualitatively on smooth objects and on objects with creases, observed from a moving camera. We also tested our method on a smoothly deforming object with a fixed camera. We conducted a comparison against PTAM [KM07] and analyzed both methods in presence of several levels of blur. Videos of the evaluations can be found in the supplementary material.

3.4.1 Quantitative evaluation

Synthetic evaluation We created a synthetic pyramid first seen from the top, then moving towards the lower left corner of the image while rotating. We used a mesh of 16 faces and 13 vertices from which only the central vertex was fixed at $z = 1$. No regularization was employed as neither noise nor degenerate configurations are present and only a maximum of five iterations per frame on pyramid level 0, *i.e.* on the original image resolution, were allowed.

The method shows low errors in both pose and shape of the object. The synthetic evaluation is illustrated in figure 3.5 and in the supplementary material.

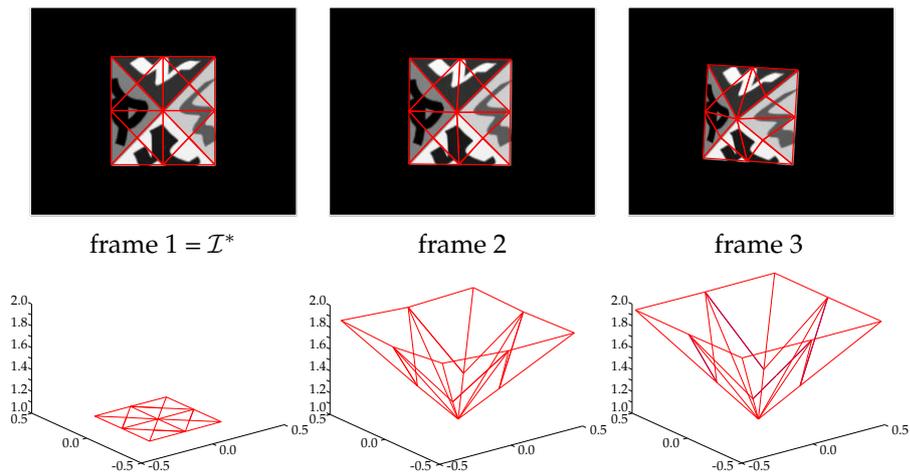


Figure 3.5: Evaluation of the shape on synthetic data. Upper row shows input frames with an overlay of the 2D projection of the recovered shape shown below.

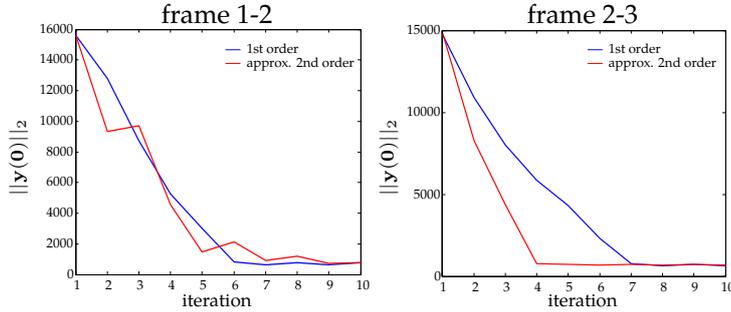


Figure 3.6: The proposed second-order approximation of \mathbf{J}_y converges 2–4 iterations earlier in case of slight mesh deformations, using the synthetic sequence from figure 3.5.

As can be seen, despite the small movement of the pyramid from frame 1 to frame 2, the recovered shape quickly adjusted towards the true shape of the pyramid. When comparing the first order linearization of \mathbf{J}_y with the presented approximated second-order linearization, we observed that they have similar convergence rate when there is strong motion in the depths like in frames 1-2 as shown in figure 3.6. However, when the estimation of the structure is changing just slightly like in frames 2–3 of this sequence, 2 to 4 iterations may be saved and our results match those of Benhimane and Malis [BM06] in terms of convergence.

Real sequence To perform a quantitative evaluation with real camera images, we have created a sequence with ground truth poses of a real camera as presented in section 2.6.2. The sequence starts from a near fronto-planar view onto a low-textured computer mouse and features a near-circular motion. It was first used to evaluate our method, ESM [BM07] and a calibrated multi-planar tracking method [BM06] referred to as DP. The algorithms were given identical parameters, *i.e.* 2 pyramid levels and (at most) 5 iterations per level. Poses were computed from the 2D–3D correspondences of the corners of the tracked template image using POSIT [DD95]. Images from the sequence and the errors of the poses of the algorithms are shown in figure 3.7. The rotational error in radians was on average 0.30 (17.2°) for DP, 0.16 (9.2°) for ESM and 0.07 (4.0°) for the proposed method; translational errors in the sequences averaged to 15.12 mm, 9.11 mm and 4.44 mm respectively. The proposed method clearly outperforms method that assume global planarity.

Furthermore, we evaluated the robustness of the proposed method with respect to blur introduced by consecutively applying a (5×5) mean filter. This kind of blur can be found in real data when the object is out-of-focus given

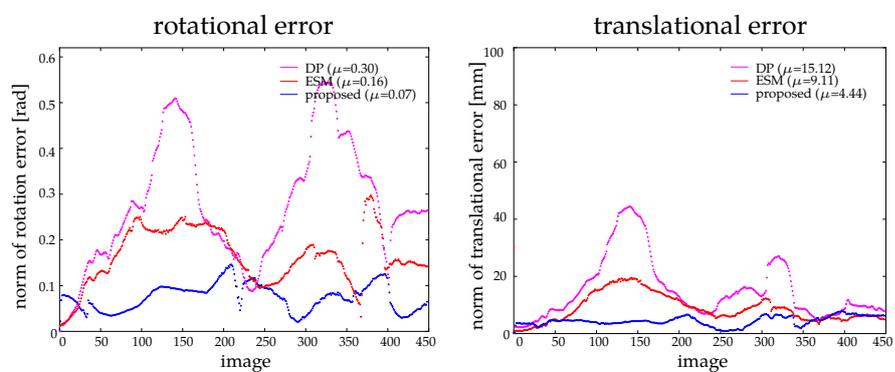
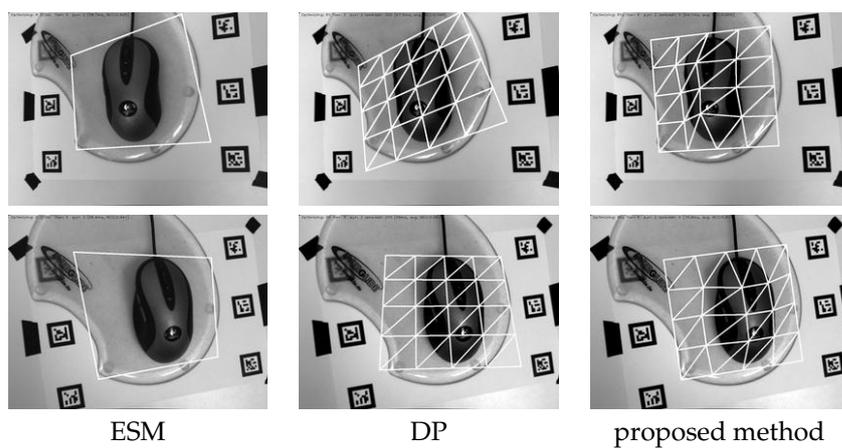


Figure 3.7: Evaluation on real data. Comparison of ESM [BM07], DP [BM06] and proposed method.

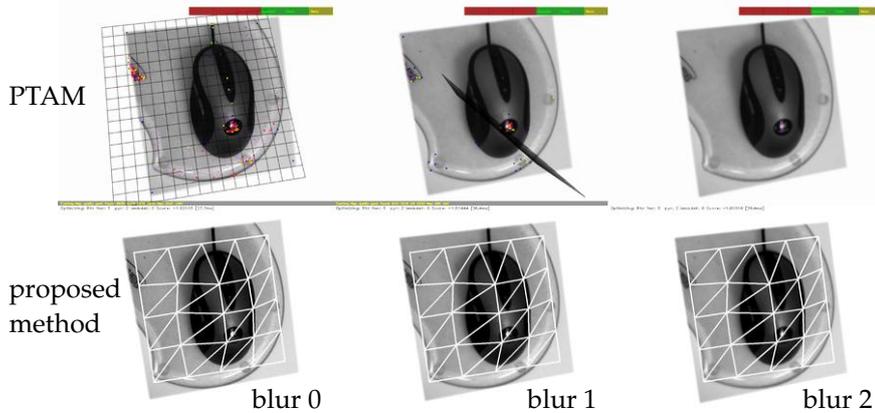


Figure 3.8: Evaluation against blur originating from a (5×5) mean filter which was applied consecutively 0–4 times. PTAM draws a ground plane into the image if it tracks while we project the deformed 3D mesh model.

e.g. a fixed-focus camera. We observed that the accuracy of the method did increase slightly as the blur increased, which could be explained by the way we construct the pyramid levels in the implementation. For speed reasons, we do not subsample the whole image, but only bilinearly interpolate at the pixel positions we have to warp. Thus, only the intensity of the four nearest neighbors of a pixel position is used. On a high pyramid level, the spacing between warped pixels may become so large that some pixels of the object in the camera image may not contribute to the computation of the photometric error. As the blur diffuses the information, the higher pyramid levels effectively use more than the intensities of four pixels per warped template pixel.

The same sequences were given to PTAM. As poses of PTAM are defined in a rather arbitrary coordinate system, we aligned them by minimizing the sum-of-squared distance to the ground truth, solving for a 6-DOF transformation and 1-DOF scale (see section 2.7.2). PTAM recovered the trajectory of the camera extremely well – however, as PTAM uses FAST [RD05] as feature detector, the vast majority of the features used were on the high-contrast fiducials. As we were rather more interested in evaluating the performance when tracking the computer mouse, we removed the markers from the images in the same way as presented in section 2.6.1. The proposed method and PTAM were evaluated on this sequence. PTAM could not successfully initialize starting from the second level of blur, as depicted in figure 3.8, presumably since there were very few features on the lowest image pyramid level to be tracked. Table 3.1 shows the average translational error and its standard deviation for both methods on the

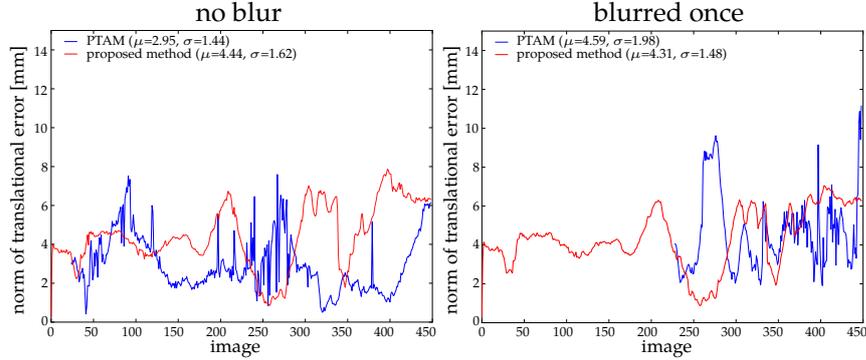


Figure 3.9: Translational errors of PTAM and the proposed method for cropped images and blur levels 0 and 1.

	blur 0	blur 1	blur 2	blur 3	blur 4
proposed method, full image	4.44, 1.62	4.32, 1.49	4.17, 1.43	4.04, 1.39	3.39, 1.38
PTAM, full image	2.49, 0.70	1.70, 0.58	2.00, 0.73	1.70, 0.60	1.61, 0.62
proposed method, cropped image	4.44, 1.62	4.31, 1.48	4.17, 1.43	4.03, 1.39	3.39, 1.38
PTAM, cropped image	2.95, 1.44	4.59, 1.98	-	-	-

Table 3.1: Average translational error and standard deviation of the pose for both cropped and original image sequence (in [mm]). While PTAM is clearly superior on the full image due to also tracking the fiducials, the proposed method also obtains almost identical results when these are removed as they are not in the template image.

full and cropped images. As can be seen, the proposed method is giving the same results both for full and cropped image.

