Robust Monocular Pose Estimation of Rigid 3D Objects in Real-Time

DISSERTATION

zur Erlangung des Grades
'Doktor der Naturwissenschaften'
am Fachbereich Physik, Mathematik und Informatik
der Johannes Gutenberg-Universität Mainz

JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

vorgelegt von

Henning Tjaden

geboren am 11. September 1986 in Gehrden

Mainz, 08.08.2018

Erstgutachter: [*Entfernt für die elektronische Version*]

Zweitgutachter: [*Entfernt für die elektronische Version*]

Drittgutachter: [*Entfernt für die elektronische Version*]

Datum der mündlichen Prüfung: 17.12.2018

Abstract

Being able to measure the spatial motion of arbitrary objects with high accuracy and low latency is vital for numerous higher level tasks in many fields of application. These include, but are not limited to: robotic perception, medical navigation and mixed reality systems. Such measurements are typically obtained by consecutively estimating the object's pose, i.e. its location and orientation in three-dimensional space, relative to a known frame of reference. The most successful are approaches based on optical sensors, such as digital cameras. But despite the large amount of literature and actively conducted research on this issue, fast, robust and accurate 3D object pose estimation still remains a key challenge in computer vision.

This dissertation presents novel approaches to visual 3D object pose estimation from 2D images. The particular feature of the proposed solutions is that they operate in real-time while only requiring a single (monocular) camera. The main parts of this work describe an innovative active infrared LED marker-based system as well as a novel algorithm for passive markerless pose estimation, both developed within the course of this thesis. For the marker-based approach, two original, nearly co-planar LED patterns are proposed. These enable high-speed, single-image pose estimation of multiple markers as well as robustly avoiding common pose ambiguities. The proposed markerless method presents a novel combination of region-based and direct photometric pose estimation. It is enabled by a new numerical pose optimization strategy derived for the region-based part as well as an innovative statistical object segmentation model. The overall approach thereby significantly improved the robustness towards challenging conditions, such as dynamic lighting, cluttered backgrounds, different object appearances, occlusions and fast and complex motion, compared to the state of the art. It is furthermore the first capable of estimating the poses of multiple arbitrarily textured objects in real-time on a commodity laptop. In addition to this, a new complex dataset dedicated to the task of monocular object pose tracking has been created and made publicly available.

Both proposed pose estimation solutions are extensively evaluated in numerous experiments, including the proposed as well as another popular public dataset. It is also shown that these solutions have been successfully applied in various practical scenarios, where they have enabled a variety of new problem solving opportunities.

Zusammenfassung

Die präzise und unmittelbare Vermessung der räumlichen Bewegung von Objekten (Trajektorien), ist eine essentielle Grundlage für die Lösung zahlreicher abstrakterer Probleme in diversen Anwendungsbereichen. Dazu gehören unter anderem: Robotik, medizinische Navigation und Mixed-Reality-Systeme. Derartige Trajektorien werden typischerweise durch fortlaufende Bestimmung der Pose des Objekts, d.h. seiner dreidimensionalen Position und Orientierung, relativ zu einem Bezugssystem berechnet. Am erfolgreichsten sind dabei optische Ansätze, die Sensoren, wie z.B. Digitalkameras, nutzen. Die schnelle, stabile und genaue Posenbestimmung von 3D-Objekten, bleibt jedoch trotz umfassender Literatur und aktiver Forschung zu diesem Thema, eine der größten Herausforderungen des maschinellen Sehens.

Diese Dissertation präsentiert neuartige Verfahren zur Posenbestimmung von 3D-Objekten aus 2D-Bildern. Das Besondere an diesen Ansätzen ist, dass sie in Echtzeit und mit nur einer einzigen (monokularen) Kamera funktionieren. Der Hauptteil beschreibt ein neues, aktives System, basierend auf Infrarot-LED-Marken, sowie einen neuartigen Algorithmus zur passiven markenlosen Posenbestimmung, die im Rahmen dieser Arbeit entwickelt wurden. Für den markenbasierten Ansatz werden zwei innovative LED-Muster vorgestellt. Diese ermöglichen Hochgeschwindigkeits-Posenbestimmung von mehreren Marken aus einem einzigen Bild sowie die zuverlässige Vermeidung von üblicherweise auftretenden Mehrdeutigkeiten. Die entwickelte markenlose Methode ist eine neuartige Kombination aus regionenbasierten und direkten, photometrischen Ansätzen. Diese wird ermöglicht durch eine neue numerische Posenoptimierung sowie ein innovatives, statistisches Segmentierungsmodell. Das Gesamtverfahren ist aktuell das stabilste gegenüber Bedingungen, wie z.B. dynamisches Licht, überladene Hintergründe, unterschiedliche Objektoberflächen, Verdeckungen und schnelle und komplexe Bewegungen. Es ist außerdem das erste Verfahren, das die Posen mehrerer beliebig texturierter Objekte auf einem Laptop in Echtzeit berechnet. Zusätzlich wurde ein neuer, komplexer Datensatz für monokulares Objektposen-Tracking erstellt und der Community zur Verfügung gestellt.

Beide Verfahren werden in zahlreichen Experimenten, inklusive des eigenen und eines weiteren populären Datensatzes, evaluiert. Weiterhin wird gezeigt, dass die Methoden bereits erfolgreich in verschiedenen realen Szenarien eingesetzt wurden.

Acknowledgements

[*Entfernt für die elektronische Version*]

Contents

1 Introduction

1.1 Measuring Spatial Motion

How can we accurately measure spatial motion? This dissertation aims to answer that question by presenting computer vision approaches involving novel algorithms for computing the absolute location and orientation of objects over time (the so-called *trajectory*) in three-dimensional space (Figure 1.1).

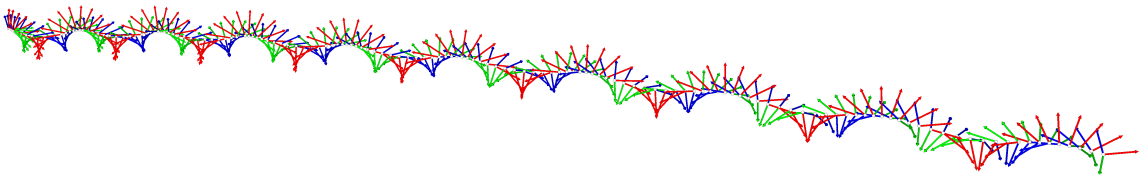
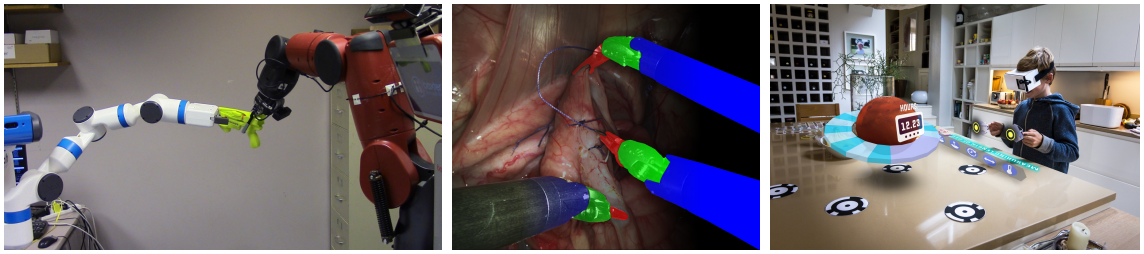


Figure 1.1: An illustration of a measured trajectory of a thrown frisbee. The spatial position and orientation at each point in time is depicted by coordinate axes. Here, as in the rest of the thesis, the colors indicate the three different principal axes, i.e. X (red), Y (green) and Z (blue) of the coordinate frame of the captured object.

Being able to determine exact trajectories of a moving object is a requirement for many higher level tasks in numerous areas of application. These include, but are not limited to, robotic perception [BMAK14, SHN+15, MKCK17], medical navigation [BASJ17, ZYCY17, AOH+18] and mixed reality systems [WLT16, MUS16,



(a) (b) (c)

Figure 1.2: Exemplary applications for real-time object pose estimation[1]. The first image shows an experimental setup from [PDG17], where a robot arm is trying to snatch an object from another robot arm which simultaneously tries evade these attempts (a). The second image was taken from of a dataset of the *Robotic Instrument Segmentation Sub-Challenge* which is part of the *Endoscopic Vision Challenge*[2]. It shows three instruments during robotic assisted minimally invasive surgery overlaid with the provided ground truth colored segmentation masks (b). The third image illustrates the capabilities of the Zappar *ZapBox*[3], a low-cost, versatile mixed reality kit for the *Google Cardboard*[4] based on passive markers (c).

MIS17] (see Figure 1.2). It is for example necessary to control the servo motors of a robot whenever it is supposed to grasp, catch or manipulate things, especially when they are moving simultaneously. It is also one of the core tasks to enable autonomous driving where navigation involves constantly estimating the locations of other vehicles or people. Medical navigation on the other hand is a prime example for applications where precise monitoring of spatial motion is needed. Here it can be used to capture the instruments of a surgeon in order to ensure precise operation and detect false positioning. In a similar way, this is currently also becoming more relevant in industrial assembly, where it is desired to autonomously supervise manufacturing steps, specifically the handling of components and tools and thereby reduce production faults. Finally, in the growing field of mixed reality systems and applications it can be used to augment individual objects with additional information or virtual modifications even when they are manipulated or moving independently from the rest of the scene (e.g. objects held in hand or a chair moved within a room).

[1]Thanks to the authors of [PDG17], the Robotic Instrument Segmentation Sub-Challenge committee and Zappar for kindly providing these images which remain entirely their property.

[2]The Endoscopic Vision Challenge: https://grand-challenge.org/site/endovis/

[3]The ZapBox mixed reality kit: https://www.zappar.com/zapbox/

[4]Google's Cardboard: https://vr.google.com/cardboard/

Here, such technology also allows to turn any captured object into a motion-based input device in order to interact with the system in a natural way.

All of the given examples are time-critical, meaning that they require the measurement result to be provided in real-time without delay along with the movement. Of course there are other scenarios in which runtime is not essential. In sports therapy for example, where captured motion sequences are usually reviewed and analyzed by the athlete or trainer retrospectively. But in order to cover the full range of practical application scenarios, it becomes clear that dedicated solutions must be able to operate on-line and in real-time.

Measuring or even describing spatial motion is an arbitrarily complex problem that depends on the object type of interest and the desired degree of abstraction. For many motion capturing problems it is suitable to describe objects as either rigid bodies, articulated or deformable soft-bodies. Rigid bodies are usually solid objects of which deformation is neglected, e.g. statues, buildings, planes or most 3D printed parts. Articulated objects such as trains, skeletons or scissors are usually modeled by multiple rigid objects each connected to at least one other by a joint. In contrast to this, living creatures, cloth or plants are deforming in a non-rigid manner with high complexity. These are therefore the hardest to capture and are typically modeled as soft-bodies. Nonetheless with regard to the previously mentioned desired degree of abstraction, an object can be modeled in different ways for recovering its spatial motion. A human being for example can either be idealized as a rigid body in form of a bounding box to only get a rough estimate of its spatial position and orientation. However, when it is required to measure posture changes, a person can also be represented with an articulated skeleton. By describing a human in form of a deformable soft-body, for example allows to recover even more details such as facial impressions or other deformation of the skin surface. This dissertation has a strong focus on real-time performance and measurement robustness. The rest of this thesis therefore focusses exclusively on handling rigid bodies in order to reduce the complexity with regard to the object type. Here, the so-called *pose* of the object changes only within six degrees-of freedom (*6DOF*), three for each the position and the orientation.

Over the last decades many approaches and sensors have been developed for captur-

ing the spatial motion of rigid bodies. The following will give a very brief summary of four such commonly used motion tracking technologies. For a more complete overview the reader is advised to refer to [WF02]. For measuring the orientation and acceleration of an object so-called inertial measurement units (IMUs) are widely used and are nowadays integrated in every smartphone and many other technical devices. These incorporate a gyroscope, an accelerometer and a compass. While IMUs provide an easy way to obtain robust relative rotation data they cannot be used to determine accurate absolute orientation, due to the accumulating measurement drift that is inherent to these kinds of sensors. This furthermore makes it generally hard to use them for relative position estimation over a longer period of time, as this involves integrating the already drift afflicted accelerometer data twice [NPM13].

The pose of a rigid object can also be estimated by using ultra sonic technology. Here at least three ultrasound sensors (i.e. microphones) and one emitter are needed in order to obtain the absolute position of that source. Based on this, absolute rotation can be obtained by using a minimum of three such emitters in a rigid compound. However, these must be processed sequentially with regard to the speed of sound, which limits the maximum measurement frequency and makes filtering necessary. Since it is based on sound, the technology is furthermore relatively limited in range and prone to interfering noise, which restricts it from being used in many environments.

Another alternative are magnetic systems that measure the local magnetic field vector at the sensor. Here, using e.g. three orthogonally oriented magnetometers in a rigid sensor unit allows to obtain the absolute orientation with respect to the excitation. In this context, it is popular to actively induce excitations by e.g. using a multicoil source unit. By energizing three of its coils in sequence and measuring the magnetic field vector at each magnetometer, the position of the sensor unit device can also be measured with respect to the source unit. The main downside of this technology is however, that the magnetic field is affected by ferromagnetic and conductive material in the environment. Distorting objects can potentially act as unwanted source units and should therefore be avoided in the working volume or must be modeled appropriately. Such systems are furthermore limited in range of operation by the inverse cubic falloff of the magnetic fields with respect to the distance from the source, which has an according negative impact on the measurement

accuracy.

With regard to physical constraints, optical methods are essentially only limited by the speed of light when it comes to transmission time and working volume. In the simplest case, when only using a single camera the absolute pose of a rigid object can be estimated from only four 3D points and their corresponding 2D projections into the image plane, as explained in detail in Section 2.2. Here the main problem are visual occlusions, meaning that something blocks the line of sight between the object of interest and the optical sensor, which must be handled appropriately. But despite this constraint, the advantages predominate, which is why optical tracking systems are probably the most widely employed for the task of pose estimation (e.g. robotics, automotive, medical navigation and mixed reality systems) nowadays. Optical systems can be divided into active and passive systems. Passive systems use just the natural light that is currently present in the scene to detect either high contrast artificial so-called *fiducials* or natural structures. In contrast, active systems use additional light sources to facilitate the detection of specially designed fiducials. For this, they usually work with infrared light to minimize the influence of ambient lighting. Optical tracking systems often determine the three-dimensional position of a ball-like fiducial based on two or more camera images and triangulation in so-called *stereo setups*. However, here all cameras have to be calibrated and registered in a rigid setup in advance, which is a complex and time-consuming process and requires a lot of expertise. While motion capturing can then be done very efficiently utilizing epipolar constraints, it requires the fiducials always to be seen by (at least) two cameras, and all cameras to remain static at runtime. In some applications, this may not always be guaranteed and a system based on the image of just a monocular (meaning single) camera is needed. Hence, this work focusses exclusively on pose estimation with a single camera.