3.4.2 Qualitative evaluation

To analyze how the method works in case of a smooth object and in case of object with creases, we evaluated it by tracking a cup and a truncated pyramid. The method was able to track both objects well and approximated the shapes reasonably. As observed also by Datta *et al.* [DSK08], best results are obtained when the structure of the mesh is able to express the structure of the underlying object. Furthermore, we evaluated the robustness of the method when tracking deformed objects. Although this violates the rigidity assumption, the method copes well with slight deformations as shown in figure 3.10. In the cup sequence, after estimating the shape we manually disabled the estimation of the depths and used the method only for tracking the pose. We show that the pose is well estimated even under severe occlusion of up to 50% of the mesh. On a 2.5 GHz dual core notebook, the speed is typically 10–30 ms per frame when

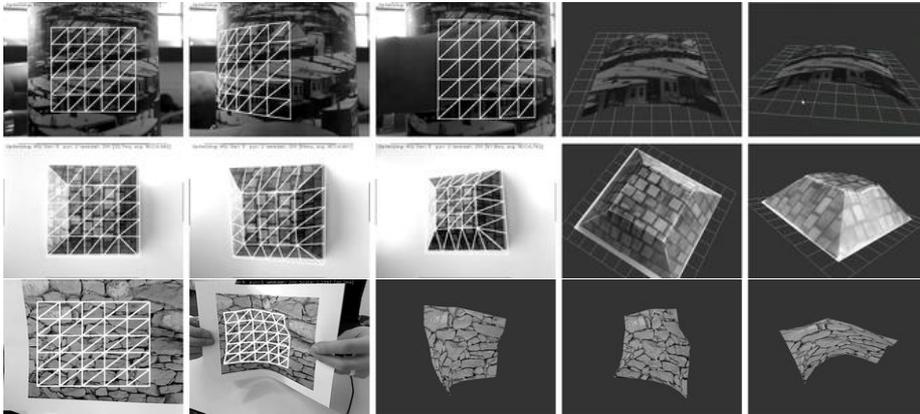


Figure 3.10: Qualitative evaluation of recovered shapes. The first frame of each sequence (shown left-most) is used as template. *Upper and middle row*: Shape recovery of the rigid objects from moving camera where also the camera motion is estimated. *Lower row*: Recovering shape of the deforming object where the camera is not moving. Although the method was not designed for such situations, we still managed to apply it to recover moderate object deformations.

estimating the camera pose and around 40–60 ms when additionally estimating the deformations. The timings were obtained using pyramid levels 3 and 2, at most 5 iterations per level and a mesh of approximately (200×200) pixels on level 0. Most of the time was spent in the direct computation of $\mathbf{J}_y^T \mathbf{J}_y$.

RGB-D BASED TRACKING AND MESHING

This chapter describes a method for tracking the pose of a moving RGB-D camera and in parallel computing a meshed version of the environment.

4.1 Motivation

For several vision-based AR applications that relied on end-user hardware, determining the relative motion of a single camera with respect to an unknown environment was made possible thanks to approaches inspired from Davison's MonoSLAM [Dav03]. This approach and its successors are performing real-time tracking of visual features extracted from live camera images. The features need to be seen in many images for which the camera has performed a motion that is sufficiently large for estimating the depth and consequently reconstructing the 3D coordinates of the features.

Generally, the reconstruction is based on the structure-from-motion principle. As briefly stated in section 2.5, to get correctly scaled 3D coordinates of the reconstructed points and therefore an *e.g.* metric camera motion, these approaches usually require an explicit manual measurement of some parts of the environment. Alternatively the environment could be equipped with known objects. Yet another possibility to induce scale is to ask the user to perform a constrained camera motion. For example, this done in the initialization of Klein and Murray's PTAM [KM07] where the camera needs to move between two user-selected frames such that its optical center position varies with a metrically



Figure 4.1: RGB-D camera Microsoft Kinect. *From left:* IR projector, RGB camera and IR camera. © Microsoft

known scaled translation. If one of these scale-inducing conditions is fulfilled, it is possible to estimate a correctly scaled relative motion of the camera with respect to an unknown environment and thus reconstruct a correctly scaled (sparse) representation of it. The reconstructed 3D point clouds are generally based on visual features as the only sensor used is a camera.

We see here some limitations of the existing approaches. First, before reconstructing a point and adding it to the map, the point needs to be tracked over multiple frames that have an estimated camera pose. This delays the participation of a newly visible physical point in the estimation of the *full* camera motion (it is possible that non-triangulated points participate in the estimation of the rotation of the camera as done by Civera *et al.* [CDM08]). Second, either the environment needs to be partially measured or pre-equipped or the user needs to have some experience with the system in order to correctly perform a constrained camera motion that allows correct scale estimation. Third, since the existing approaches are mainly based on visual features (often extracted where some texture gradient is available), the online map obtained from the existing approaches is generally sparse and could not be used, even after post-processing and meshing, for occlusion handling or similar AR tasks that require a meshed version of the environment.

Camera systems that are designed to additionally provide a correctly scaled depth of an imaged pixel would solve the above problems. However, for several years, typical depth camera systems had low resolution, noisy measurements, restricted working area and/or high cost. Whether they are based on Time-Of-Flight technology or on standard Digital Fringe Projection, these camera systems did not have the huge impact that the advent of the Microsoft Kinect (shown in figure 4.1) has had starting from its launch in November 2010. In fact, this end-user low cost and relatively high resolution RGB-D camera is based on a RGB camera registered to a stereo system composed of an infra-red structured

light projector combined with an infra-red camera which allows for pixel depth computation. Originally targeted at indoor use and intended as a supplement for gaming devices (XBox 360), the Microsoft Kinect got a high interest from the research community once PrimeSense, the company that provided its reference design, released official drivers for this device.

In this chapter, we investigate how such a device allows an important step forward in the field of computer vision in general and in AR in particular. We propose a real-time method based on such a consumer RGB-D camera that estimates metric camera motion with respect to an unknown environment and that builds a dense meshed and textured version of the surrounding environment at the same time.

The remainder of this chapter is structured as follows. We first position our contribution with respect to the existing and related state-of-the-art. Then, we describe the approach used for the real-time RGB-D camera motion estimation and meshing of the surrounding environment. The estimated trajectory of the camera using the proposed method is evaluated against PTAM on ground truth data as described in chapter 2. Furthermore, the recovered shape of a challenging industrial object is compared to its known CAD model. Finally, we illustrate the usage of the proposed method on different AR scenarios.

4.2 Related work

Visual real-time tracking with respect to known or unknown scenes is an essential component of vision-based AR applications. There were numerous algorithmic contributions in the topics in the last few years. But, if Davison *et al.*'s seminal MonoSLAM [DMM03, Dav03] showed that it is possible to perform SLAM using a single camera on end-user hardware in real-time, Klein and Murray's PTAM [KM07] showed that adapting and updating the algorithms used for estimating the camera motion according to end-user available computational capabilities allows to get impressive tracking results in small AR workspaces.

In fact, estimating the camera motion by tracking the environment and in parallel building a feature-based sparse map was made possible thanks to the generalization of multi-core processors on desktop computers and laptops. As MonoSLAM was published a few years earlier, it relied on a single-threaded paradigm to handle few (supposedly) high-quality features and adjust their state with new measurements from every camera frame. This has the benefit that every available information about a feature is integrated into its state. However, it comes at the cost of being able to handle only maps of up to 100 features (mostly due to the real-time constraint). More importantly, this approach has a low level of robustness as errors in the data association, *e.g.* due to fast camera motion, permanently corrupt the map of this SLAM system.

On the other hand, when multiple threads can be executed at the same time, a clear separation between tracking and mapping has the benefit that the mapping component is free to employ methods that traditionally were intended for offline usage. The map refinement is run on a separate CPU core and can make use of *e.g.* data-reassociations and (robust) bundle adjustment [TMHF00]. The mapping scales with the number of *mapped* features and keyframes, *i.e.* camera frames selected based on the spatial, photometric or temporal distance. The tracking is able to maintain real-time performance as it scales with the number of *tracked* features in the current camera frame. A decoupled system is typically able to handle much bigger maps (few 10.000 features) while still tracking at frame-rate.

On a side note, Strasdat *et al.* analyzed these predominant paradigms in monocular SLAM, *i.e.* *filtering* and *optimization*, in a series of real and synthetic experiments. They concluded that, "while filtering may have a niche in systems with low processing resources, in most modern applications keyframe optimization gives the most accuracy per unit of computing time" [SMD10].

Many extensions of the above approaches showed that it is possible to scale monocular SLAM approaches to a larger environment, such as the work of Eade and Drummond [ED08] or Castle *et al.* [CKM08] who employ multiple local maps. Notably, the former system also automatically merges these local maps as Eade and Drummond interpreted recovery as loop closing. As already presented in chapter 3, Newcombe and Davison [ND10] recently showed that by leveraging the incomparably higher computational power of a modern GPU, it is possible to get a dense representation of a desktop scale environment and highly textured scene using a single standard hand-held video camera. While the basis tracking system used in [ND10] was PTAM, Newcombe *et al.* [NLD11] went one step further and also use the online created dense textured surface model for tracking the camera's pose. As the pose estimation is achieved by a direct alignment method, their method is also robust to blur caused by out-of-focus or motion (*c.f.* section 3.4).

Also, recently, Castaneda *et al.* [CMN11] replaced the generally used standard hand-held video camera with a combination of a time-of-flight (204×204) resolution camera and a (640×480) RGB camera and modified the measurement model and the innovation formulas of the Extended Kalman filter used by MonoSLAM to improve the tracking results. This work did not use a powerful PC but instead a typical expensive depth camera system.

Henry *et al.* [HKH⁺10] directly used an RGB-D camera from PrimeSense, similar to the Microsoft Kinect, for surfel-based modeling of indoor environments. While their system is capable of creating models of rather large indoor scenes, it does not run on live data in real-time uses recorded videos as input. Furthermore, it does not perform any real-time inter-frame tracking crucial for AR. In parallel to the publication of the work presented in this chapter, Newcombe *et al.* [NIH⁺11] presented a system coined KinectFusion that exclusively relies on depth maps obtained from the Kinect. It utilizes a modern GPU to create a high-quality reconstruction by aligning and integrating every pixel of the captured depth maps into a global implicit surface model.

4.3 Real-time camera motion estimation

A typical SLAM system consists of three major parts: Map building, inter-frame tracking and relocalization. The system proposed in this chapter uses a map for tracking that consists of RGB-D keyframes with associated camera poses and sparse 3D points with descriptors computed from the keyframes. An example of such a map can be seen in figure 4.2. In the following, we will describe the three stages and how they take the depth image into account.

4.3.1 Inter-frame tracking

Assuming that we have an initial map, an estimate ${}^{camera}\mathbf{T}_{world,i-1}$ of the camera pose and 2D–3D correspondences \mathcal{C}_{i-1} from the last camera frame \mathcal{I}_{i-1} , the task of the inter-frame tracking is to estimate the current camera pose ${}^{camera}\mathbf{T}_{world,i}$ given the current frame \mathcal{I}_i . In our system this is realized by first updating the 2D part of the correspondences \mathcal{C}_{i-1} to obtain \mathcal{C}_i . Next, the updated correspondences and the camera pose ${}^{camera}\mathbf{T}_{world,i-1}$ are used to compute the current camera pose ${}^{camera}\mathbf{T}_{world,i}$. These two steps will be explained in the following. As in the previous chapters and appendix A, we define ${}^a\mathbf{T}_b$ as homogeneous (4×4) matrix comprising the rotation and translation necessary to transfer a point from coordinate frame b to a .

The correspondences \mathcal{C}_{i-1} can be written as a set of tuples $(\mathbf{u}_j, \mathbf{x}_j)$ where we define \mathbf{u}_j as homogeneous 2D coordinates and \mathbf{x}_j as corresponding homogeneous 3D coordinates. The coordinates are normalized, *i.e.* their homogeneous component is equal to 1. The 2D part of the correspondences is updated via sparse optical flow, *i.e.* by independently minimizing the photometric error between \mathcal{I}_{i-1} and \mathcal{I}_i within $(N \times N)$ windows around each feature position \mathbf{u}_j . In contrast to other SLAM systems that rely on 3D motion models, we employ a 2D motion model independently for each feature. In particular, we use a pyramidal implementation of the sparse optical flow algorithm by Lucas and Kanade [LK81, Bou99]. The updated 2D–3D correspondences \mathcal{C}_i are used to obtain ${}^{camera}\mathbf{T}_{world,i}$ by iteratively minimizing the cost function

$$\phi_i(\Delta) = \sum_{j=1}^{|\mathcal{C}_i|} \mathbf{w}(y_j(\Delta)) \quad (4.1)$$

with the weighting function $w(\cdot)$ defined in equation (4.3) and the error measure $y_j(\cdot)$ defined as

$$\mathbf{y}_j(\Delta) = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \left(\mathbf{u}_j - \mathbf{d} \left(\begin{bmatrix} \mathbf{K} & \mathbf{0} \end{bmatrix} \mathbf{T}(\Delta) \hat{\mathbf{T}} \mathbf{x}_j \right) \right) \quad (4.2)$$

where \mathbf{I} is a (2×2) identity matrix, $(\mathbf{u}_j, \mathbf{x}_j) \in \mathcal{C}_i$, \mathbf{K} are the known (3×3) intrinsic camera parameters and $\mathbf{0}$ are zero columns of appropriate size. The perspective division $\mathbf{d}(\cdot)$ is defined in equation (2.3). As can be seen in equation (4.2), we subsequently remove the homogeneous entry of the 2D and 3D vectors by multiplying with zero columns to obtain an inhomogeneous error measure.