1.2 Monocular Pose Estimation

Estimating the pose of an object with only a single monocular camera is the most generic and elegant optical approach. Especially in the constantly growing fields of mixed reality and robotics, object pose estimation is typically only one of many com-

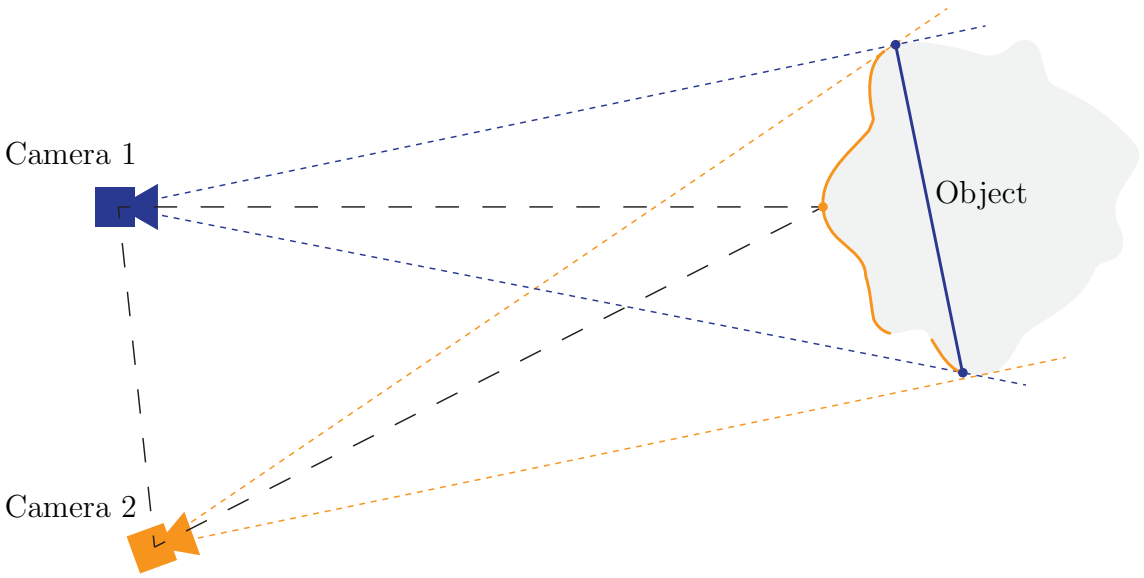


Figure 1.3: An illustration of the different modalities available for pose estimation using a single monocular camera compared to a two camera stereo setup. A single camera (depicted in blue) essentially only sees the contour (blue dots) and a flat projection of everything within the silhouette (blue line) of the object. By including a second camera (depicted in orange) all parts of the object's surface that are visible to both cameras (orange contour) can potentially be spatially reconstructed using triangulation (dashed black lines), providing a depth modality in addition to the 2D images.

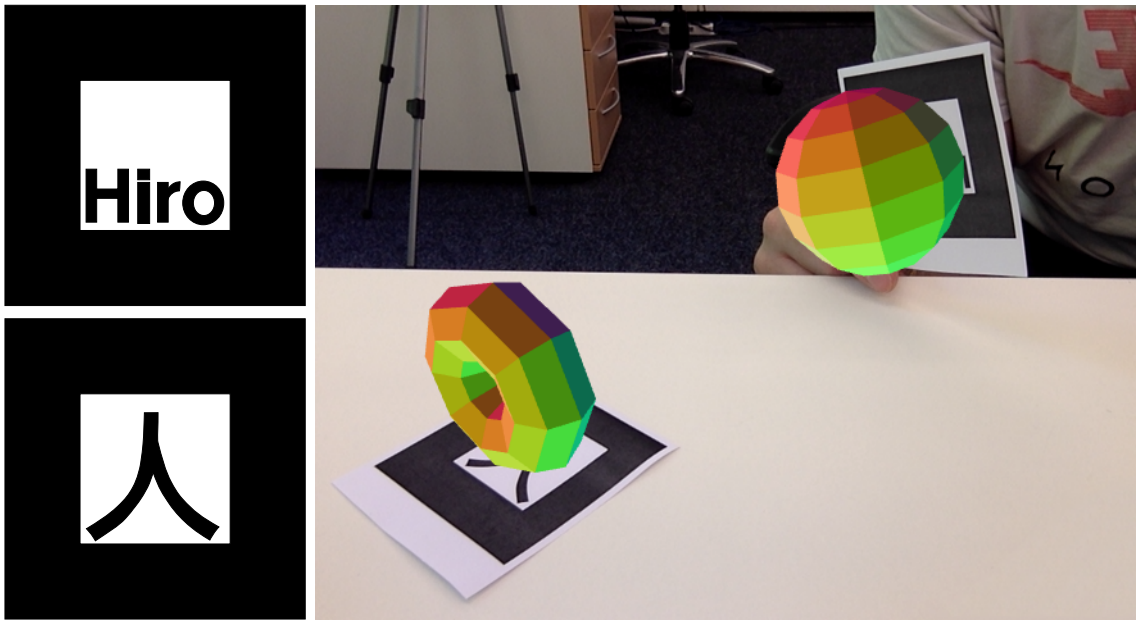
plex tasks that must all be computed simultaneously in real-time on often battery powered hardware. Therefore low runtime and power consumption of dedicated solutions are crucial aspects for them to be practical in such scenarios. In particular the latter can be achieved by using as few sensors as possible e.g. only a single camera. Also, as for most other computer vision problems, such monocular approaches are usually the most convenient compared to e.g. multi-camera stereo systems because they keep calibration requirements to a minimum and suffer least from visibility issues. Once the camera is calibrated, there is no constraint to remain in place, meaning that both the object and the camera can be moving freely at runtime. In the context of robotics this is for example attractive in grasping scenarios when the camera is attached to an arm of a robot. It also becomes relevant in mixed-reality reality systems, e.g. when a hand-held object shall be augmented with virtual content

in a head mounted display or on the screen of a mobile device.

When using more than one camera, triangulation can be used in order to compute spatial surface reconstructions of the scene at runtime [MSKS05]. This additional modality can be used along with their 2D images to simplify the task of pose estimation (see Figure 1.3). This is not possible in case of a single camera, meaning that the object's pose must be computed from only 2D images. Thus monocular approaches are always based on minimizing the difference between the true 2D projection of the object in the current image and an estimated synthetic 2D projection of a 3D model abstraction of that object, that is parametrized by the sought pose. Given such a 3D model, it is possible to render arbitrary artificial 2D projections by freely choosing the pose. Whenever the object is in the field of view of the camera, the image shows a real projection of it under its true pose relative to the camera. If the estimated pose equals this actual pose, the difference between the synthetic and the true projection is assumed to be minimal. Due to this strategy, monocular pose estimation is usually solved in two coarse steps. First the object is segmented from the background in the current image. This segmentation is then used in the second step for the actual pose estimation, by aligning the model with it, as explained above.

The problem of pose estimation can further be separated into *pose tracking* (also known as *recursive pose estimation*) and *pose detection*. In case of tracking, the object is assumed to be seen in a sequence of consecutive images such as video footage. Here, the motion of the object is assumed to be relatively small between two consecutive frames. Here, only the pose difference from one frame to the next has to be determined, which is why tracking typically can be performed more efficiently than detection. The main downside of pure pose tracking algorithms is the need for manual initialization at the beginning and re-initialization after tracking loss to get a coarse pose estimate to start from.

This leads to the problem of pose detection, where the object's pose has to be estimated from a single image without any prior pose knowledge. This lack of information makes pose detection generally more complex and computationally demanding compared to pose tracking. To obtain generic, robust real-time pose estimation solutions, tracking thus has to be combined with reliable detection, which provides a



(a) (b)

Figure 1.4: An illustration of the functionality of the ARToolkit. Two different AR-markers where used in this example (a). The live pose estimation result allows to render virtual objects (here a torus and a sphere) overlaid to the camera image with respect to the position and orientation of each marker (b).

starting solution whenever tracking is lost, e.g. in cases of strong occlusion, rapid movement or when the object leaves the camera's field of view.

Another popular approach to resolve the pose detection problems is to attach artificial fiducials or markers to the objects of interest. Their visual appearance is specifically designed in ways that they can easily and quickly be located in images and thereby simplify the task of pose detection. In many cases they thus enable *tracking by detection*, meaning that their absolute pose can be computed in each single image in real-time from scratch without exploiting temporal continuity assumptions. Over the years, numerous different both active [NF05, SSBJ06, YZY10, HHC$^+$11a, FMSS14] and passive [KB99, Fia05, AHH10, Ols11, BART11, RSI12, HSZ$^+$13] markers have been developed.

One of the earliest and still most popular works on real-time monocular marker-

(a) (b) (c)

Figure 1.5: Failure cases of the ARToolkit. In case of motion blur the edges of the marker smear and cannot be detected anymore (a). The marker is required to be fully visible in each image such that even small occlusions by fingers cause the detection to fail (b). In its original form the contrast-based segmentation strategy is furthermore prone to inhomogeneous or poor lighting (c).

based pose estimation was presented in [KB99] within the context of augmented reality (AR). They introduced passive planar square fiducials with a printed black and white pattern, so-called AR-markers (see Figure 1.4). Back then the system impressively demonstrated the vast potential for this kind of technology and sparked a lot of interest in the field of research. The well-kown related software library called the ARToolkit[5] has ever since been improved and advanced in numerous successive approaches. Even nowadays it is still widely used in research as a prototypical solution to pose estimation.

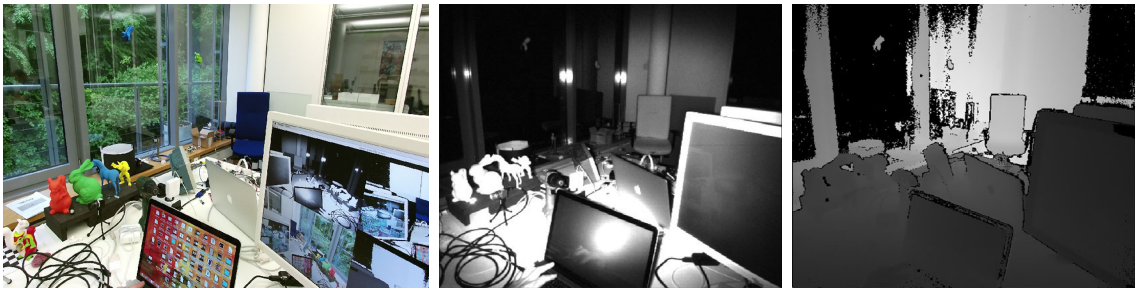
This ongoing popularity can probably be explained by the easy accessibility of the overall approach. For one, it only requires a single camera on the hardware side and even works with a standard webcam. The passive markers can easily be reproduced by printing or even drawing them on paper and require no additional lighting, which makes it very low-cost. Also even when executed on weak hardware (e.g. a mobile phone) and using multiple markers simultaneously their poses can be estimated in real-time. But despite their easy application AR-markers come with significant disadvantages. Their detection is based on high contrast edges in the image as well as overall high intensity contrast between the black and the white areas of the marker. The first is prone to motion blur caused by fast marker or camera movement which render the edges to appear unsharp or slurred in the image (Figure 1.5). The latter

[5]The ARToolkit: https://artoolkit.org/

is often violated by poor or inhomogeneous lighting which causes the segmentation to fail. They are furthermore not robust to occlusions, meaning that not even small parts are allowed to be covered in order for the pose estimation to work. While some of these problems can be overcome by the use of active lighting (see Section 3.2) or more recent and advanced passive marker approaches [SSS+17], another constraint that is inherent to all planar markers is that they can only be detected when their front side is visible which limits the viewing angles to a maximum of 180° in all directions. Also related to the planarity are so-called *pose ambiguities* that cause inaccuracies from certain perspectives as explained in Section 3.2.5. Finally and probably most importantly, although fiducials often enable highly robust and accurate pose estimation, having to attach any kind of marker to the objects at all is undesirable or even impossible in many application scenarios. For example in case of sports therapy, where ergonomics might be affected or for tracking objects in public environments such as cars or planes that cannot be modified beforehand.

Thus, over the last couple of decades a large variety of markerless monocular pose estimation methods have been proposed that differ in terms of image representation and the corresponding model abstraction of the objects. In this work, these approaches are divided into three coarse categories: firstly, point-based methods using so-called *natural image features* (see Chapter 3, Section 3.3), secondly, region-based methods using the object's silhouette (see Chapter 4) and thirdly, direct photometric methods that minimize the pixels' intensity differences between two views (see Chapter 5). Especially for markerless approaches, the biggest challenges arise from diverse object surface properties (e.g. reflectance, transparency), partial or total occlusions of the object, varying or poor lighting conditions and cluttered or dynamic background that make object segmentation an almost arbitrarily complex problem in general. An extensive survey of different strategies to handle these conditions is presented in [LF05] and Chapters 3, 4 and 5 additionally contain particular overviews of more recent related work.

For the sake of completeness it should also be mentioned, that starting in 2010 with the release of Microsofts *Kinect* so-called depth sensors emerged in the consumer market. Their availability simultaneously lead to a large interest in computer vision research involving this kind of technology, and also in the context of object pose estimation. For many scenarios these sensors are a suitable alternative



(a) (b) (c)

Figure 1.6: A visualization of the sensor data of an RGB-D camera (here a Microsoft Kinect V2 that uses time-of-flight technology). The sensor simultaneously provides images from a color camera (a), a monochrome camera extended with an infrared pass filter (b) and distance (or depth) estimate for each pixel computed from the active lighting (c). Here the depth of a pixel is represented by its grey value, where higher brightness means larger distance. Note that the sensor is unable to measure depth for the scene outside the windows (indicated by black pixels), due to the range limitation.

to purely monocular systems, especially since they are manufactured in form of a single portable device that can be held in hand which enables them to be used in dynamic setups as well. Along with live color images these sensors potentially provide simultaneous real-time depth measurements for each pixel (see Figure 1.6). Thus, they are often referred to as RGB-D (red, green blue, depth) sensors in literature. RGB-D measurements are achieved by combining the color camera with a monochrome camera and an infrared emitter in a rigid compound. Here infrared light is actively projected onto the observed scene using either a structured light pattern or time-of-flight technology [SH16].