The estimate of the current camera pose is denoted as $\hat{\mathbf{T}}$, it is initially set to ${}^{camera} \mathbf{T}_{world, i-1}$ and updated after each iteration as $\hat{\mathbf{T}} \leftarrow \mathbf{T}(\Delta) \hat{\mathbf{T}}$. The parametrization of the pose update $\mathbf{T}(\Delta)$ is based on the Lie algebra $\mathfrak{se}(3)$ and described in appendix A. To limit the influence of outliers, we weight the reprojection error with the Tukey M-Estimator [Ste99] in each iteration:

$$\mathbf{w} \left(\begin{bmatrix} u \\ v \end{bmatrix} \right) = \begin{bmatrix} \rho(u) \\ \rho(v) \end{bmatrix} \quad (4.3)$$

with $\rho(\cdot)$ defined as

$$\rho(x) = \begin{cases} c^2/6 - \left(1 - \left(1 - (x/c)^2 \right)^3 \right) & \text{if } |x| \leq c \\ c^2/6 & \text{if } |x| > c \end{cases} \quad (4.4)$$

with the tuning constant $c = 4.6851 \sigma(\mathbf{y}(\mathbf{0}))$ for a 95% asymptotic efficiency of the estimator. The standard deviation $\sigma(\mathbf{y}(\mathbf{0}))$ of the concatenated residual vector $\mathbf{y}(\mathbf{0}) = [\mathbf{y}_1(\mathbf{0})^\top \mathbf{y}_2(\mathbf{0})^\top \dots \mathbf{y}_{|\mathcal{C}_i|}(\mathbf{0})^\top]^\top$ is robustly estimated via the median absolute deviation (mad) [Ste99] as

$$\sigma(\mathbf{y}) = k \text{ mad}(\mathbf{y}(\mathbf{0})) \quad (4.5)$$

$$= k \text{ median}(|\mathbf{y}(\mathbf{0}) - \text{median}(\mathbf{y}(\mathbf{0})) \mathbf{1}|) \quad (4.6)$$

where the scalar $k = 1/\Phi^{-1}(3/4) \approx 1.4826$ is used to neutralize the mad of the normal distribution $\Phi(\cdot)$ and $\mathbf{1}$ is a vector consisting of $|\mathcal{C}_i|$ one elements. Equation (4.1) is minimized by linearizing $\mathbf{T}(\Delta)$ around $\Delta = \mathbf{0}$ and solving the resulting normal equations as described in section 3.3.2. At each iteration, the update Δ is computed as solution for the system

$$\mathbf{J}^\top \mathbf{W} \mathbf{J} \Delta = \mathbf{J}^\top \mathbf{W} \mathbf{y}(\mathbf{0}) \quad (4.7)$$

where \mathbf{J} is the Jacobian matrix $\frac{\partial}{\partial \Delta} \mathbf{y}(\Delta)$ evaluated at the origin $\Delta = \mathbf{0}$ and \mathbf{W} is a diagonal matrix containing the weights $w(y_i)$ for each component y_i of $\mathbf{y}(\mathbf{0})$. The weight matrix \mathbf{W} is updated before each iteration based on the latest residual $\mathbf{y}(\mathbf{0})$. Each weight is computed as

$$w(x) = \begin{cases} \left(1 - (x/c)^2\right)^2 & \text{if } |x| \leq c \\ 0 & \text{if } |x| > c \end{cases}. \quad (4.8)$$

Features which were classified as outliers by the M-Estimator (*i.e.* final weight equal to zero) and features with reprojection error bigger than a fixed threshold are removed from \mathcal{C}_i and thus discarded from further tracking.

Using the optical flow from the last to the current image is robust against lighting changes and sudden motions, however it is prone to drift as small frame-to-frame inaccuracies typically accumulate in the long run. To mitigate drift, we additionally determine the closest keyframe \mathcal{K}_j in terms of rotation and translation of its associated pose ${}^{world}\mathbf{T}_{keyframe,j}$ and reproject the 3D points of this keyframe (highlighted in red in figure 4.2) into the current image \mathcal{I}_i using the current camera pose ${}^{camera}\mathbf{T}_{world,i}$. The generated correspondences are added to \mathcal{C}_i and will be used for estimating the pose ${}^{camera}\mathbf{T}_{world,i+1}$ in the next camera frame \mathcal{I}_{i+1} . The reprojection has the benefit that, assuming a high number of features in the keyframes, we always can use the maximum number of features for tracking. This strategy which greatly improved the accuracy and robustness of the proposed system overall.

As the pose ${}^{camera}\mathbf{T}_{world,i}$ used for reprojection may already incorporate some drift, we refine the positions of the reprojected features individually before using them by employing Lucas-Kanade tracking from the keyframe \mathcal{K}_j to the current frame \mathcal{I}_i . The neighborhood of the updated position is checked for photo-consistency using the sum-of-squared differences of a window around the 2D position of the features. However, we assume that the tracker did not drift arbitrarily and thus require the displacement of the 2D feature to be less than a given threshold. This thresholding is unfortunately necessary because we currently use the same 2D feature detection both for relocalization and tracking; the scale-invariant features used for relocalization are based on the principles of SIFT [Low04], however we do not yet use the recovered scale of a feature in the frame-to-frame tracking. Thus it can happen that a feature that is detected on a high pyramid level starts to drift on the lower levels of the pyramid, mostly due to near-uniform appearance on these levels.

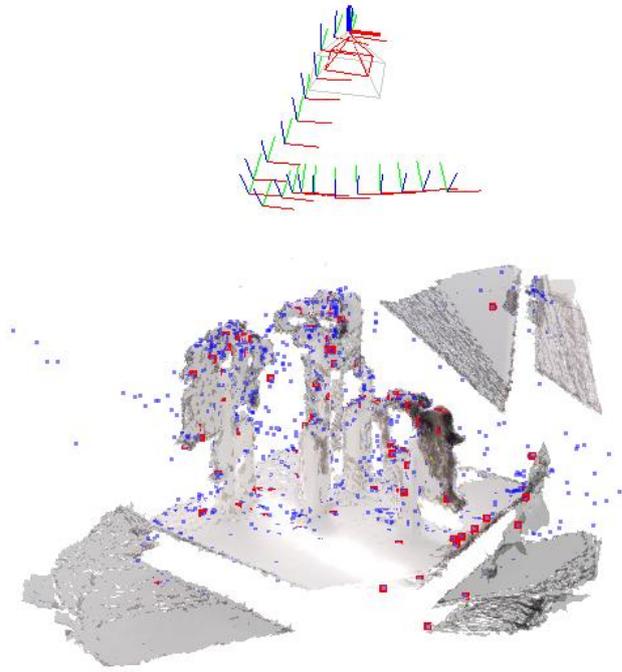


Figure 4.2: Sparse feature point cloud created online during the real-time tracking of a challenging industrial scene. Also shown: the dense reconstruction presented in section 4.4.

It turned out that the pyramidal Lucas-Kanade tracking is the main computational task of the tracking component. Due to real-time constraints, we limit the number of tracked features to around 300. All features discarded after the pose estimation are eventually replaced by reprojected features from the closest keyframe in order to maintain a high number of tracked features.

4.3.2 Sparse feature mapping

Each keyframe of the map is taken from the live image stream and has to fulfill two conditions: firstly, it has to provide a sufficient amount of features (at least 40 in our experiments) that also have an associated measured depth, and secondly, the pose ${}^{world}\mathbf{T}_{keyframe} = {}^{camera}\mathbf{T}_{world}^{-1}$ assigned to the keyframe has to be farther away from all keyframes' poses currently forming the map (see figure 4.2). Initially, the map is empty. We use the first camera image that has enough reconstructed features as first keyframe. The camera pose associated to this keyframe

serves as origin of the global coordinate system. 3 D points are created from the 2 D image coordinates $[u \ v \ 1]^T$ of each feature using

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = z\mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad (4.9)$$

where \mathbf{K} are the known intrinsics of the camera and z is the depth provided by the depth image. Next, the inter-frame tracking is initialized from these 2 D–3 D correspondences and tries to estimate the camera pose for every new camera image as described in the previous section. When a new camera image eventually qualifies as new keyframe based on the currently estimated camera pose, we put the RGB-D image into a queue for further processing on a second thread to not block the tracking on the main thread. Whenever the second thread is ready, the candidate is popped from the queue and the initial conditions on the distance to the poses associated to the existing keyframes is re-checked as closer keyframes may have been added to the map in the meantime. Then, all features are extracted and their descriptors are matched against the descriptors associated to the 3 D points of the map.

We re-estimate the pose based on these 2 D–3 D matches in the same way as described in section 4.3.3 to reduce the potential drift of the inter-frame tracking. If the estimation is successful, the keyframe \mathcal{K} is accepted and added to the map. This consists of adding the RGB-D image with associated camera pose to the known keyframes and adding the unmatched 3 D points with descriptors to the set of mapped 3 D points. Furthermore, we keep the associations of the 2 D–3 D matches used to recompute the new keyframe’s pose in order to be able to refine the 3 D position of mapped features at a later point in time, *e.g.* by Kalman Filtering. Currently we do not filter the depth maps from the Kinect but instead rely on the M-Estimator of the pose estimation, as this approach empirically already seemed to work reasonably well when using a few hundreds of correspondences.

4.3.3 Relocalization

In case the tracking is lost, we try to re-localize the camera by extracting features from the current camera image and then matching their descriptors against those of the mapped 3 D points. We currently use an exhaustive search in the descriptor space to obtain a first and second nearest neighbors and use their ratio

to determine match/no-match as in [Low04]. The matches are filtered using a RANSAC [FB81]-scheme to remove outliers and obtain an initial guess of the camera pose for the robust non-linear pose estimation presented in section 4.3.1. The initial pose is estimated from the correspondences using POSIT [DD95]. If the pose estimation was successful, we also project features from the closest keyframe the current camera image as in section 4.3.1 to potentially increase the number of tracked features. For the next camera image, the inter-frame tracking is thus re-initialized with the pose estimate and a set of 2D-3D correspondences from map and closest keyframe.

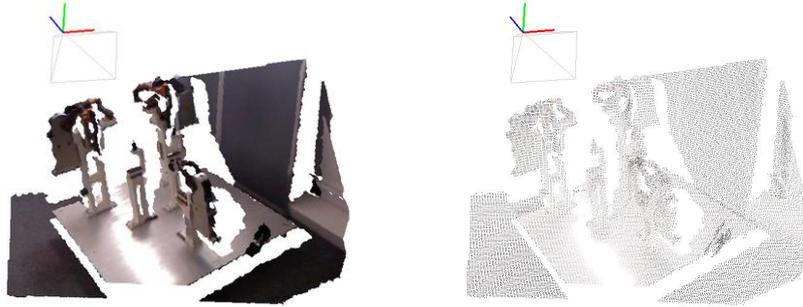


Figure 4.3: *From left:* Mesh created from single RGB-D image and underlying 3D point cloud.

4.4 Online environment mesh creation

In this section, we explain the approach we use to reconstruct the environment as a textured dense mesh. The reconstruction process is done in parallel to the tracking and consists of the following tasks: the selection of the RGB-D frames for updating the meshed model, the meshing of the point cloud corresponding to the selected RGB-D frame and finally the integration of the newly created local mesh into the global model. These steps will be explained in the following, starting with the meshing. In general, we favored integration speed of new data over global accuracy as the system should be used online with no special hardware besides the RGB-D sensor.

4.4.1 Creating local textured meshes

Similarly to the sparse feature mapping, the dense meshing is also conducted on a separate thread that gets RGB-D image and poses from the main tracking thread. The first RGB-D frame that arrives is meshed and forms the basis into which all other meshes will be integrated. We mesh all pixels on a regular grid that also have an associated depth. The stepsize of the grid was set to five for the online experiments as acceptable trade-off between speed and quality.

We eventually filter the sampled pixels such that we only keep points that are not farther than a certain distance (2 m in most experiments) from the depth sensor. This helps to improve the quality of the meshed point cloud in case the uncertainty of the depth measurement increases significantly with the depth, as it is the case with the characteristic quantization effects of the Microsoft Kinect.

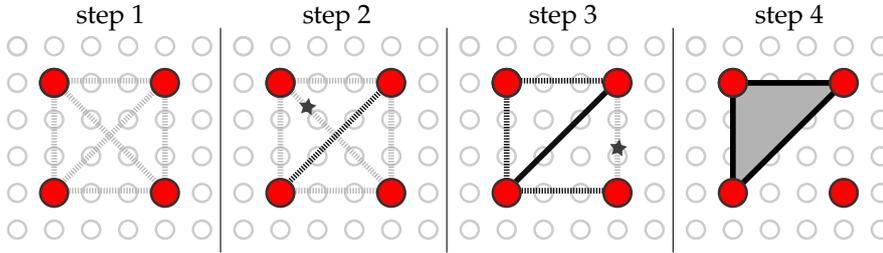


Figure 4.4: Meshing approach adopted from Turk and Levoy [TL94].

The remaining samples are reconstructed in 3 D using equation (4.9). The samples are meshed using part of the “polygon zippering” method presented by Turk and Levoy [TL94]. A single depth map is converted into a mesh by moving a window over the depth map. The four corners of this window, illustrated as red points in figure 4.4, are linked with edge candidates. First, the smaller of the two possible diagonals is selected in order to better preserve details. Then the remaining edge candidates are analyzed. They become part of the mesh if the distance between the reconstructed 3 D points is below a threshold. Up to two triangles per window are created. Finally, the window is shifted to the next position on the depth map.

Figure 4.4 shows the meshing generation on a grid of stepsize 3, *i.e.* every 3rd pixel in each direction of the underlying depth map is used. In the example, the 3 D distance corresponding to all but the rightmost edge are below the threshold. Although Turk and Levoy recommend a threshold for the maximal edge length depending on the current points, we use a fixed threshold as it empirically gave better results.

Texturing of the meshes is done via vertex coloring, *i.e.* by associating the vertices to the color of the corresponding pixels in the RGB image. The rendering then takes care of the color interpolation inside every triangle of the mesh. Figure 4.3 shows the meshing and texturing results of a 3 D point cloud originating from the Microsoft Kinect.

4.4.2 Integration of local meshes

Local meshes are integrated into the global model by first aligning the meshes to the global model. For this, we transform the underlying 3 D points of the local mesh by the inverse of the camera pose associated to the RGB-D frame, *i.e.* with

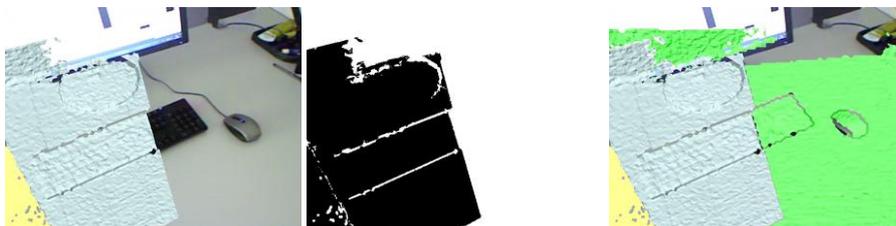


Figure 4.5: Determining the contribution of a candidate keyframe by rendering the global reconstruction into the current camera frame and determining the overlap between candidate keyframe and rendered depth map.

${}^{world}\mathbf{T}_{camera} = {}^{camera}\mathbf{T}_{world}^{-1}$. When performing real-time tracking, the camera is assumed to typically undergo small inter-frame movements. This means that there is a large overlap between meshes created from consecutive frames. For simplicity, we currently do *not* refine the existing vertices of the global mesh as *e.g.* proposed by Turk and Levoy, but instead simply add the aligned new meshes to the global mesh. We have to take care of the overlap in another way since otherwise, if for example the entire mesh of every captured RGB-D image would be added to the global mesh, the capacity of the main memory would be quickly exceeded because of the massive data volume.

We initially tried to mitigate this problem by only considering keyframes with associated camera poses that have a translation and/or rotation farther than a given threshold from the already meshed keyframes, as done in the tracking. This works well in many cases. But, choosing the threshold needs a compromise: if too high, it makes it harder to mesh some previously unmeshed parts. And if too low, the overlap between the regions becomes too large.

Instead of checking the position and the viewing angle of the camera, our criterion to accept a new keyframe respectively a new local mesh consists in the amount of non-overlapping geometry with respect to the global model. We implemented this criterion by filtering 3D points before we try to mesh them. As illustrated in figure 4.5, a binary mask is created by rendering the reconstructed scene from the current view point of the camera. Unmasked 3D points, *i.e.* points for which no geometry was rendered at their originating position in the keyframe, are directly considered for updating the mesh.

However, only relying on a binary mask would prohibit adding meshes of objects that are first observed in front of already reconstructed geometry. This happens for example when the camera is moved around a fixed object in an

attempt to scan its geometry or when an object is initially too close to a depth camera and thus no depth reading is available (*e.g.* the Microsoft Kinect does not give depth data for objects that are closer than 30 cm to it). To also add such closer objects, we keep the associated depth buffer of the rendering. For every masked pixel we check whether the depth stored in the depth buffer is greater than the value of the depth map by at least some threshold.

As can also be observed by close inspection of figure 4.5, small gaps may occur especially on the boundaries of registered depth maps. To close these gaps, we erode the binary mask directly after creation. New geometry may then also be added on these boundaries despite already existing geometry closer than the threshold.

Even though we avoid the creation of (massively) redundant meshes, still the decision of which frame should be integrated into the global reconstruction is of importance. One option is to process the frames continuously. The advantage of this method would be that the 3D model is created fluently and fast since the updates are small. Another possibility is, as discussed earlier, to decide on the basis of the current camera pose if the current frame should be processed or not. This possibility is very fast to evaluate. Since the meshing is done in a separate thread, we finally chose to process every incoming camera image, but only add those local meshes that contributed at least a certain number of triangles to the global mesh as this visually provided the best result.

4.5 Ground Truth-based evaluation

We evaluated the accuracy and precision of the camera motion estimation and the inherent scale of the proposed method using real camera data. The creation of the RGB-D ground truth data is explained in section 2. We recorded four sequences of a challenging industrial object that has multiple self-occlusions and is mainly composed of metallic, reflective and poorly textured surfaces. In the following, we will show the evaluation on these sequences. Besides the proposed method we also evaluated PTAM [KM07] on an RGB-only version of the sequences. Furthermore, we evaluated the recovered shape of our method against the known geometry of the industrial object.

4.5.1 Quantitative evaluation of the tracking

Our method is initialized on the first frame of each sequence and is able to track the complete sequences. To initialize PTAM, it is necessary to carefully move the camera a certain distance to establish an initial stereo configuration. The baseline of these frames affects the scale of the map that PTAM builds and thus the scale of the trajectory that PTAM estimates throughout the sequence. When evaluating PTAM, for all sequences, we used the first frame and varied the second image of the initial stereo setup from frame 1 to frame 50.

For some image pairs, the initialization of PTAM on the RGB images did not succeed. In contrast, thanks to the usage of the also available metric depth maps from the Kinect, the proposed method estimated an identical scene scale for all four sequences as shown in figure 4.6. As only a single frame is needed for the initialization, we tested how choosing this frame from the first 50 frames of each sequence would affect the scale factor needed for metric alignment of the trajectory. It turned out that the scale factor is relatively stable around 1.0 and has low variance. Note that the scale of PTAM (and any other RGB-only method in general) varies strongly depending on the image pair chosen for initialization.

Table 4.1 presents the results of the evaluation of the estimated camera pose. We show for the errors of the proposed method on all sequences as well as for several PTAM initializations. The evaluation shows that PTAM's accuracy and precision strongly depends on which image pairs are used for the initialization. With some image pairs, PTAM could not be initialized despite a rather large baseline between the frames. We show detailed results for the first sequence

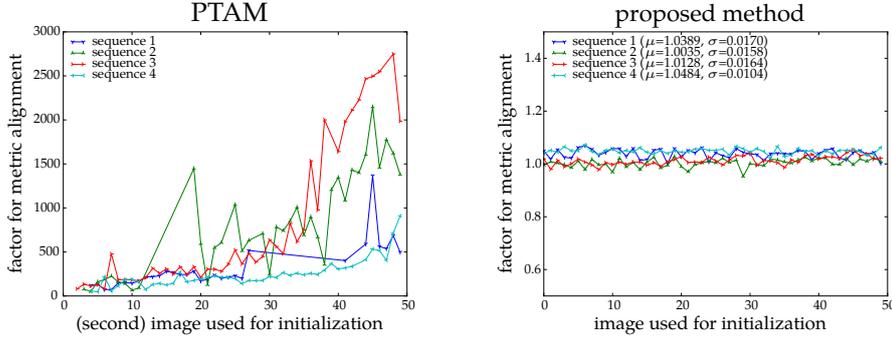


Figure 4.6: Comparison of the scale needed for aligning the trajectories of PTAM and the proposed method to the ground truth trajectory obtained from the mechanical measurement arm.

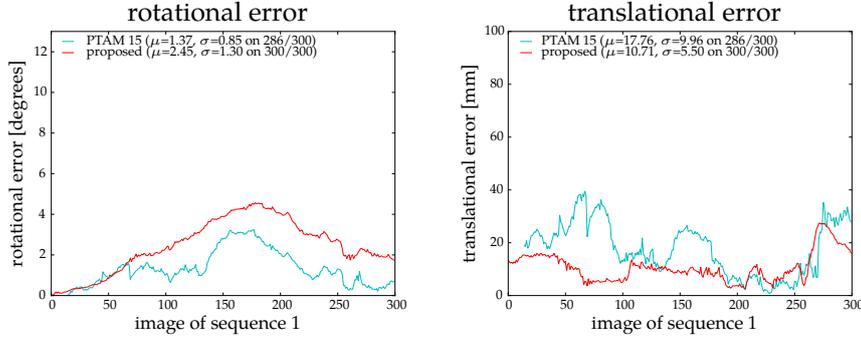


Figure 4.7: Evaluation of proposed method and PTAM against the first ground truth sequence.

in figure 4.7. Using the proposed method initialized on frame 0 and the best result that PTAM could achieve on this sequence (using frame 0 and frame 15), despite a good rotation estimation from PTAM, we are still getting much better translation estimation and our estimation of the camera trajectory with the proposed method is much closer to the GT trajectory.

4.5.2 Quantitative evaluation of the shape

As initially stated, we do not post-process the range images obtained from the Kinect other than neglecting samples further than 2 m. We extend the reconstruction only by adding new triangles to the mesh. We evaluated the quality of this reconstruction process by comparing the recovered mesh of the observed industrial object to known ground truth geometry. The reconstruction was ob-

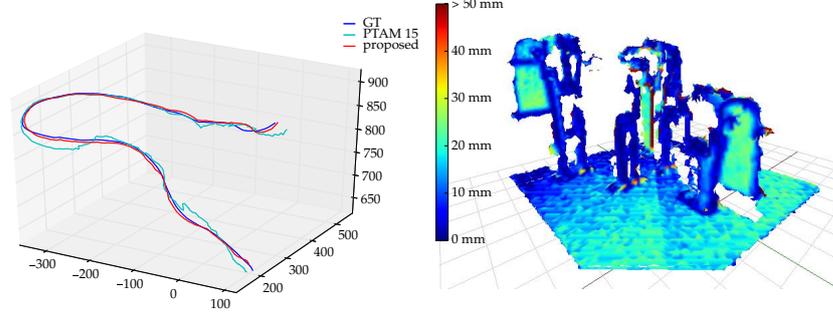


Figure 4.8: Evaluation against ground truth data. *Left*: Trajectories of proposed method and PTAM aligned to ground truth motion of measurement arm. *Right*: Evaluation of recovered shape against ground truth CAD model.

		rot. error [deg]		trans. error [mm]		images tracked
		μ	σ	μ	σ	
<i>seq01</i>	proposed	2.45	1.30	10.71	5.50	300/300
	PTAM 05	7.04	3.72	94.55	42.35	282/300
	PTAM 10	2.53	1.25	15.95	8.73	291/300
	PTAM 15	1.37	0.85	17.77	9.97	286/300
	PTAM 20	3.75	2.01	28.36	17.18	281/300
	PTAM 25	1.85	1.02	26.69	11.44	276/300
<i>seq02</i>	proposed	4.03	2.00	6.85	2.82	300/300
	PTAM 05	15.53	10.50	44.02	46.31	120/300
	PTAM 10	3.05	2.56	39.62	41.50	175/300
	PTAM 15	–	–	–	–	0/300
	PTAM 20	1.88	0.83	11.18	6.58	281/300
	PTAM 25	13.78	5.06	162.95	78.85	243/300
<i>seq03</i>	proposed	5.76	3.36	21.49	8.93	300/300
	PTAM 05	7.17	5.28	38.60	15.99	58/300
	PTAM 10	19.36	12.09	18.69	8.05	69/300
	PTAM 15	6.59	3.24	17.31	8.11	225/300
	PTAM 20	6.86	3.33	18.08	9.52	212/300
	PTAM 25	7.95	3.84	18.77	8.45	276/300
<i>seq04</i>	proposed	2.62	1.47	13.18	6.37	300/300
	PTAM 05	–	–	–	–	0/300
	PTAM 10	13.62	9.43	97.83	58.16	142/300
	PTAM 15	2.05	1.19	35.50	31.09	162/300
	PTAM 20	2.45	1.22	17.36	13.55	150/300
	PTAM 25	2.43	1.02	18.81	12.73	276/300

Table 4.1: Mean and variance of the error in rotation and translation of the proposed method and PTAM with different initializations. We used frames 0+5 for PTAM 05, 0+10 for PTAM 10 etc. The best results per sequence are highlighted in bold.

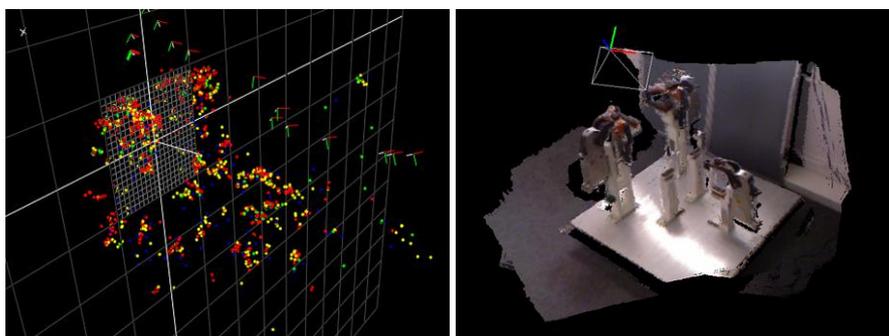


Figure 4.9: Reconstruction of the environment using PTAM [KM07] and the proposed method.

tained from the first sequence, using every 4th depth map pixel in each direction and a maximally allowed edge length of 50 mm. An update to the model was done when the new depth image provided more than 2k new triangles. The final model of the object contained around 14k vertices and 19k faces after removing the background. The recovered model was manually aligned to the known geometry for the evaluation. Next, we computed the discrepancy based on the point-to-plane distances of the reconstructed vertices to the known faces, visualized in figure 4.8. The median error was 9 mm, lower and upper quartiles 3 mm and 17 mm respectively. In the absence of any filtering mechanisms, these results should be regarded as upper bounds for Kinect-based reconstruction. This level of error is already acceptable for realistic occlusion in typical AR maintenance scenarios for mid-sized objects as can be observed in the supplementary material.