Like in stereo camera systems the additional dense (meaning per pixel) depth modality input simplifies the image processing complexity in general and often leads to overall more robust results. Therefore in recent years, research on optical pose estimation has mainly focused on RGB-D sensors for both tracking [CC13, RPMR13, RPK+14, KMB+14, KTIN17, RPK+17, TNT17] and detection [HHC+11b, HLI+12, BKM+14, KMT+16]. Although these methods usually outperform those only based on monocular RGB image data, they are limited in distance to the sensor and struggle with sunlight, due to the physical constraints caused by the active infrared

lighting. They could for example not be used in order to capture a plane in the sky and can in general not be used in outdoor environments, due to their dependence on the weather. Thus, RGB-D sensors and methods relying on them are not included in this work which focusses exclusively on pure monocular solutions to keep the prior limitations determined by the employed hardware to a minimum.

1.3 Selected Publications

The real-time methods developed within the course of this dissertation contribute to the progression of monocular object pose estimation. We started out by developing a novel system based on active infrared markers. By equipping the camera with an infrared filter the image segmentation complexity was kept to a minimum, which allowed to focus mainly on improving the robustness of the pose estimation. Here we designed a novel nearly co-planar marker pattern in order to resolve pose ambiguities which are inherent to many other marker-based methods. The complete system enables tracking by detection of multiple active markers simultaneously at high frame rates (a single marker can be tracked within $\sim$1 ms). To our knowledge it is still superior in terms of runtime and precision to other purely monocular systems: *High-Speed and Robust Monocular Tracking* [TSSS15].

Based on the current state of the art for real-time markerless pose estimation, we then developed an improved region-based algorithm. In contrast to the marker-based approach, it is capable of tracking the poses of multiple objects without modifications using prior shape knowledge. While at first adapting an existing image segmentation approach, here the main focus was again on improving the robustness and accuracy of the pose optimization as well as reducing the runtime. The resulting so-called *Gauß-Newton-like* optimization strategy enabled region-based pose tracking of multiple objects in real-time and handling even strong mutual occlusions: *Real-Time Monocular Segmentation and Pose Tracking of Multiple Objects* [TSS16].

In succeeding work we extended the system by replacing the previous segmentation strategy by a more sophisticated approach based on temporally consistent, local color histograms. It improved tracking robustness in case of heterogenous objects,

cluttered backgrounds and arbitrary occlusions significantly. This furthermore lead to a novel object descriptor which enabled region-based pose detection that can be used in conjunction with tracking to recover from temporary tracking losses. This resulted in the first complete real-time, region-based object pose estimation system: *Real-Time Monocular Pose Estimation of 3D Objects using Temporally Consistent Local Color Histograms* [TSS17].

Based on the advances presented in the two conference papers [TSS16, TSS17], we proposed a further improved version of our region-based pose tracking approach in a more detailed article, that was submitted to a journal. This work was done in cooperation with Daniel Cremers from the Technical University of Munich. Together we developed a novel systematic derivation of a true Gauß-Newton scheme. This was previously missing in [TSS16] for the *Gauß-Newton-like* method, which was originally discovered by empirical studies. In this work, we further introduced a newly created dataset dedicated to the task of passive monocular object pose estimation and made it publicly available to the community. We also used it in this article to demonstrate the improvements achieved by the derived Gauß-Newton optimization: *A Gauss-Newton Approach to Real-Time Monocular Multiple Object Tracking* [TSSC18].

1.4 Thesis Outline

Following this introduction, Chapter 2 presents an overview of all basic mathematical and algorithmic concepts and introduces the corresponding notation used throughout the rest of the thesis. It starts by showing how to model and parametrize the spatial motion of a rigid body. It goes to by explain the fundamental principle of projective geometry. This includes the introduction of a mathematical model of a real camera and the calibration of the corresponding parameters. Based on this, it is shown how synthetic images can be rendered with respect to the intrinsic properties of a real camera. The chapter concludes with explaining the mathematical concept of parametric numerical optimization forming the basis for pose tracking and refinement.

Next are three main chapters on different object pose estimation approaches including respective related work. Chapter 3 deals with pose estimation methods based on so-called *2D-to-3D point correspondences*. It gives a detailed description and evaluation of the active marker system developed in [TSSS15]. This includes image segmentation, correspondence finding, pose estimation and a strategy to semi-automatically reconstruct the 3D model of the markers. In addition to this, an overview of related passive methods based on so-called *natural features* is provided.

Chapter 4 is about so-called *region-based* pose estimation that uses a silhouette and contour abstraction of the object in the images. It starts by describing the general concept of region-based pose estimation, introduces a corresponding cost function and then presents the system proposed in [TSS16, TSS17, TSSC18]. Here the main parts cover different approaches to statistical object segmentation, robust level-set-based pose optimization and a strategy for combined region-based pose detection and tracking based on the proposed object descriptor. The chapter concludes with an extensive experimental evaluation of the developed approach within the dataset introduced in [TSSC18] as well as another popular dataset.

Chapter 5 then describes how modern so-called *direct photometric* methods can be applied to object pose tracking. It first explains the concept of direct image alignment and introduces a corresponding photometric cost function. Subsequently, it is shown how this can be optimized with respect to the object's pose parameters. Based on this, a novel combined region-based and direct photometric cost function for object pose estimation is introduced. Enabled by the proposed region-based strategy, it is derived how this combined cost function can be efficiently optimized in a joint fashion. Another experimental evaluation in the proposed dataset eventually demonstrates how this combined approach leads to further significant improvement compared to the previously presented purely region-based solution.

Finally, the thesis concludes with Chapter 6. It contains a summary of the achieved contributions, current and potential practical applications of the developed pose estimation systems and future research directions based on this work.

2

Background Theory

This chapter provides an overview of notation and basic concepts used throughout this thesis. It starts by introducing the two fundamental sets of transformations that are involved when estimating the pose of a moving 3D object from 2D camera images based on [MSKS05]. The first being *rigid body motion* also known as *Euclidian motion* that models how an object moves (Section 2.1) and the second being *projective geometry* that describes how a 2D camera image of a 3D object is formed (Section 2.2). The latter also includes geometric camera calibration and it is explained how to use the derived camera model in order to render synthetic views of an object model as if they were captured by a real camera. The chapter concludes with an overview of selected numerical optimization methods (Section 2.3), that are typically used in the context of monocular pose estimation on the basis of the previously introduced concepts.

2.1 Rigid Body Motion

When a rigid object moves in front of a camera its pose changes within six degrees of freedom (6DOF), namely three for rotation about the X-, Y- and Z-axis and three for translation along these axes. A pose can thus be seen as a so-called *rigid*

body transform from one Cartesian coordinate frame of reference to another, that preserves distances as well as orientations. Such a displacement can be denoted by

$$g : \mathbb{R}^3 \to \mathbb{R}^3; \quad g(\mathbf{X}) \doteq R\mathbf{X} + \mathbf{t}, \tag{2.1}$$

with

$$\mathbf{X} \doteq \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \in \mathbb{R}^3, \quad R \doteq \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \in \mathbb{SO}(3) \quad \text{and} \quad \mathbf{t} \doteq \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \in \mathbb{R}^3,$$

where R applies the rotational part of the transformation to the coordinates of a 3D point $\mathbf{X}$ and $\mathbf{t}$ the translation. Here the rotation R is represented as a 3×3 matrix being an element of the special orthogonal group

$$\mathbb{SO}(3) \doteq \left\{ R \in \mathbb{R}^{3\times 3} \;\middle|\; R^\top R = RR^\top = \mathbf{I}_{3\times 3},\, \det(R) = +1 \right\},$$

that contains all three-dimensional orientation-preserving linear transformations, with $\mathbf{I}_{3\times 3}$ being the 3×3 identity matrix.

Let a vector between two arbitrary points $\mathbf{X}$ and $\mathbf{X}'$ be defined as

$$\mathbf{a} \doteq \mathbf{X} - \mathbf{X}' = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \in \mathbb{R}^3,$$

and the Euclidian distance between any two points be given by the L_2-norm $\|\mathbf{a}\|_2$ of their corresponding vector, g preserves distances if

$$\|\mathbf{a}\|_2 = \|\mathbf{X} - \mathbf{X}'\|_2 = \|g(\mathbf{X}) - g(\mathbf{X}')\|_2, \quad \forall \mathbf{a} \in \mathbb{R}^3, \tag{2.2}$$

and orientations when

$$g(\mathbf{a} \times \mathbf{a}') = g(\mathbf{a}) \times g(\mathbf{a}'), \quad \forall \mathbf{a}, \mathbf{a}' \in \mathbb{R}^3, \tag{2.3}$$

meaning that the cross product of any two vectors $\mathbf{a}$ and $\mathbf{a}'$ is preserved. Given the

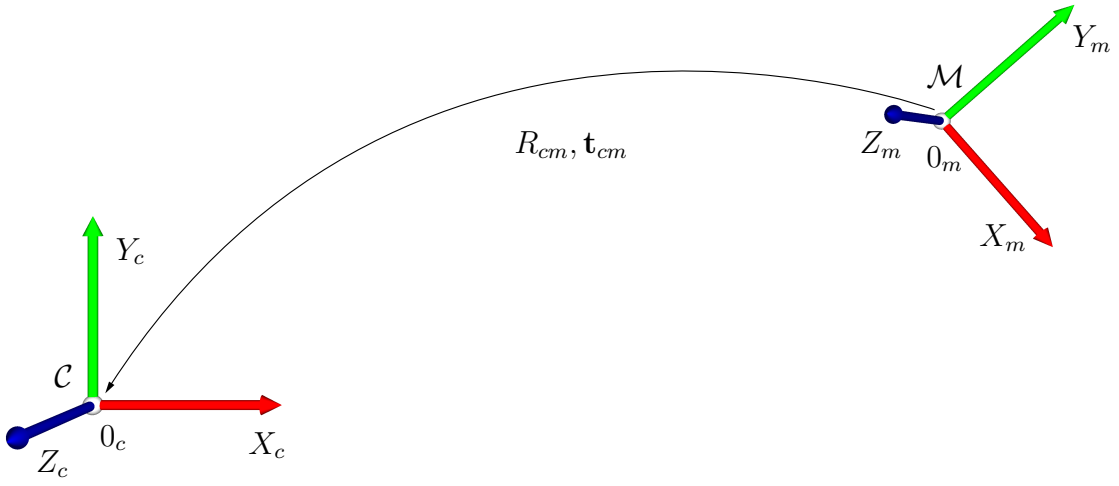


Figure 2.1: A rigid body transform g_{cm} from the model frame $\mathcal{M}$ to the camera frame $\mathcal{C}$ defined by the rotation matrix R_{cm} and the translation vector $\mathbf{t}_{cm}$.

group properties of the $\mathbb{SO}(3)$, it is immediate to show that (2.1) satisfies both (2.2) and (2.3), and therefore performs a rigid body transformation.

In case of monocular pose estimation this displacement typically describes the relative position and orientation of a model coordinate frame $\mathcal{M}$ to the camera coordinate frame $\mathcal{C}$

$$\mathbf{X}_m = [X_m, Y_m, Z_m]^\top \mapsto g_{cm}(\mathbf{X}_m) = \mathbf{X}_c = [X_c, Y_c, Z_c]^\top,$$

where the subscript indicates their affiliation to the respective frame of reference (see Figure 2.1). Note that using this notation the subscript of g_{cm} furthermore indicates that transformation is performed in direction from model to camera which is written as

$$\mathbf{X}_c = g_{cm}(\mathbf{X}_m) \doteq R_{cm}\mathbf{X}_m + \mathbf{t}_{cm}, \tag{2.4}$$

where $\mathbf{t}_{cm}$ is now explicitly the translation of the origin of the coordinate frame $\mathcal{M}$ relative to that of $\mathcal{C}$ and R_{cm} describes the relative orientation from $\mathcal{M}$ to $\mathcal{C}$. Inverting the direction of the transformation such that a point $\mathbf{X}_c$ given in the camera is mapped to its coordinates in the model frame as

$$g_{cm}^{-1}(\mathbf{X}_c) = R_{cm}^\top(\mathbf{X}_c - \mathbf{t}_{cm}) = R_{cm}^\top\mathbf{X}_c - R_{cm}^\top\mathbf{t}_{cm} \doteq g_{mc}(\mathbf{X}_c), \tag{2.5}$$

leads to

$$\mathbf{X}_m = g_{mc}(\mathbf{X}_c) \doteq R_{mc}\mathbf{X}_c + \mathbf{t}_{mc},$$

with $R_{mc} = R_{cm}^\top$ and $\mathbf{t}_{mc} = -R_{cm}^\top \mathbf{t}_{cm}$. Since any rigid body transform always describes the relative relationship between two coordinate frames, it becomes clear that knowing the pose of an object relative to the camera one can directly derive the pose of the camera relative to the object from it, regardless of which is moving. It is thus an arbitrary choice of design which direction is assumed to be estimated directly. In this work the term *object pose* always refers to the rigid body transform g_{cm} of the model $\mathcal{M}$ into the camera $\mathcal{C}$, i.e. the object's pose relative to the camera.

In contrast to the pure rotational case, the full rigid body transformation in its current form is not linear but affine, as can be seen in (2.4). Such an affine transformation may be converted to a linear one by using *homogeneous coordinates*. Simply appending a "1" to the coordinates of any point $\mathbf{X} \in \mathbb{R}^3$ yields a corresponding point

$$\tilde{\mathbf{X}} \doteq \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \in \mathbb{R}^4, \tag{2.6}$$

embedded in a hyperplane in $\mathbb{R}^4$. Using this notation (2.4) can now be rewritten in linear form as

$$\tilde{\mathbf{X}}_c = \tilde{g}_{cm}(\tilde{\mathbf{X}}_m) = T_{cm}\tilde{\mathbf{X}}_m = \begin{bmatrix} R_{cm} & \mathbf{t}_{cm} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X}_m \\ 1 \end{bmatrix} = \begin{bmatrix} R_{cm}\mathbf{X}_m + \mathbf{t}_{cm} \\ 1 \end{bmatrix},$$

where $\tilde{g}_{cm}$ is the homogenous representation of the rigid body transform g_{cm} applied by the homogenous matrix T_{cm}. Transformation in the opposite direction can now simply be performed by matrix inversion as

$$\tilde{\mathbf{X}}_m = \tilde{g}_{mc}(\tilde{\mathbf{X}}_c) = T_{cm}^{-1}\tilde{\mathbf{X}}_c = T_{mc}\tilde{\mathbf{X}}_c,$$

since

$$T_{cm}^{-1} = T_{mc} = \begin{bmatrix} R_{cm} & \mathbf{t}_{cm} \\ \mathbf{0}^\top & 1 \end{bmatrix}^{-1} = \begin{bmatrix} R_{cm}^\top & -R_{cm}^\top \mathbf{t}_{cm} \\ \mathbf{0}^\top & 1 \end{bmatrix} = \begin{bmatrix} R_{mc} & \mathbf{t}_{mc} \\ \mathbf{0}^\top & 1 \end{bmatrix},$$

which applies the same inverse transformation as previously shown for the affine representation in (2.5).

Hence, from now on any rigid body transform will be utilized in form of its homogeneous 4×4 matrix representation

$$T \doteq \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{SE}(3), \tag{2.7}$$

being an element of the special Euclidean Group

$$\mathbb{SE}(3) \doteq \left\{ T = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \;\middle|\; R \in \mathbb{SO}(3),\, \mathbf{t} \in \mathbb{R}^3 \right\},$$

that describes the space of all possible rigid body transforms i.e. 6DOF poses in $\mathbb{R}^3$. This further enables combining rigid body transformations between multiple coordinate frames in form of a simple matrix multiplication, e.g. $T_{da} = T_{dc}T_{cb}T_{ba}$.

So far this section has only considered static *rigid body transforms* at a single point in time from one coordinate frame to another. In cases where either the object, the camera or both are moving this relative transform changes over time t along a continuous path. It thereby describes a trajectory $T_{cm}(t) : \mathbb{R} \mapsto \mathbb{SE}(3)$, that can be modeled by *rigid body motion*. Within monocular pose estimation, the pose along the path can potentially be determined at any time t_k that coincides with a video frame captured by the camera. Here, a sequence of images is denoted by $I(t_k)$, with $k = 0, \ldots, l$ being the frame index and $I(t_l)$ being the current live camera image. Hence, a trajectory is represented as a time-discrete sequence of rigid body transforms

$$T_{cm}(t_k) \in \mathbb{SE}(3), \quad t_k \in \mathbb{R}, \quad k = 0, \ldots, l,$$

each pose corresponding to a frame (here meaning image) with index k (see Figure 2.2). With $\mathbf{X}_m(t_0)$ being the model coordinates in its initial configuration, its coordinates with respect to the camera in a frame k are then given by $\tilde{\mathbf{X}}_c(t_k) = T_{cm}(t_k)\tilde{\mathbf{X}}_m(t_0)$. Given such a trajectory the pose change that occurred between two

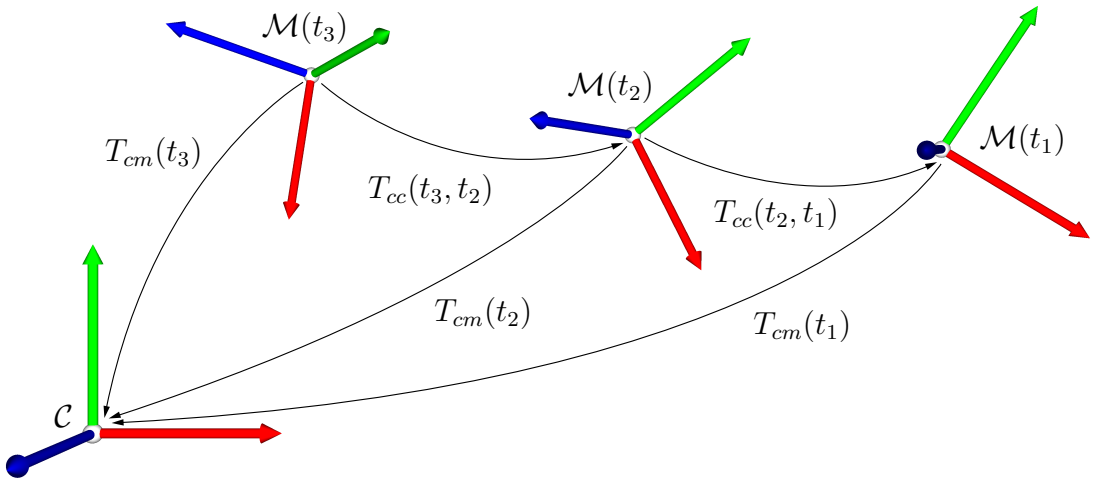


Figure 2.2: Rigid body motion over time. The trajectory is visualized at the three consecutive points in time t_1, t_2 and t_3 that could correspond to camera frames. Both the inter-frame pose changes $T_{cc}(t_k, t_{k-1})$ as well as the individual complete rigid body transforms $T_{cm}(t_k)$ are illustrated in relation to the camera $\mathcal{C}$.

consecutive frames k and $k-1$ is described by

$$T_{cc}(t_k, t_{k-1}) \doteq T_{cm}(t_k) T_{cm}(t_{k-1})^{-1} = T_{cm}(t_k) T_{mc}(t_{k-1}), \quad \forall k = 1, \ldots, l,$$

which recursively maps model coordinates $\tilde{\mathbf{X}}_c(t_{k-1}) = T_{cm}(t_{k-1}) \tilde{\mathbf{X}}_m(t_0)$ given with respect to a frame $k-1$ to those in the next as

$$\tilde{\mathbf{X}}_c(t_k) = T_{cc}(t_k, t_{k-1}) \tilde{\mathbf{X}}_c(t_{k-1}), \quad \forall k \in 1, \ldots, l,$$

with $T_{cc}(t_1, t_0) = T_{cm}(t_1)$ since $T_{cm}(t_0) = \mathbf{I}_{4\times4}$. Given this notation, the object's pose in the current live frame $T_{cm}(t_l)$ can now be written in form of a composition of consecutive rigid body transforms achieved by simple matrix concatenation as

$$T_{cm}(t_l) = T_{cc}(t_l, t_{l-1}) \ldots T_{cc}(t_3, t_2) T_{cc}(t_2, t_1) T_{cm}(t_1),$$

describing the entire trajectory up to the current frame.

With regard to the introduction chapter, this means that in case of pose detection each $T_{cm}(t_k)$ is estimated directly from scratch without knowing the pose in the

preceding frame $T_{cm}(t_{k-1})$, whereas for tracking only the inter-frame pose change from the previous to the current frame $T_{cc}(t_l, t_{l-1})$ has to be determined. Thus, tracking is often also referred to as recursive pose estimation.

2.1.1 Parametrization of Rigid Body Motion

Although homogeneous 4×4 matrices $T \in \mathbb{SE}(3)$ are a convenient pose representation for applying, combining and inverting rigid body transforms, they are not an ideal parametrization of the 6DOF that they describe. Each $T \in \mathbb{SE}(3)$ has sixteen entries (2.7) of which only twelve contain the actual pose information. The nine entries of $R \in \mathbb{SO}(3)$ describe the 3DOF of the rotation and the three entries of $\mathbf{t} \in \mathbb{R}^3$ the other 3DOF of the translation. While $\mathbf{t}$ is already a minimal representation for translation, R is not for rotation. Its nine entries are not free and must satisfy properties of the $\mathbb{SO}(3)$ which impose six independent constraints on them. Therefore, the space of three-dimensional rotation should ideally also be represented by three parameters as the other six are redundant. Such a minimal representation for rotation can be derived with help of an alternative notation of the cross product, as shown in the following.

The cross product of two vectors $\mathbf{a}$, $\mathbf{b} \in \mathbb{R}^3$ is commonly denoted by $\mathbf{a} \times \mathbf{b} \in \mathbb{R}^3$. This definition of the cross product is linear in both its arguments and can thus be represented by a linear map from $\mathbb{R}^3$ to $\mathbb{R}^3$ in form of a skew-symmetric matrix denoted by

$$\hat{\mathbf{a}} \doteq \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 3},$$

that allows for the cross product to be expressed by $\hat{\mathbf{a}}\mathbf{b} = \mathbf{a} \times \mathbf{b}$. Therefore, the cross product defines an isomorphic (one-to-one) map between any vector $\mathbf{a}$ and a corresponding 3×3 skew-symmetric matrix $\hat{\mathbf{a}}$ for which it holds

$$\hat{\mathbf{a}} = -\hat{\mathbf{a}}^\top. \tag{2.8}$$

Note that a matrix $A \in \mathbb{R}^{3\times 3}$ is skew-symmetric if and only if $A = \hat{\mathbf{a}}$ for any vector $\mathbf{a} \in \mathbb{R}^3$.

Given this, next a trajectory of pure rotational motion $R(t) : \mathbb{R} \mapsto \mathbb{SO}(3)$ is considered. Here, the constraint

$$R(t)R(t)^\top = \mathbf{I}_{3\times 3} \tag{2.9}$$

must always be satisfied. Computing the derivative of (2.9) with respect to time t by applying the product rule yields

$$\frac{dR(t)}{dt}R(t)^\top + R(t)\frac{dR(t)^\top}{dt} = 0$$

and therefore

$$\frac{dR(t)}{dt}R(t)^\top = -\left(\frac{dR(t)}{dt}R(t)^\top\right)^\top, \tag{2.10}$$

which shows that $\frac{dR(t)}{dt}R(t)^\top$ must be a 3×3 skew-symmetric matrix, according to (2.8). This then further implies that there must exist a vector $\mathbf{w}(t) = [\omega_1(t), \omega_2(t), \omega_3(t)]^\top \in \mathbb{R}^3$, such that

$$\frac{dR(t)}{dt}R(t)^\top = \hat{\mathbf{w}}(t).$$

Right hand multiplication of both sides with $R(t)$ yields

$$\frac{dR(t)}{dt} = \hat{\mathbf{w}}(t)R(t), \tag{2.11}$$

and thus

$$\frac{dR(t_0)}{dt} = \hat{\mathbf{w}}(t_0),$$

assuming $R(t_0) = \mathbf{I}_{3\times 3}$. Hence a skew-symmetric matrix gives a first-order approximation to a rotation matrix around $\mathbf{I}_{3\times 3}$ as

$$R(t_0 + dt) \approx \mathbf{I}_{3\times 3} + \hat{\mathbf{w}}(t_0)dt.$$

The space of all skew-symmetric matrices denoted by

$$\mathfrak{so}(3) \doteq \left\{ \hat{\mathbf{w}} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 3} \;\middle|\; \mathbf{w} \in \mathbb{R}^3 \right\}, \tag{2.12}$$

thus also describes the tangent space at the identity of the rotation group $\mathbb{SO}(3)$. Note that the $\mathbb{SO}(3)$ is a Lie group and thus $\mathfrak{so}(3)$ is called its Lie algebra. More details on the structure of Lie groups can be found in [MLS94]. For any $R(t) \neq \mathbf{I}_{3\times 3}$, the tangent space at $R(t)$ is simply $\mathfrak{so}(3)$ transported to $R(t)$ by a multiplication by $R(t)$ on the right, as indicated by (2.11). This further means that any rotation matrix $R \in \mathbb{SO}(3)$ locally only depends on three parameters $(\omega_1, \omega_2, \omega_3)$. This local approximation can eventually be used to obtain the desired representation of the rotation group, as explained in the following.

In order to show that in fact every rotation matrix in $\mathbb{SO}(3)$ can be represented by a vector in $\mathbb{R}^3$, it is first assumed that $\hat{\mathbf{w}}(t)$ is constant i.e. $\hat{\mathbf{w}}(t) = \hat{\mathbf{w}}$ in (2.11), which yields

$$\frac{dR(t)}{dt} = \hat{\mathbf{w}}R(t). \tag{2.13}$$

Since the product of $R(t)$ with any vertex $\mathbf{X} \in \mathbb{R}^3$ at an initial time t_0 gives $\mathbf{X}$ at a later time t as

$$\mathbf{X}(t) = R(t)\mathbf{X}(t_0), \tag{2.14}$$

$R(t)$ can be interpreted as the *state transition matrix* for the ordinary differential equation (ODE)

$$\frac{d\mathbf{X}(t)}{dt} = \hat{\mathbf{w}}\mathbf{X}(t). \tag{2.15}$$

Now assuming $R(0) = \mathbf{I}_{3\times 3}$ is the initial condition for (2.13), with $t_0 = 0$, the unique solution to (2.15) is then given by

$$\mathbf{X}(t) = e^{\hat{\mathbf{w}}t}\mathbf{X}(0), \tag{2.16}$$

where $e^{\hat{\mathbf{w}}t}$ or $\exp(\hat{\mathbf{w}}t)$, is the matrix exponential

$$e^{\hat{\mathbf{w}}t} = \mathbf{I}_{3\times 3} + \hat{\mathbf{w}}t + \frac{(\hat{\mathbf{w}}t)^2}{2!} + \cdots + \frac{(\hat{\mathbf{w}}t)^n}{n!} + \ldots. \tag{2.17}$$

Hence, with regard to (2.14) and (2.16) it must hold

$$R(t) = e^{\hat{\mathbf{w}}t}. \tag{2.18}$$

It can directly be verified that $e^{\hat{\mathbf{w}}t}$ is in fact a rotation matrix, by showing that it

satisfies both group properties of the $\mathbb{SO}(3)$ as

$$\left(e^{\hat{\mathbf{w}}t}\right)^{-1} = e^{-\hat{\mathbf{w}}t} = e^{\hat{\mathbf{w}}^\top t} = \left(e^{\hat{\mathbf{w}}t}\right)^\top$$

and

$$\det\left(e^{\hat{\mathbf{w}}t}\right) = e^0 = +1,$$

since $\det(e^A) = e^{\text{trace}(A)}$ for any square matrix $A \in \mathbb{R}^{n\times n}$ with $\text{trace}(A) \doteq \sum_{i=1}^n a_{ii}$. A physical interpretation of (2.18) is that if $\|\mathbf{w}\|_2 = 1$, it simply describes a rotation around $\mathbf{w} \in \mathbb{R}^3$ by an angle of t radians, which will thus from now on be denoted by θ. In general θ can be absorbed into $\mathbf{w}$, resulting in

$$R = \exp(\hat{\mathbf{w}}), \tag{2.19}$$

for $\mathbf{w} = \theta[\omega_1, \omega_2, \omega_3]^\top$ with arbitrary norm $\|\mathbf{w}\|_2 = \theta$. Hence, from now on whenever explicitly denoted as $\theta\mathbf{w}$ or $\theta\hat{\mathbf{w}}$, it is implied that $\|\mathbf{w}\|_2 = 1$. This means that the matrix exponential (2.17) defines a map from the space $\mathfrak{so}(3)$ to $\mathbb{SO}(3)$ as,

$$\exp : \mathfrak{so}(3) \to \mathbb{SO}(3); \quad \hat{\mathbf{w}} \mapsto e^{\hat{\mathbf{w}}}, \tag{2.20}$$

called the *exponential map*. The expression (2.18) was obtained by assuming that $\mathbf{w}(t)$ in (2.11) is constant. However, it remains to show that in fact every rotation matrix $R \in \mathbb{SO}(3)$ can be expressed in exponential form, i.e that there exists a (not necessarily unique) $\theta\mathbf{w} \in \mathbb{R}^3$ such that $R = \exp(\theta\hat{\mathbf{w}})$. For this, the inverse of the exponential map is denoted by

$$\theta\hat{\mathbf{w}} = \log(R).$$

It can be shown by construction that for any $R \neq \mathbf{I}_{3\times3}$ the corresponding so-called *exponential coordinates* $\theta\mathbf{w}$ are given by

$$\theta = \cos^{-1}\left(\frac{\text{trace}(R) - 1}{2}\right), \quad \mathbf{w} = \frac{1}{2\sin(\theta)}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}. \tag{2.21}$$

If and only if $R = \mathbf{I}_{3\times3}$, then $\theta = 0$. In this case $\mathbf{w}$ is not determined and can there-

fore be chosen arbitrarily. This now explicitly means that every three-dimensional rotation can be realized by rotating around a fixed axis $\mathbf{w}$ by a certain angle θ. Thus $\theta\mathbf{w}$ is also often called the *axis-angle* representation of rotation.