4.6 Application on different AR scenarios

We now demonstrate the proposed method on different AR scenes and scenarios. It has been tested on scenes similar to the one shown in the supplementary material and in figure 4.10. The initial map of the scene is built from the first image and extended afterwards. The tracking equally handles moderate and fast camera motion. The runtime of the tracking part of the system depends on the convergence of inter-frame feature tracking both from the last camera frame as well as from the closest keyframe to the current camera frame. On the desktop computer used for testing (Intel Core i7 2.8 GHz CPU), the inter-frame feature tracking takes around 23 ms on average for 200 tracked features. During fast

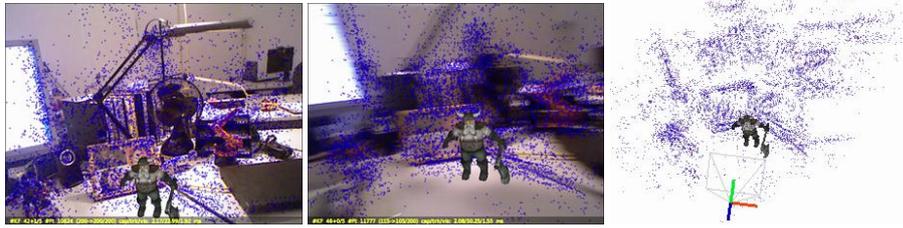


Figure 4.10: Evaluation using a live camera stream. *From left:* Tracking a desktop scenario with moderate and fast camera motion. The map reconstructed online consists of 63 keyframes and 14,775 reconstructed features.

motion, many features can be lost due to motion blur and the time per frame may reach 50-70 ms. The relocalization consists of matching the feature descriptors extracted from the current image to those of the map. This is currently done exhaustively, *i.e.* the runtime scales with the product of the number of features in the map and camera image. For the sequence, the relocalization together with reprojection from the closest keyframe takes between 25 ms and 230 ms for a map with around 15,000 mapped features from 63 keyframes.

4.6.1 AR-based virtual furniture trial

This scenario is meant to help the typical user who needs to virtually try new furniture (*e.g.* a closet) in the room before buying it. The user would not only check the color and the model of the furniture but also its size. This requires a correctly scaled camera pose estimation. Thanks to the proposed approach, the furniture can now be placed at the desired position with the correct scale, without modifying the environment. Furthermore, due to the reconstruction of the environment, the user gets a more realistic impression of the possible future look.

Figure 4.11 shows a correctly scaled shelf and chair augmented both without and with occlusion from real objects. To further assist the user, one could use the dense reconstruction also to restrict the movement of the virtual furniture such that *e.g.* it cannot be accidentally pushed “through” a wall. In case there are moving parts on the virtual furniture like doors or drawers, it could be automatically checked whether they can be operated using their full intended range of motion.



Figure 4.11: AR-based virtual furniture trial: correctly handling the occlusion thanks to the proposed parallel tracking and meshing of the environment makes the AR visualization much more realistic. While the camera moves, the camera motion is estimated and the environment model is updated.

4.6.2 Visual discrepancy check

Discrepancy check is of great use in an industrial application like prototyping. It is often required to visually compare a prototype with a produced model. Using AR allows to reduce the costs of construction since there is no need for manual as-is analysis by a construction engineer.

The presented example assumes a high precision of the tracking for which currently a mechanical measurement system like FaroArm is used best. However, for coarser discrepancy checks like *e.g.* the repositioned part shown in figure 4.12, the dense mesh created online by the proposed method is sufficient. Once the currently observed state of the object is registered to *e.g.* its constructed state, potential differences can be easily highlighted (*e.g.* marked red as in figure 4.12). Even simpler and also working in case there is no depth information of the current state of an object, one can use a virtual clipping plane to perform a visual discrepancy check.

4.6.3 Maintenance scenario

AR maintenance can be used as virtual replacement for a printed manual *e.g.* to guide a technician during a repair or to unmount an industrial machine. For this task, the single steps could be displayed *e.g.* as illustrated in figure 4.13, which shows the specific screws that should be loosened next. The realism of the augmentation can be again improved by using the meshed reconstruction of the environment as occlusion model. As a next step, one could think of combining

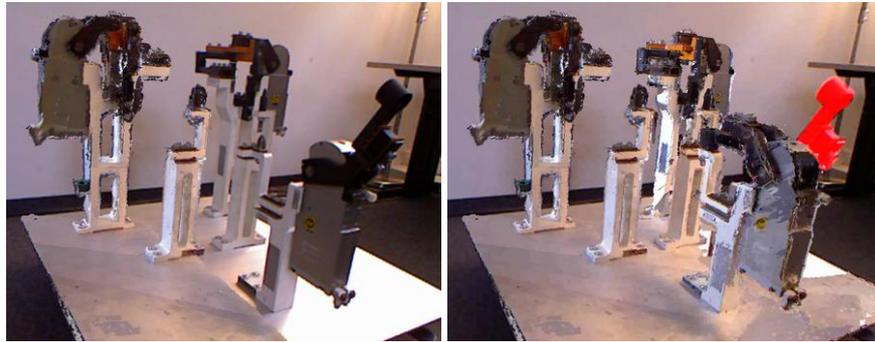


Figure 4.12: Live meshing allows to quickly create a textured meshed model of an object for visual discrepancy check. The main discrepancy of the previously scanned configuration and the current one is highlighted in red on the right image.



Figure 4.13: Maintenance instructions can be better understood when their occlusions are handled correctly as done for the lower of the two wrenches.

this scenario with the discrepancy check, *e.g.* only proceeding to the next step of a repair manual when the correct execution of the current step is validated using the current depth images of the RGB-D camera.

CHAPTER 5

FUTURE WORK

This chapter presents specific future steps for each of the main contributions of the thesis. The steps were revealed during the course of the thesis and present useful extensions of the presented methods.

Concerning the creation of ground truth datasets, a major improvement would be to remove the current limitation of 1,200 images per sequence. The current length of the sequences presented in chapter 2 was primarily based on the amount of addressable memory of the capturing computer and available 32 bit device drivers. The length could be increased by changing the software, *e.g.* via implementing a threaded writing to pre-allocated disk space. Alternatively, a hardware upgrade *e.g.* towards high-throughput parallel storage systems or solid state drives could further increase the recording bandwidth.

An increased duration of the sequences could be used in many ways. For example, it would enable a “static” marker-based synchronization of the sequences capturing 3D objects for SLAM-like methods. This approach would thus decouple the ground truth poses from the results of an algorithm. Extra slots in the sequences at the beginning and the end could further be used to capture calibration and/or synchronization objects. The current background-removal of the template-based dataset could then also be dropped in favor of a more realistic performance of the algorithms *e.g.* concerning real background clutter.

An evaluation based on the reprojection error enables an objective comparison of different methods. However, in case the target scenario is known, using an

error measure that is relative to the 3D dimension of the tracked object and/or of the planned augmentation may be more appropriate.

To estimate a 3D shape of a static object from images of a moving camera, two methods were presented in the thesis. The method presented in chapter 3 works well for static objects that can be fully visible from the top. As shown in the evaluation in section 3.4, it does not depend on as much texture as feature-based methods and thus has a wider range of use. The main source of error observed in the experiments originated from fast translational camera motion. The optimization then gets attracted by a local minimum as the assumption of small motion is considerably violated. We believe this could be mitigated by using a 3D motion model that predicts the camera pose in the next frame and thus initializes the optimization closer to the global minimum.

To further increase the robustness of the reconstruction, a possible direction could be a regularization term that penalizes deformation caused by errors in camera tracking; in other words, *e.g.* to only reconstruct when the recovered photometric error is low. Another interesting aspect of the regularization is the assumption that the shape of non-deformable and non-moving objects is recovered. Ideally, the method should stop estimating the shape at some point. Currently, the algorithm continues estimating the structure unless the user manually stops the estimation. However, as we have information about the variance of each vertex when computing its reference depth, it could be incorporated directly into the regularization term.

Another promising direction of future work is an analysis of the types of targets that are tracked and reconstructed well by the method of chapter 3. It is of great practical importance to identify a well-trackable target systematically or improve a badly-tracking target in a principled way, *e.g.* as done by Gruber *et al.* [GZW⁺10] for the feature-based mobile tracking system by Wagner *et al.* [WRM⁺08].

When given per-pixel estimates of both color/intensity and depth, the RGB-D method presented in chapter 4 can be employed to obtain camera pose and a reconstruction of the environment. This method can be improved by introducing refinement steps for both mesh and 3D point cloud. The 3D position of a feature used for tracking or a vertex used for meshing are currently determined from the aligned keyframe that first observed it. There is a variety of literature on mesh refinement, including the early works of Turk and Levoy [TL94] and Curless and Levoy [CL96]. An excellent tutorial for refinement of sparse feature maps

made from *e.g.* RGB images was published by Triggs *et al.* [TMHF00]. RGB-D images can be aligned also using a modified iterative closest point algorithm as *e.g.* proposed by Henry *et al.* [HKH⁺10].

In the RGB-D based method, we rely on sparse optical flow for inter-frame tracking and do not use an explicit 3 D motion model. The speed of the tracking is thus primarily depending on the convergence properties of the optical flow. On the one hand, this allows more flexible camera motions, but on the other hand this is computationally more expensive than template matching with a 3 D motion model. The amount of tracked features seems to also have the biggest influence on the precision of the pose. Therefore, one direction of future work may consist in analyzing how tracking more features with template matching instead of currently tracking fewer features with optical flow could affect the accuracy, speed and robustness of the system. Further, we currently do not distinguish between features used for relocalization and features used for tracking. As shown in the evaluation, this simplification works fairly well. However, it may be desirable to decouple these orthogonal components and thus track features that satisfy the requirements of only the tracking component.

Another part of future work may consist in an analysis of the error of the depth measurements from a consumer RGB-D camera like the Microsoft Kinect. From the evaluation of section 4.5, it is not clear whether the rather small offset from metric scale is coming directly from the Kinect or from the RGB-D method. Newcombe *et al.* [NIH⁺11] pre-processed the data with a bilateral filter to compensate the inaccuracies in the depth channel. It should be analyzed whether directly incorporating these uncertainties into the motion and structure estimation would further improve the result.

CHAPTER 6

CONCLUSION

This thesis has explored marker-less visual tracking for augmented reality. For conducting evaluations based on real images, a methodology for the creation of ground truth datasets was developed and applied on several existing tracking methods and on the two contributed methods focusing on real-time dense reconstruction of the environment. In the following, the thesis concludes by summarizing each contribution.

Chapter 2 presented a methodology to create datasets for evaluating both planar and non-planar tracking algorithms. The goal was to create image sequences with precisely known poses of the camera so that they can be used as objective ground truth to evaluate algorithms and enable fair comparisons. As key component, we used a highly precise measurement arm. For planar and small convex objects, we recorded ground truth sequences with an industrial global-shutter camera. The sequences feature realistic imaging conditions and motions. We carefully selected the level of texture of the chosen targets as well as the camera motions to create a representative dataset. Using these sequences, we evaluated several state-of-the-art algorithms. Furthermore, we created similar ground truth sequences using a RGB-D camera and an industrial object.

The template-based sequences can be used by the vision and AR communities. We offer them at <http://metaio.com/research> to give other researchers the opportunity to extensively evaluate template-based tracking algorithms and to enable an objective comparison of different methods. We provide the sequences both undistorted and containing the natural distortion of the lens. Furthermore, the distortion coefficients are also available for evaluating tracking algorithms

that explicitly model the distortion in their objective function. Since publication of the dataset, it was used by researchers of several universities and research institutes. For example, the dataset is used in the work of Dame and Marchand on template tracking using mutual information [DM10], the work of Megret *et al.* on gradient based image alignment [MAB10] and the work of Dupac and Matas on zero-shift interest points [DM11].

Our dataset can be used to complement manual testing as, despite of our ambitions to make it very general, the dataset represents only a selected subset of motions and targets. For example, the rotations only cover a quarter of the semisphere around the targets. While the dataset gives clues of an algorithm's performance in general, we suggest to still test the chosen algorithm(s) in the specific target scenario in a project setting.