While (2.21) can be used to directly obtain the exponential coordinates $\theta\mathbf{w}$ from a given R, there is not yet a closed form solution to the matrix exponential (2.17) in order to compute R efficiently from a given $\theta\mathbf{w}$. Such a solution can be derived knowing that if $\|\mathbf{w}\|_2 = 1$, then the powers of $\hat{\mathbf{w}}$ can be reduced by the two formulae

$$\hat{\mathbf{w}}^2 = \mathbf{w}\mathbf{w}^\top - \mathbf{I}_{3\times 3} \quad \text{and} \quad \hat{\mathbf{w}}^3 = -\hat{\mathbf{w}},$$

which is immediate to verify. Hence (2.17) with $t = \theta$ can be simplified as

$$e^{\theta\hat{\mathbf{w}}} = \mathbf{I}_{3\times 3} + \left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \dots\right)\hat{\mathbf{w}} + \left(\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} - \dots\right)\hat{\mathbf{w}}^2,$$

where the two sets of brackets contain the Taylor series for $\sin(\theta)$ and $1-\cos(\theta)$, respectively. Thus the matrix exponential can now be written and explicitly computed in closed form as

$$\exp(\theta\hat{\mathbf{w}}) = \mathbf{I}_{3\times 3} + \hat{\mathbf{w}}\sin(\theta) + \hat{\mathbf{w}}^2\left(1 - \cos(\theta)\right), \tag{2.22}$$

for $\|\mathbf{w}\|_2 = 1$, which is commonly known as *Rodrigues' formula*. Given this formula it is immediate to see that adding 2π to the norm θ of any $\mathbf{w}$ would result in the same rotation matrix, or in general

$$R = \exp(\theta\hat{\mathbf{w}}) = \exp((\theta + 2k\pi)\hat{\mathbf{w}}), \quad \forall k \in \mathbb{Z}.$$

Therefore there are infinitely many exponential coordinates to a given rotation matrix, which means that the exponential map $\exp : \mathfrak{so}(3) \to \mathbb{SO}(3)$ is not a one-to-one mapping. Furthermore the exponential map is not commutative, meaning that $e^{\hat{\mathbf{w}}_1}e^{\hat{\mathbf{w}}_2} \neq e^{\hat{\mathbf{w}}_2}e^{\hat{\mathbf{w}}_1} \neq e^{\hat{\mathbf{w}}_1+\hat{\mathbf{w}}_2}$, for $\hat{\mathbf{w}}_1, \hat{\mathbf{w}}_2 \in \mathfrak{so}(3)$, unless $\hat{\mathbf{w}}_1\hat{\mathbf{w}}_2 = \hat{\mathbf{w}}_2\hat{\mathbf{w}}_1$.

The three dimensional exponential coordinates provide an unconstrained and minimal parametrization of the 3DOF described by the rotation matrix and together with the translation vector they seem like an overall ideal representation of a 6DOF

pose. In fact when it comes to pose optimization or tracking there is an even more useful way to represent full rigid body motion, that extends the idea of exponential coordinates to the $\mathbb{SE}(3)$. Building up on the results for rotational motion, the rest of the section will introduce a similar representation for full rigid body motion which will be used extensively throughout this thesis.

Given a trajectory $T(t) : \mathbb{R} \to \mathbb{SE}(3)$ that describes the continuous motion of a rigid body as

$$T(t) = \begin{bmatrix} R(t) & \mathbf{t}(t) \\ \mathbf{0}^\top & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4},$$

one can first look at the structure of the matrix

$$\begin{aligned} \frac{dT(t)}{dt}T^{-1}(t) &= \begin{bmatrix} \frac{dR(t)}{dt} & \frac{d\mathbf{t}(t)}{dt} \\ \mathbf{0}^\top & 0 \end{bmatrix} \begin{bmatrix} R^\top(t) & -R^\top(t)\mathbf{t}(t) \\ \mathbf{0}^\top & 1 \end{bmatrix} \\ &= \begin{bmatrix} \frac{dR(t)}{dt}R^\top(t) & \frac{d\mathbf{t}(t)}{dt} - \frac{dR(t)}{dt}R^\top(t)\mathbf{t}(t) \\ \mathbf{0}^\top & 0 \end{bmatrix} \in \mathbb{R}^{4\times 4}, \end{aligned}$$

in analogy with the pure rotational case. From (2.10) it is known that $\frac{dR(t)}{dt}R^\top(t)$ is a skew-symmetric matrix, which means that there exists a $\hat{\mathbf{w}}(t) \in \mathfrak{so}(3)$ such that $\hat{\mathbf{w}}(t) = \frac{dR(t)}{dt}R^\top(t)$. By further defining

$$\mathbf{v}(t) \doteq \frac{d\mathbf{t}(t)}{dt} - \hat{\mathbf{w}}(t)\mathbf{t}(t)$$

and

$$\hat{\xi}(t) \doteq \frac{dT(t)}{dt}T^{-1}(t),$$

the above equation can be rewritten as

$$\hat{\xi}(t) = \begin{bmatrix} \hat{\mathbf{w}}(t) & \mathbf{v}(t) \\ \mathbf{0}^\top & 0 \end{bmatrix} \in \mathbb{R}^{4\times 4}.$$

This then gives

$$\frac{dT(t)}{dt} = \left(\frac{dT(t)}{dt}T^{-1}(t)\right)T(t) = \hat{\xi}(t)T(t), \tag{2.23}$$

where $\hat{\xi}$ can be interpreted as the "tangent vector" along the curve of $T(t)$, that

thus gives a first-order approximation to a rigid body transform around $\mathbf{I}_{4\times4}$ as

$$T(t+dt) \approx T(t) + \hat{\xi}(t)T(t)dt = \left(\mathbf{I}_{4\times4} + \hat{\xi}(t)dt\right)T(t).$$

A 4×4 matrix of the form

$$\hat{\xi} \doteq \begin{bmatrix} 0 & -\omega_3 & \omega_2 & v_1 \\ \omega_3 & 0 & -\omega_1 & v_2 \\ -\omega_2 & \omega_1 & 0 & v_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \in \mathfrak{se}(3),$$

is called a *twist*. The set of all twists is denoted by

$$\mathfrak{se}(3) \doteq \left\{ \hat{\xi} = \begin{bmatrix} \hat{\mathbf{w}} & \mathbf{v} \\ \mathbf{0}^\top & 0 \end{bmatrix} \in \mathbb{R}^{4\times4} \;\middle|\; \hat{\mathbf{w}} \in \mathfrak{so}(3),\, \mathbf{v} \in \mathbb{R}^3 \right\},$$

which is also called the tangent space (or Lie algebra) of the $\mathbb{SE}(3)$. Each twist $\hat{\xi} \in \mathfrak{se}(3)$ is parametrized by a six-dimensional vector

$$\xi \doteq \begin{bmatrix} \mathbf{w} \\ \mathbf{v} \end{bmatrix} = [\omega_1, \omega_2, \omega_3, v_1, v_2, v_3]^\top \in \mathbb{R}^6,$$

the so-called *twist coordinates*. Here, $\mathbf{w}$ is usually referred to as the *angular velocity* and $\mathbf{v}$ as the *linear velocity*, which indicates their relation to either the rotational or the translational part of the full rigid body motion.

Based on this, in the following it will now be derived how in fact every rigid body transform can be represented by such vector of twist coordinates. For this, again first a special case is considered, where $\hat{\xi}(t)$ is assumed to be constant i.e. $\hat{\xi}(t) = \hat{\xi}$ in (2.23), yielding

$$\frac{dT(t)}{dt} = \hat{\xi}T(t).$$

As previously in (2.13) $T(t)$ can now be interpreted as a state transition matrix of a linear ODE which can be integrated to give

$$T(t) = e^{\hat{\xi}t}T(0).$$

When also assuming the initial condition $T(0) = \mathbf{I}_{4\times4}$ it holds

$$T(t) = e^{\hat{\xi}t},$$

with the twist exponential being

$$e^{\hat{\xi}t} = \mathbf{I}_{4\times4} + \hat{\xi}t + \frac{(\hat{\xi}t)^2}{2!} + \cdots + \frac{(\hat{\xi}t)^n}{n!} + \ldots. \tag{2.24}$$

Just as in (2.19) t can again be absorbed into $\hat{\xi}$, giving

$$T = \exp(\hat{\xi}).$$

Similar to the previous notation $\theta\mathbf{w}$ the rotation angle θ can now also be factored out of any twist coordinate vector with $\mathbf{w} \neq \mathbf{0}$ as

$$\theta\xi = \theta[\omega_1, \omega_2, \omega_3, v_1, v_2, v_3]^\top \in \mathbb{R}^6,$$

such that $\|\mathbf{w}\|_2 = 1$, which is indicated by the notation $\theta\xi$ and $\theta\hat{\xi}$ in the following. This illustrates one of the main advantages of twist coordinates, being that in case of any rotation $R \neq \mathbf{I}_{3\times3}$, θ provides a one-parametric coupling of the rotational with the translational component of the corresponding rigid body motion. This property will be of great use in the rest of the thesis whenever it comes to numerical optimization with respect to the object's pose. By using this notation and exploiting additional properties of the matrix exponential, it can be shown that there exists a closed form solution to (2.24) which is given by

$$\exp(\theta\hat{\xi}) = \begin{bmatrix} \exp(\theta\hat{\mathbf{w}}) & (\mathbf{I}_{3\times3} - \exp(\theta\hat{\mathbf{w}}))\hat{\mathbf{w}}\mathbf{v} + \mathbf{w}\mathbf{w}^\top\mathbf{v}\theta \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad \text{if} \quad \theta \neq 0, \tag{2.25}$$

and otherwise simply

$$\exp(\hat{\xi}) = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{v} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad \text{if} \quad \theta = 0,$$

where in the first case $\exp(\theta\hat{\mathbf{w}})$ can be computed using Rodrigues' formula (2.22). Clearly in both cases the matrix exponential of a twist results in a rigid body

transformation and therefore defines an exponential map from the space $\mathfrak{se}(3)$ to $\mathbb{SE}(3)$ as,

$$\exp : \mathfrak{se}(3) \to \mathbb{SE}(3); \quad \hat{\xi} \mapsto e^{\hat{\xi}}.$$

Thus, a twist $\hat{\xi} \in \mathfrak{se}(3)$ is also called the *exponential coordinates* for $\mathbb{SE}(3)$, like $\hat{\mathbf{w}} \in \mathfrak{so}(3)$ is for $\mathbb{SO}(3)$. As before, the question arises whether this holds without the assumption of $\hat{\xi}$ being constant, i.e. if there exist (not necessarily unique) twist coordinates ξ such that $T = \exp(\hat{\xi})$, for every $T \in \mathbb{SE}(3)$. The answer is again yes, which can also be shown by construction of the inverse of the exponential map denoted by $\hat{\xi} = \log(T)$.

As previously shown in (2.21), given any $T \in \mathbb{SE}(3)$ one can always determine $\theta\mathbf{w}$ from the rotation matrix $R \in \mathbb{SO}(3)$ therein. If $R \neq \mathbf{I}_{3\times3}$, i.e. $\theta \neq 0$ the twist coordinates can be determined as

$$\theta\xi = \begin{bmatrix} \theta\mathbf{w} \\ \theta\mathbf{v} \end{bmatrix}, \quad \text{with} \quad \mathbf{v} = \left((\mathbf{I}_{3\times3} - R)(\hat{\mathbf{w}} + \mathbf{w}\mathbf{w}^\top\theta)\right)^{-1}\mathbf{t},$$

resolved from $\mathbf{t} = (\mathbf{I}_{3\times3} - \exp(\theta\hat{\mathbf{w}}))\hat{\mathbf{w}}\mathbf{v} + \mathbf{w}\mathbf{w}^\top\mathbf{v}\theta$ in (2.25). Otherwise, if $R = \mathbf{I}_{3\times3}$, i.e. $\theta = 0$ the twist coordinates are simply

$$\xi = \begin{bmatrix} \mathbf{0} \\ \mathbf{t} \end{bmatrix} = [0, 0, 0, t_x, t_y, t_z]^\top.$$

The physical interpretation of this is that the rigid body motion between any two arbitrary rigid body transforms can be realized as a rotation around a particular axis in space and a translation along that axis, which is also known as *Chasles' theorem* [Bal98, MLS94].

This section concludes by demonstrating how twists can be used in order to linearly interpolate the rigid body motion that occurred between two frames. In general by scaling any twist $\hat{\xi} = \log(T)$ by a factor $\lambda \in [0, 1]$ the matrix exponential $\exp(\lambda\hat{\xi})$ interpolates linearly between the initial configuration $\exp(0\hat{\xi}) = \mathbf{I}_{4\times4}$ and the final configuration $\exp(1\hat{\xi}) = T$. Now with regard to inter-frame rigid body motion, let $\hat{\xi}(t_2, t_1) = \log(T_{cc}(t_2, t_1))$ be the twist representation of the pose change that occurred between the two consecutive frames at t_1 and t_2 (see Figure 2.3). By

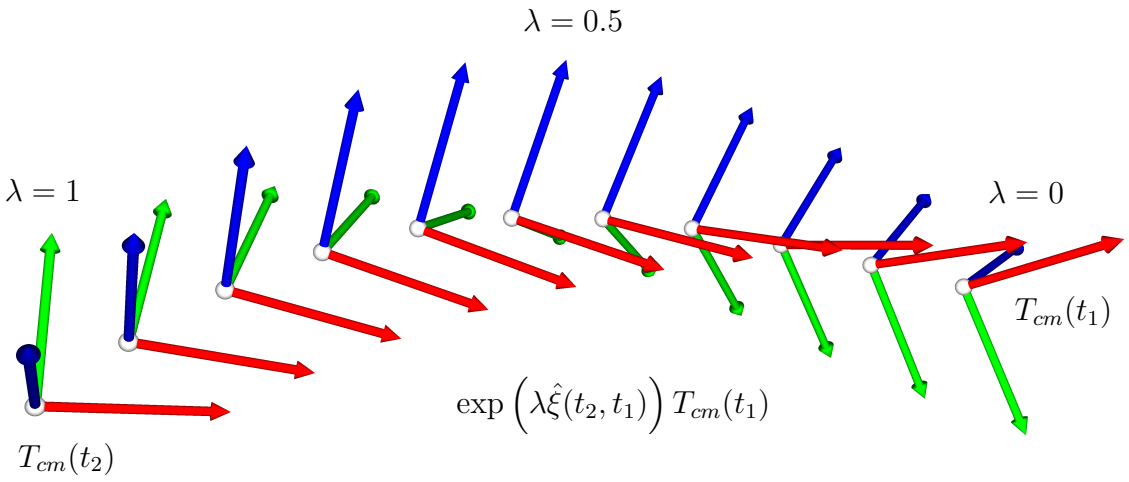


Figure 2.3: Linear interpolation of rigid body motion using twists. A single parameter λ can be used to interpolate a 6DOF motion path between the two rigid body transforms $T_{cm}(t_1)$ and $T_{cm}(t_2)$ by linearly scaling the twist $\hat{\xi}(t_2, t_1)$, that parametrizes the inter-frame motion $T_{cc}(t_2, t_1)$. This visualizes the one-parametric coupling of the rotation with the translation within the twist coordinates.

multiplying it on the right, the matrix exponential is transported to $T_{cm}(t_1)$ as

$$\exp\left(\lambda\hat{\xi}(t_2,t_1)\right) T_{cm}(t_1).$$

Accordingly the above expression results in $T_{cm}(t_1)$ for $\lambda = 0$ and $T_{cm}(t_2)$ for $\lambda = 1$ respectively, meaning that by changing λ one can obtain a continuous 6DOF motion path between any two rigid body transforms i.e. estimated poses.

2.1.2 Other Representations of Rotation

The $\mathfrak{so}(3)$ and the $\mathfrak{se}(3)$ are optimal representations for interpolating between orientations and rigid body transforms respectively. However they are not very intuitively applicable for constructing a specifying pose. Especially with regard to rotation, the axis-angle representation $\theta\mathbf{w}$ is not very descriptive when trying to realize a desired rotation of e.g. *76 degrees around the X-axis and 43 degrees around the Y-axis* to obtain a view of an object from that perspective. For such purposes a more descriptive

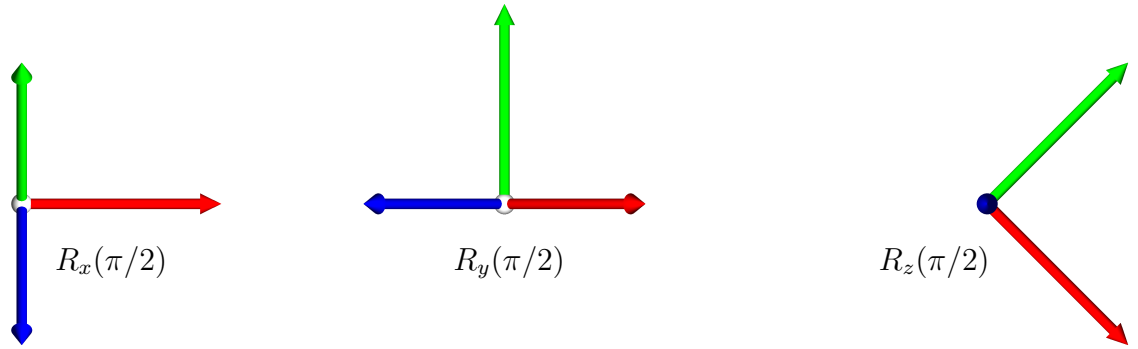


Figure 2.4: A visualization of the rotation described by matrices $R_x(\alpha)$, $R_y(\beta)$ and $R_z(\gamma)$ corresponding to the Euler angles. Here each matrix rotates around the respective principal axis by an Euler angle of 45°.

parametrization that also defines a mapping from $\mathbb{R}^3$ to $\mathbb{SO}(3)$ are the well-known Euler angles denoted by α, β, γ (see Figure 2.4). Here, α is the angle of rotation around the X-axis $\mathbf{e}_1 \doteq [1,0,0]^\top$ expressed by the matrix

$$R_x(\alpha) \doteq \exp(\alpha \hat{\mathbf{e}}_1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \in \mathbb{SO}(3),$$

β the angle of rotation around the Y-axis $\mathbf{e}_2 \doteq [0,1,0]^\top$ expressed by the matrix

$$R_y(\beta) \doteq \exp(\beta \hat{\mathbf{e}}_2) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \in \mathbb{SO}(3),$$

and γ the angle of rotation around the Z-axis $\mathbf{e}_3 \doteq [0,0,1]^\top$ expressed by the matrix

$$R_z(\gamma) \doteq \exp(\gamma \hat{\mathbf{e}}_3) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{SO}(3),$$

where $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ are the principal axes of the Cartesian coordinate system in $\mathbb{R}^3$. The angles α, β, γ are also often referred to as "roll", "pitch" and "yaw". In the context of monocular pose estimation the optical axis orthogonal to the image plane of the

camera is commonly chosen to be $\mathbf{e}_3$ as explained in the following Section 2.2. Thus, here γ is also called the *in-plane rotation* while α, β are called the *outer image plane rotation*. Given these three corresponding matrices it can be shown that for every $R \in \mathbb{SO}(3)$ there exist e.g. so-called ZYX Euler angles γ, β, α such that

$$R(\gamma, \beta, \alpha) = R_z(\gamma)R_y(\beta)R_x(\alpha) \in \mathbb{SO}(3).$$

In general the order of multiplication of the three matrices can be chosen in multiple ways, meaning that for example given a rotation matrix $R(\gamma_1, \beta_1, \alpha_1)$ from ZYX Euler angles there also exist YXZ Euler angles $\beta_2, \alpha_2, \gamma_2$ such that $R(\gamma_1, \beta_1, \alpha_1) = R(\beta_2, \alpha_2, \gamma_2)$. In total there are twelve unique sequences of proper Euler angles, namely XYX, XYZ, XZX, XZY, YXY, YXZ, YZX, YZY, ZXY, ZXZ, ZYX and ZYZ. This ambiguity is one of two main downsides of the Euler angles representation. The other is that there are instances of this representation that result in singularities. For example, the ZYX Euler angles become singular when $\beta = \pi/2$. The presence of such singularities along with the rotation sequence ambiguity make them a bad parametrization choice with regard to rigid body motion. Thus in this work Euler angles will exclusively be used whenever it is required to construct specific static views of an object (e.g Section 4.6.2).

Finally, for the sake of completeness, the popular concept of representing spatial rotation with *unit quaternions* will be briefly explained in the following. For this, given that complex numbers can be defined as $\mathbb{C} \doteq \mathbb{R} + \mathbb{R}i$ with $i^2 = -1$, the set of all quaternions can simply be defined as $\mathbb{H} \doteq \mathbb{C} + \mathbb{C}j$, with $j^2 = -1$ and $i \cdot j = -j \cdot i$. In the following an arbitrary quaternion is then denoted by

$$\mathbf{q} \doteq (q_w, \mathbf{q}_v) = (q_w, [q_x, q_y, q_z]^\top) \in \mathbb{H}, \quad q_w, q_x, q_y, q_z \in \mathbb{R},$$

where q_w is the scalar real component and $\mathbf{q}_v$ the vector of the three imaginary components such that $\mathbf{q} = q_w + q_x i + (q_y + q_z i)j = q_w + q_x i + q_y j + q_z ij$. The set of all unit quaternions is then denoted by

$$\mathbb{S}^3 \doteq \{\mathbf{q} \in \mathbb{H} \mid \|\mathbf{q}\|_2 = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2} = 1\},$$

forming a special subgroup that embeds the rotation group $\mathbb{SO}(3)$ in the quaternion

field $\mathbb{H}$. In fact any unit quaternion $\mathbf{q}$ can be rewritten as

$$\mathbf{q} = (q_w, \mathbf{q}_v) = \left(\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\mathbf{w}\right) \in \mathbb{S}^3,$$

describing a rotation by angle θ around an axis $\mathbf{w}$, with $\|\mathbf{w}\|_2 = 1$. Thus, any given $R = \exp(\theta\hat{\mathbf{w}}) \in \mathbb{SO}(3)$ can be associated with a unit quaternion $\mathbf{q}(R) \in \mathbb{S}^3$. It can be shown that this association preserves the group structure between $\mathbb{SO}(3)$ and $\mathbb{S}^3$ as

$$\mathbf{q}(R^\top) = \mathbf{q}^{-1}(R), \quad \mathbf{q}(R_1R_2) = \mathbf{q}(R_1)\mathbf{q}(R_2), \quad \forall R, R_1, R_2 \in \mathbb{SO}(3),$$

and that it is genuine, meaning that different rotation matrices are associated with different unit quaternions. In the opposite direction, given a unit quaternion $\mathbf{q} = (q_w, \mathbf{q}_v)$ the corresponding rotation matrix $R(\mathbf{q}) = \exp(\theta\hat{\mathbf{w}})$ is determined by the angle $\theta = 2\cos^{-1}(q_w)$ and the axis $\mathbf{w} = \mathbf{q}_v/\sin(\theta/2)$, if $\theta \neq 0$ and $\mathbf{w} = [0,0,0]^\top$ otherwise. Therefore, $R(\mathbf{q})$ could then be computed using Rodrigues' formula.

Alternatively, in the context of unit quaternions a different closed form solution is commonly used to compute this conversion to the corresponding rotation matrix directly from $\mathbf{q}$ as

$$R(\mathbf{q}) = \begin{bmatrix} 1-2(q_y^2+q_z^2) & 2(q_xq_y - q_zq_w) & 2(q_xq_z + q_yq_w) \\ 2(q_xq_y + q_zq_w) & 1-2(q_x^2+q_z^2) & 2(q_yq_z - q_xq_w) \\ 2(q_xq_z - q_yq_w) & 2(q_yq_z + q_xq_w) & 1-2(q_x^2+q_y^2) \end{bmatrix} \in \mathbb{SO}(3),$$

without having to first explicitly extract the axis-angle representation. According to the above formula it becomes clear that $R(\mathbf{q}) = R(-\mathbf{q})$, meaning that always two unit quaternions $\mathbf{q}$ and $-\mathbf{q}$ correspond to the same rotation matrix. Therefore, compared to the exponential coordinates, where infinitely many (all related by periodicity) exist for each rotation matrix, by using unit-quaternions there is less redundancy in the representation. Both quaternions and exponential coordinates are singularity free and provide smooth interpolations between two rotations, whereas for quaternions the constraint to be of unit length must always be enforced in order to remain within the $\mathbb{S}^3$. This constraint along with requiring four instead of three parameters are the main drawbacks compared to the unconstrained exponential coordinates, especially when it comes to numerical optimization [Gra98].

2.2 Projective Geometry

Recovering the pose of a 3D object from a 2D image can be seen as the inverse problem of understanding how an image of that object is formed. Therefore in order to be able to solve the actual pose estimation problem, first a mathematical model of image formation is needed, that describes how the 3D geometry of objects projects into the camera and eventually results in an image of them.

In this work an image (also called a frame with regard to an image sequence) is denoted by I, in general defined on a compact image domain $\Omega \subset \mathbb{R}^2$. In case of a common camera Ω is a planar, rectangular region representing the 2D image sensor. Points within the image are denoted by

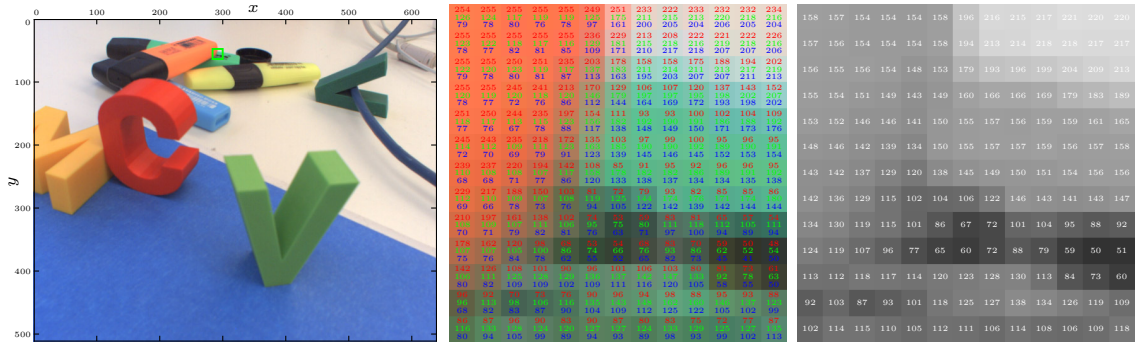
$$\mathbf{x} \doteq \begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{R}^2.$$

Therefore I is a function of image intensities

$$I : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^k; \quad \mathbf{x} \mapsto I(\mathbf{x}),$$

where k is the number of so-called intensity channels. For instance, in case of an RGB color camera image, $I : \Omega \to \mathbb{R}^3$, the vector of red, green and blue intensities i.e. the color at each point $\mathbf{x}$ in the image is given by $\mathbf{y} = I(\mathbf{x}) = [r, g, b]^\top \in \mathbb{R}^3$. In case of a monochrome (e.g. grayscale) camera image, $I : \Omega \to \mathbb{R}$, $l = I(\mathbf{x}) \in \mathbb{R}$ gives the brightness or luminance at that image point represented in form of a single intensity channel. In the rest of the thesis the number of channels of a given image will be made clear in the respective context whenever necessary.