We further presented two methods for real-time tracking and reconstruction. Chapter 3 presented a method for non-planar convex templates using intensity images, chapter 4 presented a method based on a RGB-D images. While the method of chapter 3 removes the planarity constraint inherent to classical template tracking, we still benefit from all available pixels of the template when building our objective function. No constraints are imposed on the model deformation, therefore we can equally reconstruct and track templates that are smooth or have creases. The performance of the proposed method was evaluated on both synthetic and real video sequences. Further, we performed quantitative analysis and compared the method to ground truth sequences which we obtained as described in chapter 2. These sequences were also used to compare standard planar template tracking methods and a state-of-the-art SLAM system.

Despite recovering only an approximate shape, the tracking precision increased and it also turned out to compute a more stable camera pose compared to the evaluated planar methods. Not surprisingly, because the method uses all pixels of the template image, it copes better with blur and low-textured surfaces than a sparse point-based method.

The real-time method based on a consumer RGB-D camera was presented in chapter 4. It estimates the camera motion with respect to an unknown environment while at the same time reconstructing a dense textured mesh of it. The system is initialized using a single frame, the reconstruction does not need constrained camera motion; especially pure rotational movement of the camera is handled transparently.

The scale of the reconstructed model is fixed and, in case of the Microsoft Kinect used in our experiments, seems to be at most 5–6% off from metric scale. We adopted the meshing algorithm of Turk and Levoy [TL94] for live incremental meshing of the environment and showed several AR scenarios that directly benefit from such a mesh. The proposed RGB-D tracking method and a state-of-the-art RGB-SLAM method were evaluated on sequences with ground truth camera motion and structure. In general, we obtain more accurate and precise results while tracking a higher number of images from the sequences. The method works already well and makes it possible to get satisfactory results despite the efficiency of the computations involved.

APPENDICES

TRANSFORMATIONS AND LIE ALGEBRA OF $\mathbb{SE}(3)$

This appendix describes rigid transformations between different coordinate frames and their notation used throughout the thesis. Further, a minimal parametrization using Lie algebra is presented.

The class of rigid transformations in \mathbb{R}^3 describes the spatial relationship between two objects, *e.g.* a camera and a target. We use the notation ${}^{camera}\mathbf{T}_{target}$ to indicate that a point \mathbf{x}_{target} is transformed from the coordinate frame of the target to the coordinate frame of the camera. Using homogeneous coordinates, this relationship is written as

$$\mathbf{x}_{camera} = {}^{camera}\mathbf{T}_{target} \mathbf{x}_{target} \quad (\text{A.1})$$

where ${}^{camera}\mathbf{T}_{target}$ is a (4×4) matrix and \mathbf{x} are (4×1) vectors.

The set of all rigid transformations in \mathbb{R}^3 is known as the Special Euclidean group $\mathbb{SE}(3)$. A transformation ${}^b\mathbf{T}_a \in \mathbb{SE}(3)$ consists of a (3×3) rotation matrix ${}^b\mathbf{R}_a$ and a (3×1) translation vector ${}^b\mathbf{t}_a \in \mathbb{R}^3$ as

$${}^b\mathbf{T}_a = \begin{bmatrix} {}^b\mathbf{R}_a & {}^b\mathbf{t}_a \\ \mathbf{0} & 1 \end{bmatrix} \quad (\text{A.2})$$

The rotation is a member of the Special Orthogonal group $\mathbb{SO}(3)$, *i.e.* for ${}^b\mathbf{R}_a$ holds that ${}^b\mathbf{R}_a^\top = {}^b\mathbf{R}_a^{-1} = {}^a\mathbf{R}_b$ and $\det({}^b\mathbf{R}_a) = 1$. The inverse of a transformation ${}^b\mathbf{T}_a$ thus becomes

$${}^b\mathbf{T}_a^{-1} = \begin{bmatrix} {}^b\mathbf{R}_a^\top & -{}^b\mathbf{R}_a^\top {}^b\mathbf{t}_a \\ \mathbf{0} & 1 \end{bmatrix}. \quad (\text{A.3})$$

For notational simplicity, we denote ${}^b\mathbf{T}_a$ as \mathbf{T} in the following. Using the Lie algebra $\mathfrak{se}(3)$ associated to $\mathbb{SE}(3)$, a pose $\mathbf{T}(\mathbf{a})$ with $\mathbf{a} = [a_1 \ a_2 \ a_3 \ a_4 \ a_5 \ a_6]$ can be parameterized as

$$\mathbf{T}(\mathbf{a}) = \text{expm} \left(\sum_{i=1}^6 a_i \mathbf{A}_i \right) \quad (\text{A.4})$$

with the $\text{expm}(\cdot)$ being the exponential map defined as

$$\text{expm}(\mathbf{A}) = \sum_{j=0}^{\infty} \frac{1}{j!} \mathbf{A}^j \quad (\text{A.5})$$

for any $(n \times n)$ matrix \mathbf{A} . We now choose \mathbf{A}_i as the generator matrices of $\mathbb{SE}(3)$, namely

$$\mathbf{A}_1 = \begin{bmatrix} [\mathbf{e}_1]_x & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \quad \mathbf{A}_2 = \begin{bmatrix} [\mathbf{e}_2]_x & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \quad \mathbf{A}_3 = \begin{bmatrix} [\mathbf{e}_3]_x & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \quad (\text{A.6})$$

$$\mathbf{A}_4 = \begin{bmatrix} \mathbf{0} & \mathbf{e}_1 \\ \mathbf{0} & 0 \end{bmatrix} \quad \mathbf{A}_5 = \begin{bmatrix} \mathbf{0} & \mathbf{e}_2 \\ \mathbf{0} & 0 \end{bmatrix} \quad \mathbf{A}_6 = \begin{bmatrix} \mathbf{0} & \mathbf{e}_3 \\ \mathbf{0} & 0 \end{bmatrix} \quad (\text{A.7})$$

with $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ being the canonical basis for \mathbb{R}^3 and $[\cdot]_x$ constructing a skew symmetric matrix as

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_x = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}. \quad (\text{A.8})$$

The rotation is represented by a_1 to a_3 while a_4 to a_6 represent the translation of \mathbf{T} . This parametrization is singularity-free and has the advantage that a partial differentiation around the origin ($\mathbf{a} = \mathbf{0}$) as needed for minimization is trivial, as $\frac{\partial}{\partial a_i} \mathbf{T}(\mathbf{a}) = \mathbf{A}_i$. Compared to other parameterizations of *e.g.* rotation, this is favorable in terms of computational effort and numerical stability.

Furthermore, there are closed-form solutions available [Agr06] for evaluating $\mathbf{T}(\mathbf{a})$. The forms were obtained by identifying the series of sine and cosine in the

series resulting from applying the matrix exponential. Using the partitioning of \mathbf{a} into $\boldsymbol{\omega} = [a_1 \ a_2 \ a_3]^\top$ and $\boldsymbol{\mu} = [a_4 \ a_5 \ a_6]^\top$ for readability, we have

$$\mathbf{T}(\mathbf{a}) = \expm \left(\begin{bmatrix} [\boldsymbol{\omega}]_x & \boldsymbol{\mu} \\ \mathbf{0} & 0 \end{bmatrix} \right) = \begin{bmatrix} \mathbf{R}(\boldsymbol{\omega}) & \mathbf{B}(\boldsymbol{\omega})\boldsymbol{\mu} \\ \mathbf{0} & 1 \end{bmatrix} \quad (\text{A.9})$$

with

$$\mathbf{R}(\boldsymbol{\omega}) = \mathbf{I} + \frac{\sin \|\boldsymbol{\omega}\|}{\|\boldsymbol{\omega}\|} [\boldsymbol{\omega}]_x + \frac{1 - \cos \|\boldsymbol{\omega}\|}{\|\boldsymbol{\omega}\|^2} [\boldsymbol{\omega}]_x^2 \quad (\text{A.10})$$

$$\mathbf{B}(\boldsymbol{\omega}) = \mathbf{I} + \frac{1 - \cos \|\boldsymbol{\omega}\|}{\|\boldsymbol{\omega}\|^2} [\boldsymbol{\omega}]_x + \frac{\|\boldsymbol{\omega}\| - \sin \|\boldsymbol{\omega}\|}{\|\boldsymbol{\omega}\|^3} [\boldsymbol{\omega}]_x^2 \quad (\text{A.11})$$

for $\boldsymbol{\omega} \neq \mathbf{0}$ and where \mathbf{I} is a (3×3) identity matrix. For the case of $\boldsymbol{\omega} = \mathbf{0}$, we define $\mathbf{R}(\mathbf{0}) = \mathbf{B}(\mathbf{0}) = \mathbf{I}$.

Please refer to [Var84] for a more general introduction to Lie algebras.

APPENDIX **B**

INTERPOLATION

We define an $(U \times V)$ image \mathcal{I} to provide intensity values at integer pixel positions $\mathbf{u} = [u \ v \ 1]^\top$ for $u \in \{1, \dots, U\}$ and $v \in \{1, \dots, V\}$. We use normalized homogeneous 2D coordinates for accessing pixel intensities, *i.e.* the coordinates have to be scaled such that the last element is equal to one. The measured intensities are associated to the center of a pixel.

To obtain sub-pixel accuracy, intensities usually also have to be associated for pixel positions within the imaged region that are not referring to pixel centers, *i.e.* for $u \in [\frac{1}{2}, U + \frac{1}{2}]$ and $v \in [\frac{1}{2}, V + \frac{1}{2}]$. Intensities for these sub-pixel locations are provided by interpolating the closest intensities available. In the following, two basic interpolation schemes are shown which are often used in real-time computer vision algorithms.

Nearest Neighbour Interpolation

A simple interpolation scheme consists of using the nearest neighbouring pixel at integer coordinates:

$$\mathcal{I}(\mathbf{u}) = \mathcal{I}\left(\left\lfloor \mathbf{u} + \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}^\top \right\rfloor\right) \quad (\text{B.1})$$

where $\lfloor \cdot \rfloor$ is the floor function defined as

$$\lfloor a \rfloor = b \mid b \leq a < b + 1 \quad \text{with } a \in \mathbb{R}, b \in \mathbb{Z}. \quad (\text{B.2})$$

This scheme is very fast to execute. However, as can be seen on figure B.1, nearest neighbour interpolation tend to produces non-smooth images.

Bilinear Interpolation

This scheme uses the four closest integer neighbours of \mathbf{u} :

$$\mathcal{I}(\mathbf{u}) = \begin{bmatrix} 1 - \delta_v \\ \delta_v \end{bmatrix}^\top \begin{bmatrix} \mathcal{I}(\mathbf{z}_{0,0}(\mathbf{u})) & \mathcal{I}(\mathbf{z}_{1,0}(\mathbf{u})) \\ \mathcal{I}(\mathbf{z}_{0,1}(\mathbf{u})) & \mathcal{I}(\mathbf{z}_{1,1}(\mathbf{u})) \end{bmatrix} \begin{bmatrix} 1 - \delta_u \\ \delta_u \end{bmatrix} \quad (\text{B.3})$$

with $\mathbf{z}_{u,v}(\cdot)$ obtaining the nearest neighbour pixel intensities as

$$\mathbf{z}_{u,v}(\mathbf{u}) = \lfloor \mathbf{u} \rfloor + \begin{bmatrix} u \\ v \\ 0 \end{bmatrix} \quad \text{for } u, v \in \mathbb{Z}. \quad (\text{B.4})$$

The weights δ_u and δ_v are computed from the distance

$$\begin{bmatrix} \delta_u \\ \delta_v \\ 0 \end{bmatrix} = \mathbf{u} - \mathbf{z}_{0,0}(\mathbf{u}). \quad (\text{B.5})$$

Bilinear interpolation yields smoother results than nearest neighbour interpolation. It is still fast to execute. However, it is not differentiable at integer pixel coordinates.



Figure B.1: Example for interpolation. *From left:* vector image of the letter “a”, nearest-neighbour and bilinear interpolation to a resolution of (40×43) .

PHYSICAL IMAGING PROCESS

A digital camera creates a two dimensional image by measuring the light intensity on each cell of its photoactive image sensor. Before hitting the sensor, the incoming light passes the lens(es) and diaphragm as shown in figure C.1. The lens focuses the incoming light on the sensor plane. Parts of the lens can be moved perpendicular to the image sensor to adjust the focus, *i.e.* the minimal depth relative to the sensor in which objects are imaged sharply. Zoom lenses have additional movable lenses to further change the field of view of the camera.

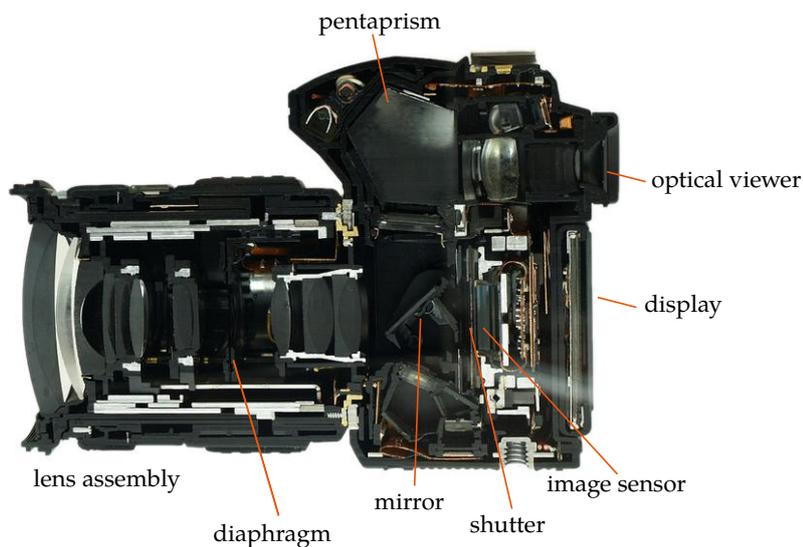


Figure C.1: Annotated cut-through of digital camera. The light passes the lens assembly and the diaphragm until it is measured by the sensor. Based on [dsl].

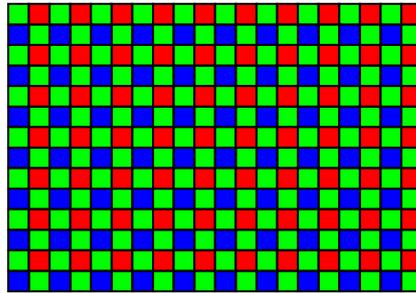


Figure C.2: Sample Bayer filter used to create a color image from a single image sensor. Only selected wavelengths pass the filters to each image sensor element. The color image is created by interpolation.

The diaphragm is used to adjust the amount of light and the depth of field, *i.e.* the distance between the nearest and farthest object relative to the camera that are in focus. When the light hits the image sensor, the photoactive elements convert the light into the electric charges. The charges in each of its cells are then sampled, amplified, discretized and mapped into a pre-calibrated color space to create the image \mathcal{I}_{raw} . The sampling is done either at once for all pixels (*global shutter*) or continuously (*rolling shutter*). In the latter case, the pixels of an image are not taken at the same time, which increases artifacts related to motion.

Depending on the amount of light and the sensitivity of the sensor (*signal gain / ISO setting*), the exposure of an image *e.g.* with a modern digital single-lens reflex camera [Can] (DSLR) may take from 1/8000s to 30s (*shutter speed*). A high gain/ISO setting reduces the time needed to capture the image. On the other hand, the level of sensor noise increases notably with higher ISO settings (see figure 2.2). In case the camera or part of the scene moves while an image is captured, the same object is observed in multiple poses and thus blur along the direction of the relative motion is apparent.

For color cameras, filters are mounted on top of the image sensor such that certain cells mainly respond to a range of wavelengths corresponding to either green, blue or red spectra. There are two possibilities to create a color image: Dedicated sensor cells for each colored pixel, *e.g.* three cells per RGB pixel, or a 1:1 sensor cell–colored pixel relationship by using a filter that combines the readings of neighbouring sensor cells into a colored pixel. The former is typically realized by separating the wavelengths with a prism and using three dedicated image sensors. The latter coloring technique employs a *Bayer* filter, an example is shown in figure C.2. This filter typically uses twice as many green cells as red and blue cells as the human eye is most sensitive to green.

Using the RGB color space, a $(U \times V)$ color image \mathcal{I}_c consists of the three channels $\mathcal{I}_{c|red}$, $\mathcal{I}_{c|green}$ and $\mathcal{I}_{c|blue}$ of the same resolution. The $(U \times V)$ raw image \mathcal{I}_{raw} is decomposed into three images $\mathcal{I}_{raw|co}$ for $co \in \{red, green, blue\}$. The resolution of each decomposed image are $(f_{co}U \times f_{co}V)$ where the scalar f_{co} depends on the structure of the filter. For a filter with twice as many green samples the factors are $f_{green} = \frac{1}{2}$ and $f_{red} = f_{blue} = \frac{1}{4}$. Each channel of \mathcal{I}_c is created from the decomposed \mathcal{I}_{raw} as

$$\mathcal{I}_{c|co}(\mathbf{u}) = \mathcal{I}_{raw|co}(f_{co}\mathbf{u}) \quad (\text{C.1})$$

for all pixel positions $\mathbf{u} = [u \ v \ 1]^\top$ with $u \in \{1, \dots, U\}$ and $v \in \{1, \dots, V\}$. The intensities at sub-pixel locations in the images $\mathcal{I}_{raw|co}$ may be interpolated as described in appendix B.

APPENDIX D

IMAGE CREDITS

The images we used for the template-based tracking dataset of section 2.6.1 were downloaded from <http://flickr.com>. They are licensed under a creative commons licence that allows commercial use and adaption given the authors of the original work are mentioned. Since publication of the ground truth dataset, two of these images were removed from flickr. The following table provides an overview:

<i>Target</i>	<i>Author/Username</i>	<i>Source</i>
	Richard Drdul	http://www.flickr.com/photos/drdul/180849815/
	Robyn Gallagher	http://www.flickr.com/photos/robyn-gallagher/185435145/
	redjar	http://www.flickr.com/photos/redjar/113728402/
	<i>unknown</i>	Flickr-ID 2676908272_3acda48db1_o
	maazbot	http://flickr.com/photos/maazbot/218926578/
	<i>unknown</i>	Flickr-ID 390864316_c117e70442_o
	AliSabki	http://www.flickr.com/photos/27711119@N07/2584365102/
	Boby Dimitrov	http://www.flickr.com/photos/bobydimitrov/2269383646/

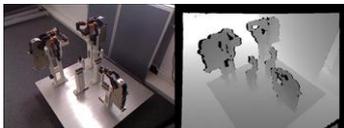
SUPPLEMENTARY MATERIAL

In the following, a list of the supplementary videos is given. The videos are available for download at <http://lieberknecht.net/dissertation>. The content of each video is summarized.

E.1 Ground Truth for AR tracking methods

The following videos demonstrate the proposed methodology of chapter 2. Excerpts of the 40 sequences created for the evaluation of template-based tracking and detection methods are shown in section 2.6.1. All videos of this ground truth dataset are available for download at <http://metaio.com/research>.

The following videos demonstrate the ground truth RGB-D sequences we created to evaluate the method presented in chapter 4.



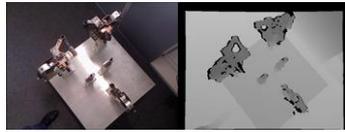
gt_rgbd_seq01.mp4

This video shows the first RGB-D sequence with known camera pose.



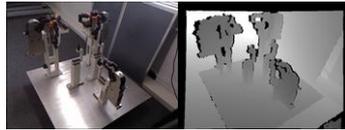
gt_rgbd_seq02.mp4

This video shows the second RGB-D sequence with known camera pose.



gt_rgbd_seq03.mp4

This video shows the third RGB-D sequence with known camera pose.

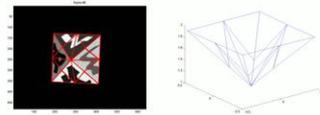


gt_rgbd_seq04.mp4

This video shows the fourth RGB-D sequence with known camera pose.

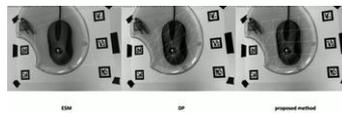
E.2 Dense deformable template tracking

The following videos demonstrate the proposed method of chapter 3.



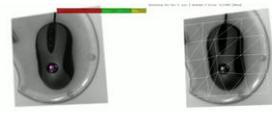
ddt_synthetic_sequence.mp4

This video shows the tracking and reconstruction of the proposed method on a synthetic sequence.



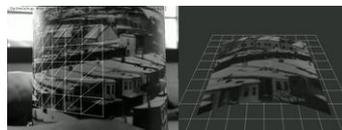
ddt_comparison_planar.mp4

This video shows a quantitative comparison of two planar template tracking methods [BM07, BM06] to the proposed method on real image data with known ground truth camera pose.



ddt_comparison_PTAM.mp4

This video shows a quantitative comparison of the proposed method and PTAM [KM07] on a cropped and blurred sequence with known ground truth camera pose.



ddt_occlusion.mp4

This video shows the reconstruction and tracking of a real object. The proposed method succeeded to track despite up to 50% occlusion of the target.



ddt_deforming_object.mp4

This video shows the proposed method tracking and reconstructing a deformed object. The method copes well with minor deformations despite assuming object-rigidity.

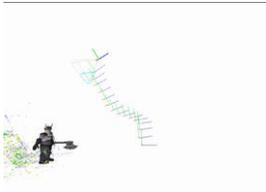
E.3 RGB-D based tracking and meshing

The following videos demonstrate the proposed method of chapter 4.



rgbd_tracking.mp4

This video shows the creation of the sparse map, re-localization and tracking while the camera is maneuvered rapidly.



rgbd_adding_keyframes.mp4

This video shows how a sparse map is created. New keyframes are added based on translational and rotational distance to registered keyframes.



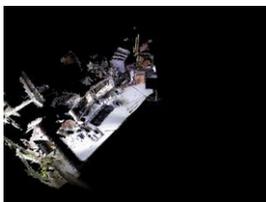
rgbd_tracking_keyframes.mp4

This video shows the reprojection of features from keyframes. Reprojected features are highlighted in yellow.



rgbd_occlusion.mp4

This video shows the creation of a dense map. After tracking a scene, a real box is added to the tracked scene. The box is reconstructed, its mesh is then used to occlude a virtual object.



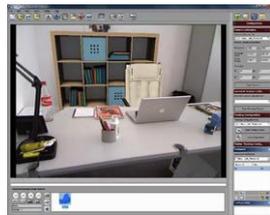
rgbd_scanning.mp4

This video shows the live creation of a dense map. An RGB-D camera is maneuvered in an office. The mesh is updated by integrating all non-overlapping mesh segments from the current camera frame.



rgbd_ar_maintenance.mp4

This video shows how occlusion of virtual objects improves an AR maintenance scenario. Initially, the augmentations are not occluded. Once the object is reconstructed, the depth perception and thus usefulness of the scenario improves.



rgbd_virtual_furniture.mp4

This video shows how occlusion of virtual objects improves an scenario with virtual furniture. Similar to the previous scenario, the depth perception notably improves once the real environment is meshed.

REFERENCES

- [ABB⁺01] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre, *Recent advances in augmented reality*, *CG&A* **21** (2001), 34–47. (cited on p. 1)
- [Agr06] Motilal Agrawal, *A Lie algebraic approach for consistent pose registration for motion estimation*, IROS, 2006. (cited on p. 98)
- [AIC] AICON 3D Systems GmbH, <http://aicon.de/>. (cited on p. 16)
- [All] Allied Vision Technologies GmbH, <http://alliedvisiontec.com/>. (cited on p. 15)
- [Azu97] Ronald Azuma, *A survey of augmented reality*, *Presence* **6** (1997), no. 4, 355–385. (cited on p. 1)
- [Ble] Blender Foundation, <http://blender.org>. (cited on p. 8)
- [BM01] Simon Baker and Iain Matthews, *Equivalence and efficiency of image alignment algorithms.*, *CVPR*, 2001. (cited on p. 11)
- [BM04] ———, *Lucas-Kanade 20 years on: A unifying framework*, *IJCV* **56** (2004), no. 3, 221–255. (cited on p. 11, 45, 47)
- [BM06] Selim Benhimane and Ezio Malis, *Integration of Euclidean constraints in template based visual tracking of piecewise-planar scenes*, IROS, 2006. (cited on p. 45, 47, 54, 59, 60, 110)
- [BM07] ———, *Homography-based 2d visual tracking and servoing*, Special Joint Issue IJCV/IJRR on Robot and Vision. Published in The International Journal of Robotics Research **26** (2007), no. 7, 661–676. (cited on p. 3, 36, 47, 59, 60, 110)
- [Bou99] Jean-Yves Bouguet, *Pyramidal implementation of the Lucas-Kanade feature tracker*, OpenCV Documentation, 1999. (cited on p. 70)
- [BSL⁺07] Simon Baker, Daniel Scharstein, J.P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski, *A database and evaluation methodology for optical flow*, *ICCV*, 2007, pp. 1–8. (cited on p. 11, 36)
- [BTG06] Herbert Bay, Tinne Tuytelaars, and Luc J. Van Gool, *SURF: Speeded up robust features*, *ECCV*, 2006, pp. 404–417. (cited on p. 11, 36)
- [BZ04] Adrien Bartoli and Andrew Zisserman, *Direct estimation of non-rigid registrations*, *BMVC*, 2004. (cited on p. 45)