In practice the image domain of a digital image is discretized such that $\Omega = [0, w-1] \times [0, h-1] \subset \mathbb{Z}^2$ represents a 2D array of $w \times h$ pixels (see Figure 2.5). Thus $w \in \mathbb{Z}_+$ is the image width, i.e. the number of pixels per image row, and $h \in \mathbb{Z}_+$ is the image height i.e. the number of pixel rows. The coordinate origin of the image plane is typically in its top left corner in relation to the coordinate frames of most digital screens. Each pixel contains a k-dimensional vector of discretized intensities. For images captured by a digital camera, each intensity channel is commonly mapped to



(a) (b) (c)

Figure 2.5: An example of a digital RGB color camera image I of 640×512 pixels (a). An image patch of 13×13 pixels is marked by a green outline that is shown in detail in (b) and (c). Here the RGB intensity values are depicted as an overlay for each pixel in the original color image patch (b) as well as the luminance for the same pixels after a grayscale conversion (c).

an interval of integers $[0, 255] \subset \mathbb{Z}$ using 8-bit quantization. Note that this intensity discretization does not necessarily apply to all types of image used in this work, e.g. synthetically generated images (see Section 2.2.4).

The pixel intensity describes the light energy falling onto the corresponding area of the imaging sensor integrated over time (determined by the shutter speed of the camera). Apart from the shutter speed and the area of the pixel region the intensity also depends on the scene (meaning the objects and the lighting) in front of the camera as well as the camera's optics. In general a camera is composed of a set of lenses that direct light within the optical system by means of diffraction, refraction and reflection. This makes it very complex to model mathematically in order to be physically correct. Therefore, in this work (as mostly done in computer vision) the camera is modeled by a so-called *ideal pinhole camera*, that is a suitable trade off between physical correctness and mathematical simplicity within the context of object pose estimation. Here, only the refraction of a single lens with an infinitely small circular aperture (the hole) is considered for light propagation. In reality, when the aperture decreases to zero the amount of light going through the lens would also become zero. Thus, the pinhole model will just be considered a good geometric approximation of a well-focussed camera (see Figure 2.6).

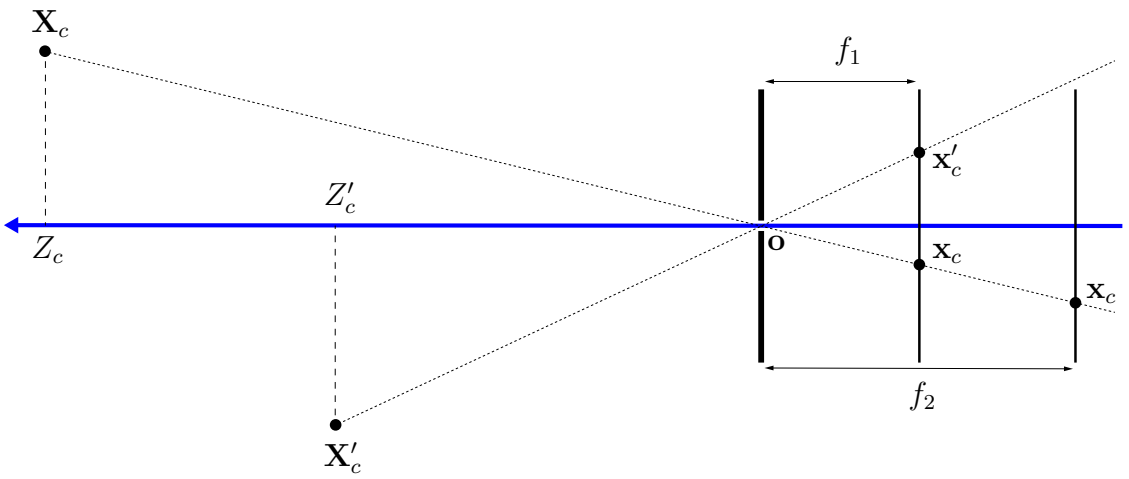


Figure 2.6: The pinhole camera model in a schematic 2D side view in parallel to the optical axis. In this example two object points $\mathbf{X}_c$ and $\mathbf{X}'_c$ project into the image plane shown at two different focal lengths f_1 and f_2. In case of the smaller focal length f_1 both projections $\mathbf{x}_c$ and $\mathbf{x}'_c$ are within the image plane whereas for the larger focal length f_2, the ray going through $\mathbf{X}'_c$ and $\mathbf{o}$ does not intersect with the image plane.

2.2.1 The Pinhole Camera Model

A pinhole camera is defined by an axis, called the *optical axis* and a plane perpendicular to that axis, called the *focal plane*. At the intersection of the optical axis with the focal plane is the optical center $\mathbf{o}$ (the hole) which all rays are forced to go through. In this model the image plane is behind the optical center at a distance f, called the focal length. Here the optical axis is the Z_c-axis of the camera coordinate frame of reference $\mathcal{C}$ with $\mathbf{o}$ being its origin. A projected point $\mathbf{x}_c$ in the image plane that corresponds to a point $\mathbf{X}_c$ given with respect to the camera frame of reference is then simply determined by the intersection of the ray going trough $\mathbf{X}_c$ and $\mathbf{o}$ with the image plane. The coordinates of $\mathbf{x}_c$ are thus given by

$$x_c = -f\frac{X_c}{Z_c}, \quad y_c = -f\frac{Y_c}{Z_c}, \tag{2.26}$$

which is called ideal *perspective projection* and hence $\mathbf{x}_c = [x_c, y_c]^\top$ are called *ideal image coordinates*. Accordingly the pinhole camera model has one parameter, the

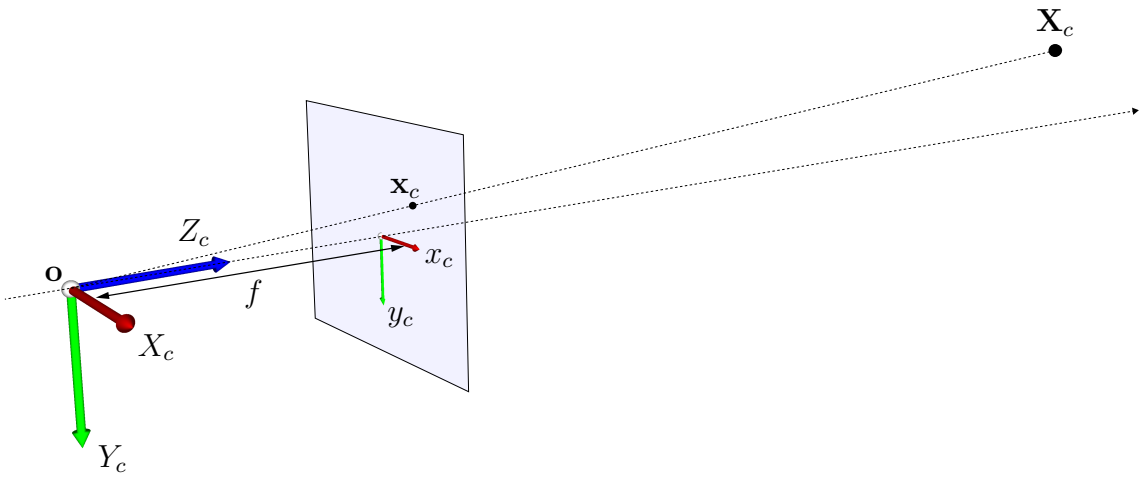


Figure 2.7: An example 3D visualization of the frontal pinhole camera model under the assumption of ideal perspective projection.

focal length f, that implicitly determines the field of view of the camera. Since the image plane is assumed to be of limited size, the focal length determines at which angle of entry a ray still intersects with it. In general it holds, the smaller the focal length, the larger the field of view of the camera and vice versa. Due to the negative sign in each equation of (2.26) the projection of any object appears flipped in both axes in the image plane. This can be eliminated by flipping the image by $(x_c, y_c) \mapsto (-x_c, -y_c)$, which is equivalent to placing the image plane in front of the optical center (i.e. $Z_c = +f$) instead of behind it (i.e. $Z_c = -f$) as previously shown (see Figure 2.7).

With this small tweak the model turns into a more convenient so-called *frontal* pinhole camera, which will be used throughout the rest of this thesis. Within this model the coordinates of image point $\mathbf{x}_c = [x_c, y_c]^\top$ corresponding to $\mathbf{X}_c = [X_c, Y_c, Z_c]^\top$ are then given by

$$x_c = f\frac{X_c}{Z_c}, \quad y_c = f\frac{Y_c}{Z_c}, \tag{2.27}$$

with the optical axis being the Z_c-axis of the camera $\mathcal{C}$, its X_c-axis and its Y_c-axis are chosen in parallel direction to the edges of the image plane and the optical center $\mathbf{o}$ being its origin. Note that the principal axes of $\mathcal{C}$ are typically depicted with Y_c pointing downwards and X_c pointing to the right (see Figure 2.7), such that their

directions coincide with the coordinate frame of the digital image intensity array (shown in Figure 2.5).

In the previous section it was shown how to transform model points into the camera frame of reference under rigid body motion in linear form as $\tilde{\mathbf{X}}_c = T_{cm}\tilde{\mathbf{X}}_m$ using the homogeneous representation of both the transformation as well as the 3D coordinates. After having performed this rigid body transformation into the camera, the homogeneous representation $\tilde{\mathbf{X}}_c$ is no longer needed for subsequent transformations with regard to image formation. In this work the notation $\mathbf{X}_c = (\tilde{\mathbf{X}}_c)_{3\times 1}$ is used in order to simply drop the homogeneous component and thereby obtain the corresponding 3D point, whenever required.

In analogy to the homogeneous representation of 3D points in (2.6), the same notation

$$\tilde{\mathbf{x}} \doteq \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \in \mathbb{R}^3,$$

is used to represent any 2D point $\mathbf{x} = [x, y]$ in homogeneous form. Given this, the formulae in (2.27) describing ideal perspective projection of a 3D point $\mathbf{X}_c$ can then also be expressed in linear form as

$$Z_c\tilde{\mathbf{x}}_c = Z_c \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} = Z_c \begin{bmatrix} f\frac{X_c}{Z_c} \\ f\frac{Y_c}{Z_c} \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}, \tag{2.28}$$

by using homogeneous coordinates. This leads to the definition of the 3×3 homogeneous matrix

$$K_f \doteq \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3}.$$

In order to obtain the 2D coordinates in the image plane, perspective projection is denoted by

$$\mathbf{x}_c = \pi\left(Z_c\tilde{\mathbf{x}}_\mathbf{c}\right) \in \mathbb{R}^2,$$

where the so-called projection map

$$\pi : \mathbb{R}^n \mapsto \mathbb{R}^{n-1}; \quad \mathbf{a} \mapsto \pi(\mathbf{a}) = [a_1/a_n, \ldots, a_{n-1}/a_n]^\top, \quad \forall \mathbf{a} \in \mathbb{R}^n, \tag{2.29}$$

where $a_n \neq 0$, applies perspective division by the last entry and de-homogenization, e.g. $\pi(\mathbf{X}) = [X/Z, Y/Z]^\top$.

Here it becomes clear that all points on the line through $\mathbf{X}_c$ and $\mathbf{o}$ project to the same $\mathbf{x}_c$, regardless of their distance to the camera. Therefore in homogeneous representation, only the direction of $\tilde{\mathbf{x}}_c$ is important. As given in (2.28), $\mathbf{x}_c$ can be represented by $Z_c\tilde{\mathbf{x}}_\mathbf{c}$ for any non-zero $Z_c \in \mathbb{R} \setminus \{0\}$, where any such vector uniquely determines the intersection of the ray with the image plane at $Z_c = f$. Therefore $Z_c \in \mathbb{R}_+$ is also often called the *depth* of the image point $\mathbf{x}_c$, which is usually unknown when only given a 2D image. When f is known it can be normalized to 1. In this case the projection of a 3D point $\mathbf{X}_c$ in the image plane at $Z_c = 1$ is simply given by $\mathbf{x}_c = \pi(\mathbf{X}_c)$, which are then called *normalized ideal image coordinates*.

So far, the (normalized) ideal image coordinates $\mathbf{x}_c$ are specified with respect to an image plane within the camera coordinate frame $\mathcal{C}$, with its origin being located at the *principal point* (i.e. the intersection of the Z_c-axis with the image plane). However, as shown in Figure 2.5 the measurements of a digital camera are given with respect to the image coordinate frame of the pixel array with its origin located in the top-left corner. Thus in order to model the complete projection of 3D point to a pixel the relationship between normalized coordinates $\mathbf{x}_c = [x_c, y_c]^\top$ and coordinates within the pixel array, simply denoted by $\mathbf{x} = [x, y]^\top$, must be established (see Figure 2.8). Assuming x_c and y_c are specified in terms of metric units (e.g. millimeters), then they can be scaled to pixel coordinates by

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x_c \\ y_c \end{bmatrix},$$

where the scaling matrix that depends on the size of a pixel (in metric units) in x_c and y_c direction. While the pixels of current cameras are usually square, meaning $s_x \approx s_y$, in general they can have an aspect ratio of $s_x/s_y \neq 1$. In case of the pixels

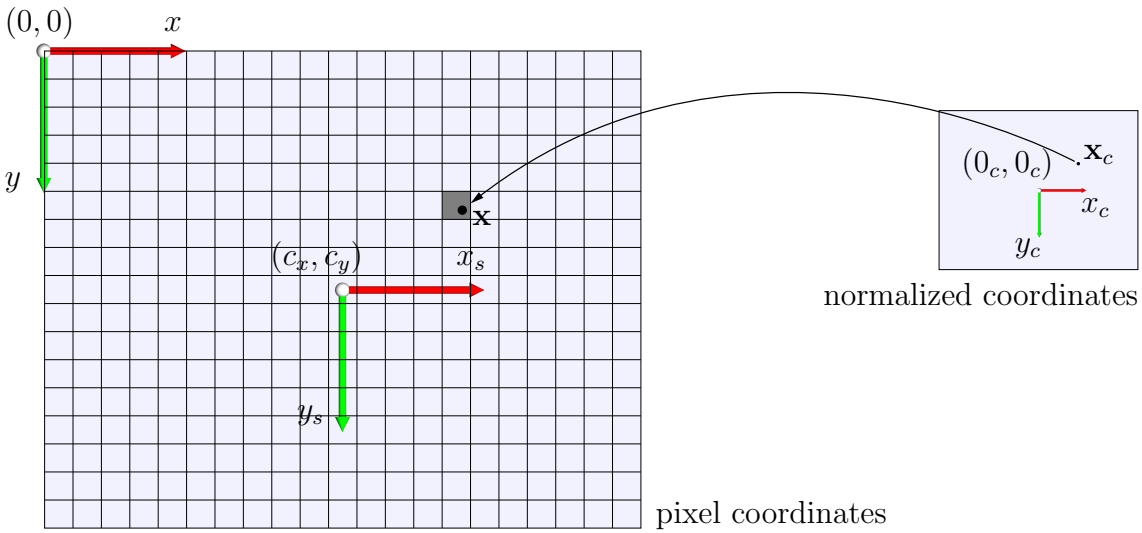


Figure 2.8: The 2D coordinate frame transformation from normalized image coordinates $\mathbf{x}_c$ to pixel coordinates $\mathbf{x}$ within a digital camera image.