- [Can] Canon U.S.A., Inc., *Specifications of EOS-1D Mark III*, http://www.usa.canon.com/cusa/consumer/products/cameras/slr_cameras/eos_1d_mark_iv. (cited on p. 104)
- [CDM08] Javier Civera, Andrew J. Davison, and J. M. M. Montiel, *Unified inverse depth parametrization for monocular slam*, *IEEE Transactions on Robotics* **24** (2008), no. 5, 932–945. (cited on p. 66)
- [CKM08] Robert Castle, Georg Klein, and David Murray, *Video-rate localization in multiple maps for wearable augmented reality*, *IEEE Int. Symp. on Wearable Computers*, 2008. (cited on p. 69)
- [CL96] Brian Curless and Marc Levoy, *A volumetric method for building complex models from range images*, *SIGGRAPH*, 1996. (cited on p. 88)
- [CM11] Robert Castle and David Murray, *Keyframe-based recognition and localization during video-rate parallel tracking and mapping*, *Image and Vision Computing* **29** (2011), 524 – 532. (cited on p. 7)
- [CMN11] Victor Castaneda, Diana Mateus, and Nassir Navab, *SLAM combining ToF and high-resolution cameras*, *IEEE Workshop on Motion and Video Computing*, 2011. (cited on p. 69)
- [Dav03] Andrew Davison, *Real-time simultaneous localisation and mapping with a single camera*, *ICCV*, 2003. (cited on p. 65, 68)
- [DC02] Tom Drummond and Roberto Cipolla, *Real-time visual tracking of complex structures*, *PAMI* **24** (2002), 932 – 946. (cited on p. 3)
- [DD95] Daniel DeMenthon and Larry Davis, *Model-based object pose in 25 lines of code*, *IJCV* **15** (1995), no. 1, 123–141. (cited on p. 41, 59, 75)
- [DM10] Amaury Dame and Eric Marchand, *Accurate real-time tracking using mutual information*, *ISMAR*, 2010, pp. 47–56. (cited on p. 92)
- [DM11] Jan Dupac and Jiri Matas, *Ultra-fast tracking based on zero-shift points.*, *ICASSP*, 2011. (cited on p. 92)
- [DMM03] Andrew Davison, Walterio Mayol, and David Murray, *Real-time localisation and mapping with wearable active vision*, *ISMAR*, 2003. (cited on p. 68)
- [DRMS07] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse, *MonoSLAM: Real-time single camera SLAM*, *PAMI* **26** (2007), no. 6, 1052–1067. (cited on p. 48)

- [DSK08] Ankur Datta, Yaser Sheikh, and Takeo Kanade, *Linear motion estimation for systems of articulated planes*, CVPR, 2008. (cited on p. 47, 62)
- [dsl] <http://bestphotosaroundtheworld.blogspot.de/2011/06/cutaway-of-olympus-e-30-dslr.html>. (cited on p. 103)
- [DT05] Navneet Dalal and Bill Triggs, *Histograms of oriented gradients for human detection*, CVPR, 2005. (cited on p. 11)
- [ED08] Ethan Eade and Tom Drummond, *Unified loop closing and recovery for real time monocular SLAM*, BMVC, 2008. (cited on p. 69)
- [FARa] FARO Europe GmbH and Co. KG, <http://faro.com/>. (cited on p. 14)
- [FARb] FARO Technologies, Inc., *Basic measurement training workbook*, <http://www.scribd.com/doc/46495149/3/Chapter-3-Introduction-to-CAM2-Measure-X>. (cited on p. 18)
- [FB81] Martin Fischler and Robert Bolles, *Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography*, Communications of the ACM **24** (1981), no. 6, 381–395. (cited on p. 41, 75)
- [Fia10] Mark Fiala, *Designing highly reliable fiducial markers*, PAMI **32** (2010), no. 7, 1317–1324. (cited on p. 2)
- [GBBS09] Vincent Gay-Bellile, Adrien Bartoli, and Patrick Sayd, *Direct estimation of non-rigid registrations with image-based self-occlusion reasoning*, PAMI **32** (2009), 87–104. (cited on p. 47)
- [GGV⁺10] Lukas Gruber, Steffen Gauglitz, Jonathan Ventura, Stefanie Zollmann, Manuel Huber, Michael Schlegel, Gudrun Klinker, Dieter Schmalstieg, and Tobias Höllerer, *The city of sights: Design, construction, and measurement of an augmented reality stage set*, ISMAR, 2010. (cited on p. 12)
- [GHT11] Steffen Gauglitz, Tobias Höllerer, and Matthew Turk, *Evaluation of interest point detectors and feature descriptors for visual tracking*, IJCV **94** (2011), no. 3, 335–360. (cited on p. 13)
- [GZW⁺10] Lukas Gruber, Stefanie Zollmann, Daniel Wagner, Dieter Schmalstieg, and Tobias Höllerer, *Evaluating the trackability of natural feature-point sets*, ICPR, 2010. (cited on p. 88)

- [Hee87] David J. Heeger, *Model for the extraction of image flow*, *Journal of the Optical Society of America A* **4** (1987), no. 8, 1455–1471. (cited on p. 11)
- [HKH⁺10] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox, *RGB-D Mapping: Using depth cameras for dense 3D modeling of indoor environments*, *Int. Symposium on Experimental Robotics*, 2010. (cited on p. 69, 89)
- [HSE10] Anna Hilsmann, David Schneider, and Peter Eisert, *Realistic cloth augmentation in single view under occlusion*, *Computers & Graphics* **35** (2010), no. 5, 567–574. (cited on p. 47)
- [JD02] Frederic Jurie and Michel Dhome, *Hyperplane approximation for template matching*, *PAMI* **24** (2002), 996–1000. (cited on p. 45)
- [KB99] Hirokazu Kato and Mark Billinghurst, *Marker tracking and hmd calibration for a video-based augmented reality conferencing system*, *Proceedings of the 2nd International Workshop on Augmented Reality (IWAR 99)*, 1999. (cited on p. 2)
- [Kle06] Georg Klein, *Visual tracking for augmented reality*, Ph.D. thesis, University of Cambridge, 2006. (cited on p. 2)
- [KM07] Georg Klein and David Murray, *Parallel tracking and mapping for small AR workspaces*, *ISMAR*, 2007. (cited on p. 7, 25, 46, 48, 58, 65, 68, 80, 83, 110)
- [LBI11] Sebastian Lieberknecht, Selim Benhimane, and Slobodan Ilic, *Simultaneous reconstruction and tracking of non-planar templates*, *DAGM*, 2011. (cited on p. 27, 33)
- [LBMN09] Sebastian Lieberknecht, Selim Benhimane, Peter Meier, and Nassir Navab, *A dataset and evaluation methodology for template-based tracking algorithms*, *ISMAR*, 2009. (cited on p. 27)
- [LBMN11] ———, *Benchmarking template-based tracking algorithms*, *International Journal of Virtual Reality, Special Issue on Augmented Reality* **15** (2011), 99–108. (cited on p. 27)
- [LF06] Vincent Lepetit and Pascal Fua, *Keypoint Recognition using Randomized Trees*, *PAMI* **28** (2006), no. 9, 1465–1479. (cited on p. 3)
- [LHIB11] Sebastian Lieberknecht, Andrea Huber, Slobodan Ilic, and Selim Benhimane, *RGB-D camera-based parallel tracking and mapping*, *ISMAR*, 2011. (cited on p. 27)

- [LK81] Bruce Lucas and Takeo Kanade, *An iterative image registration technique with an application to stereo vision*, IJCAI, 1981. (cited on p. 11, 45, 47, 70)
- [Low04] David G. Lowe, *Distinctive image features from scale-invariant keypoints*, IJCV **60** (2004), no. 2, 91–110. (cited on p. 3, 36, 72, 75)
- [Luh03] Thomas Luhmann, *Nahbereichsphotogrammetrie (close range photogrammetry)*, Wichmann, 2003. (cited on p. 16)
- [MAB10] Remi Megret, Jean-Baptiste Authesserre, and Yannick Berthoumieu, *Bidirectional composition on lie groups for gradient-based image alignment*, IEEE Transactions on Image Processing **19** (2010), 2369 – 2381. (cited on p. 92)
- [MP07] Pierre Moreels and Pietro Perona, *Evaluation of features detectors and descriptors based on 3d objects*, IJCV **73** (2007), no. 3, 263–284. (cited on p. 11)
- [MS04] Krystian Mikolajczyk and Cordelia Schmid, *Scale and affine invariant interest point detectors*, IJCV **60** (2004), no. 1, 63–86. (cited on p. 11)
- [ND10] Richard Newcombe and Andrew Davison, *Live dense reconstruction with a single moving camera*, CVPR, 2010. (cited on p. 48, 69)
- [NIH⁺11] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon, *Kinectfusion: Real-time dense surface mapping and tracking*, ISMAR, 2011. (cited on p. 69, 89)
- [NLD11] Richard A. Newcombe, Steven Lovegrove, and Andrew J. Davison, *DTAM: Dense tracking and mapping in real-time*, ICCV, 2011. (cited on p. 48, 69)
- [NS06] David Nister and Henrik Stewenius, *Scalable recognition with a vocabulary tree*, CVPR, 2006. (cited on p. 11)
- [OFL07] Mustafa Ozuysal, Pascal Fua, and Vincent Lepetit, *Fast keypoint recognition in ten lines of code*, CVPR, 2007, pp. 1–8. (cited on p. 36)
- [PCB02] Christopher J. Price, Ian D. Coope, and David Byatt, *A convergent variant of the nelder-mead algorithm*, J. Optim. Theory Appl. **113** (2002), no. 1, 5–19. (cited on p. 23, 39)

- [PLF08] Julien Pilet, Vincent Lepetit, and Pascal Fua, *Fast non-rigid surface detection, registration and realistic augmentation*, IJCV **76** (2008), no. 2, 109–112. (cited on p. 47)
- [PMK06] Katharina Pentenrieder, Peter Meier, and Gudrun Klinker, *Analysis of tracking accuracy for single-camera square-marker-based tracking*, Proc. Dritter Workshop Virtuelle und Erweiterte Realität der GI-Fachgruppe VR/AR, 2006. (cited on p. 2, 23)
- [RD05] Edward Rosten and Tom Drummond, *Fusing points and lines for high performance tracking.*, ICCV, vol. 2, October 2005, pp. 1508–1511. (cited on p. 61)
- [SCD⁺06] Steve M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski, *A comparison and evaluation of multi-view stereo reconstruction algorithms*, CVPR, 2006, pp. 519–528. (cited on p. 11)
- [SF09] Matthieu Salzmann and Pascal Fua, *Reconstructing sharply folding surfaces: A convex formulation*, CVPR, 2009. (cited on p. 46, 47)
- [SM10] Geraldo Silveira and Ezio Malis, *Unified direct visual tracking of rigid and deformable surfaces under generic illumination changes in grayscale and color images*, IJCV **89** (2010), no. 1, 84–105. (cited on p. 45, 47)
- [SMD10] Hauke Strasdat, J. M. M. Montiel, and Andrew J. Davison, *Real-time monocular slam: Why filter?*, ICRA, 2010. (cited on p. 48, 68)
- [SS00] Heung-Yeung Shum and Richard Szeliski, *Construction of panoramic image mosaics with global and local alignment.*, IJCV **16** (2000), 63–84. (cited on p. 11)
- [Ste99] Charles Stewart, *Robust parameter estimation in computer vision*, SIAM Review **41** (1999), no. 3, 513–537. (cited on p. 39, 71)
- [SUF08] Mathieu Salzmann, Raquel Urtasun, and Pascal Fua, *Local deformation models for monocular 3D shape recovery*, CVPR, 2008. (cited on p. 46, 47)
- [SZS⁺08] Richard Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall Tappen, and Carsten Rother, *A comparative study of energy minimization methods for markov random fields with smoothness-based priors*, PAMI **30** (2008), no. 6, 1068–1080. (cited on p. 11)

- [TL89] Roger Y. Tsai and Reimar K. Lenz, *A new technique for fully autonomous and efficient 3d robotics hand/eye calibration*, IEEE Transactions on Robotics and Automation **5** (1989), no. 3, 345–358. (cited on p. 19)
- [TL94] Greg Turk and Marc Levoy, *Zippered polygon meshes from range images*, Computer Graphics Proceedings, Annual Conference Series, 1994. (cited on p. 77, 88, 93)
- [TMHF00] Bill Triggs, Philip F. Mclauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon, *Bundle adjustment – a modern synthesis*, Proceedings of the International Workshop on Vision Algorithms: Theory and Practice, 2000. (cited on p. 48, 68, 89)
- [Ume91] Shinji Umeyama, *Least-squares estimation of transformation parameters between two point patterns*, PAMI **13** (1991), 376–380. (cited on p. 39)
- [Var84] Veeravalli S. Varadarajan, *Lie groups, lie algebras and their representations (graduate texts in mathematics)*, Springer-Verlag, 1984. (cited on p. 99)
- [VF08] Andrea Vedaldi and Brian Fulkerson, *VLFeat: An open and portable library of computer vision algorithms*, <http://www.vlfeat.org/>, 2008. (cited on p. 41)
- [VSTF09] Aydin Varol, Mathieu Salzmann, Engin Tola, and Pascal Fua, *Template-free monocular reconstruction of deformable surfaces*, ICCV, 2009. (cited on p. 47)
- [WPZ⁺09] Andreas Wedel, Thomas Pock, Christopher Zach, Horst Bischof, and Daniel Cremers, *Statistical and geometrical approaches to visual motion analysis*, ch. An improved algorithm for TV-L1 optical flow, pp. 23–45, Springer-Verlag, 2009. (cited on p. 48)
- [WRM⁺08] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg, *Pose tracking from natural features on mobile phones*, ISMAR, 2008. (cited on p. 2, 7, 88)
- [ZMS09] Karel Zimmermann, Jiri Matas, and Tomas Svoboda, *Tracking by an optimal sequence of linear predictors*, PAMI **31** (2009), no. 4, 677–692. (cited on p. 12)