not being rectangular, a more general form is given by

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} s_x & s_\theta \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x_c \\ y_c \end{bmatrix},$$

with a so-called *skew factor* s_θ, proportional to $\cot(\theta)$, where θ is the angle between the x_s- and y_s-axes. However, in practice this factor is neglectable due to the increased production quality of today's digital cameras and will thus be considered $s_\theta = 0$ in the following. Since x_s and y_s are still specified relative to the principal point, next the origin of the frame of reference must be translated to the top left corner by

$$x = x_s + c_x, \quad y = y_s + c_y,$$

in order to arrive at the final pixel coordinates. Here c_x and c_y are the coordinates (in pixels) of the principal point relative to the digital image reference frame. The above steps of coordinate scaling and translation can be expressed by a single linear transformation as

$$\tilde{\mathbf{x}} \doteq \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & c_x \\ 0 & s_y & c_x \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix},$$

where $\tilde{\mathbf{x}} = [x, y, 1]^\top$ is the homogeneous representation of the actual image coordinates in pixels (see again Figure 2.8). This linear transformation leads to a definition of another homogeneous 3×3 matrix

$$K_s \doteq \begin{bmatrix} s_x & 0 & c_x \\ 0 & s_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3},$$

that can directly be combined with the previous projection model (2.28). As a result, a more realistic projection model is obtained, that describes the complete transformation from a 3D point $\mathbf{X}_c$ in camera coordinates to the corresponding pixel coordinates $\mathbf{x}$ in a digital image I as

$$Z_c \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & c_x \\ 0 & s_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix},$$

expressed in homogeneous form. The combination of K_s and K_f is called the *calibration matrix* or *intrinsic parameter matrix*

$$K \doteq K_s K_f = \begin{bmatrix} s_x & 0 & c_x \\ 0 & s_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3},$$

as it contains all parameters that are "intrinsic" to a particular camera. Here, the geometric interpretation of $f_x = f s_x$ and $f_y = f s_y$ is the size of unit length in horizontal and vertical pixels respectively, whereas c_x and c_y are the x- and y-coordinates of the principal point measured in pixels.

These four intrinsic parameters can be obtained by the process of camera calibration (see Section 2.2.3). When K is known, the normalized ideal image coordinates $\mathbf{x}_c$ corresponding to given pixel coordinates $\mathbf{x}$ can be obtained by a simple inversion of K as

$$\tilde{\mathbf{x}}_c = K^{-1}\tilde{\mathbf{x}}, \tag{2.30}$$

where K^{-1} can be computed explicitly by

$$K^{-1} \doteq \begin{bmatrix} \frac{1}{f_x} & 0 & -\frac{c_x}{f_x} \\ 0 & \frac{1}{f_y} & -\frac{c_y}{f_y} \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3}.$$

Furthermore, when also the depth of a pixel is known the above equation can be extended in order to obtain the corresponding 3D point $\mathbf{X}_c$ in the camera via so-called *back-projection* as

$$\mathbf{X}_c = Z_c \tilde{\mathbf{x}}_c = Z_c K^{-1} \tilde{\mathbf{x}}, \tag{2.31}$$

which can eventually be transformed back to its model coordinates $\tilde{\mathbf{X}}_m = T_{cm}^{-1} \tilde{\mathbf{X}}_c$, assuming that the rigid body transform between $\mathcal{C}$ and $\mathcal{M}$ is known as well.

To summarize, the overall image formation starting with a 3D point $\mathbf{X}_m$ in model coordinates is captured by

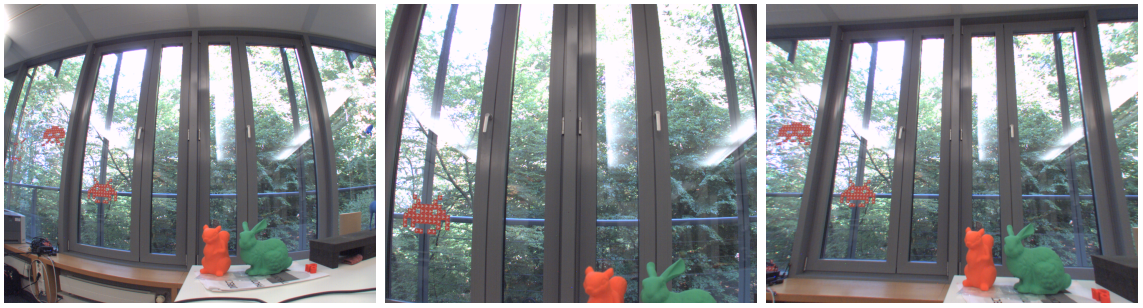
$$\mathbf{x} = \pi(K(T_{cm}\tilde{\mathbf{X}}_m)_{3\times 1}), \tag{2.32}$$

where $\mathbf{x}$ are the resulting 2D pixel coordinates within an image I giving rise to the corresponding intensity $I(\mathbf{x})$.

2.2.2 Radial Lens Distortion

While the parameters in K describe the linear properties of the camera, there are also non-linear effects in images mostly caused by distortions of the lens along radial directions. These radial distortions become more significant for small focal lengths i.e. cameras with a large field of view, causing straight lines in the scene to appear curved in the image (see Figure 2.9). Thus, in order to increase the reliability and accuracy of computer vision algorithms involving such images (e.g. 6DOF pose estimation) these distortions must be removed and therefore modeled appropriately. In general the relationship between distorted $\mathbf{x}_d = [x_d, x_d]^\top$ and undistorted pixel coordinates $\mathbf{x} = [x, y]^\top$ is described by

$$\mathbf{x} = \mathbf{c} + f(r)\,(\mathbf{x}_d - \mathbf{c})\,,$$



(a) (b) (c)

Figure 2.9: Digital images of a scene captured by the same camera with different focal lengths. As expected, the image captured with a focal length of ∼4 mm yields significantly more radial distortion (a) than the image captured with a focal length of ∼10 mm (b). The distortion coefficients can be used in order to rectify (or undistort) the images by removing the radial distortion (here shown for image (a)) (c).

with respect to the distance $r = \|\mathbf{x}_d - \mathbf{c}\|_2$ to the principal point $\mathbf{c} = [c_x, c_y]^\top$, which is considered the center of distortion. The distortion correction factor $f(r)$ is commonly modeled by a fourth degree polynomial

$$f(r) = 1 + k_1 r + k_2 r^2 + k_3 r^3 + k_4 r^4,$$

where k_1, k_2, k_3 and k_4 are the so-called *distortion coefficients* that also belong to the intrinsic parameters in addition to K.

2.2.3 Camera Calibration

The previous sections have shown how the projection of 3D points to pixels is modeled by the intrinsic parameters of the camera. These consist of f_x, f_y, c_x and c_x defining the intrinsic matrix K as well as the distortion coefficients k_1, k_2, k_3 and k_4. Therefore, in order to be able to compute accurate 3D measurements from a 2D digital image by inverting this image formation process, these intrinsic parameters have to be known. For monocular pose estimation it is valid to assume that the optics of the camera remain unchanged during the entire time of at least one motion measurement sequence. This means that the lens and the image sensor are in a fixed setup without changing the focal length (i.e. the zoom). Hence, for every

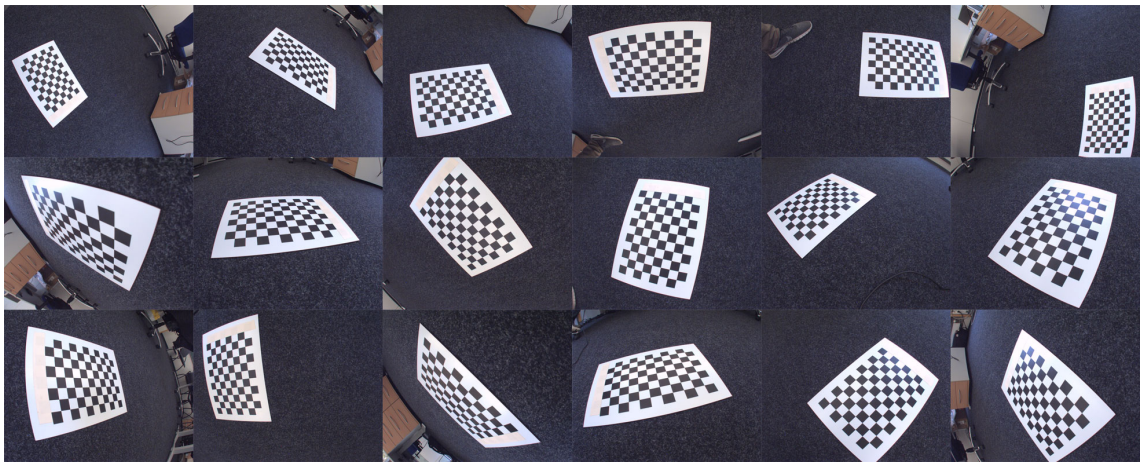


Figure 2.10: A set of example images used for camera calibration with a planar chessboard pattern. These images are a subset of those that were used to calibrate the camera that captured Figure 2.9 (a) in order to compute the undistorted result in Figure 2.9 (c).

camera such a calibration of the intrinsic parameters only has to be performed once in advance of the measurements. This process can furthermore be performed offline, independent of the employed pose estimation algorithm.

In the past sophisticated methods for camera calibration have been proposed (see e.g. [Zha00]) that are publicly available as software packages[1]. For such approaches it is required to capture a set of images of a so-called *calibration rig* (with known spatial geometry). These should ideally cover a large variety of distinct perspectives in order to maximize their information entropy, which increases the calibration's accuracy (see Figure 2.10). Here, usually a calibration rig in form of a planar chessboard pattern is used. Due to the high contrast of this pattern the projected corner locations of the squares can robustly be detected in all images. Together with the known geometry of the calibration rig these sets of observed 2D projections eventually allow to estimate all intrinsic parameters in a joint but relatively time-consuming fashion. Therefore, in the rest of this thesis the intrinsic parameters of a camera are assumed to be known from such an offline calibration procedure and the optics of the camera to remain fixed afterwards for all measurements.

[1]For example as part of the OpenCV (open computer vision) library: www.opencv.org

2.2.4 Synthetic Image Rendering

The core concept of the approaches presented in this work is to determine the quality of a pose hypothesis by measuring some sort of similarity between the object in a real camera image and a synthetic 2D projection of its 3D model abstraction, parametrized by the sought pose. Here, these synthetic views have to be generated using the same projection model as the camera including its calibrated intrinsic parameters in order to optimally match the real captured images.

For point-based approaches (see Chapter 3) where models consist of a sparse set of 3D points such projections can efficiently be computed using (2.32). On the other hand, region- and texture-based approaches require a dense 3D surface model of the object, here represented in form of a triangle mesh (see Figure 2.11). Such a mesh is defined by a set of so-called *corner vertices* $\mathbf{X}_{i_m}$, $i = 1, \ldots, n$, of which always three are connected to a triangle as part of the common surface. Thus rendering such a surface mesh requires not only projecting all corner vertices into the image plane but also filling the pixels of the triangle areas in between and thereby handle occlusions and intersections appropriately. This per pixel process of filling the triangles in the image plane is called rasterization. Here, also additional surface properties (e.g. the texture) have to be interpolated per pixel between the corner vertices with regard to their depth per triangle in order to yield a perspectively correct projection.

Depending on the model's complexity such a mesh can potentially consist of millions of triangles and thus in general cannot be rendered in real-time on a current CPU. However, rendering 3D triangle meshes is highly parallelizable. Thus, in this work 3D mesh models are rendered using the GPU of a graphics card, which is specifically well suited to perform such parallel computations. For this, the standard rendering functionality of the platform independent graphics API OpenGL[2] is used. In the following a brief overview of the image formation process within OpenGL with a strong focus on the projection model will be given. For more elaborate and in-depth information on the rendering pipeline please refer to e.g. [Ang06].

In OpenGL, perspective projection is also described using a pinhole model with 3D coordinates within the respective OpenGL camera coordinate frame denoted by $\mathbf{X}_{gl}$.

[2]More information on OpenGL (open graphics library) at: https://www.opengl.org/

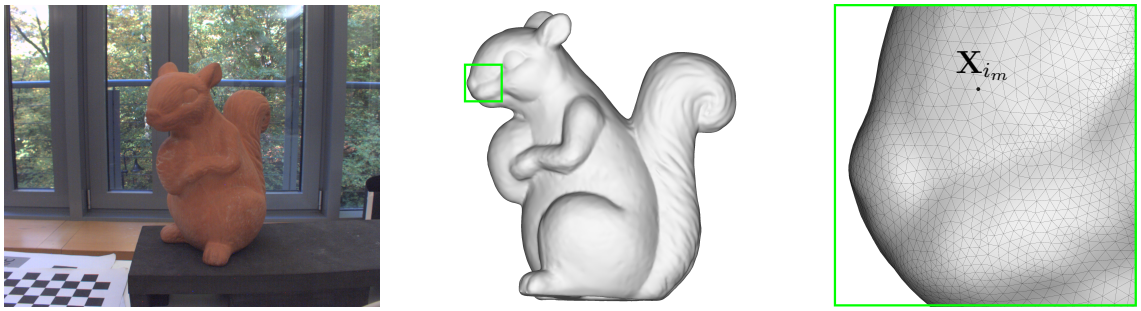


Figure 2.11: An example 3D triangle mesh surface model of a squirrel figurine. Left: An image showing the original figurine that was captured with a 3D laser scanner. Middle: A side view of the corresponding 3D model in form of a shaded rendering with regard to the surface normals. Right: The nose of the squirrel in a magnified detail view (marked left by a green outline) with the triangle mesh structure superimposed and one of the corner vertices $\mathbf{X}_{i_m}$ depicted.

The Y_{gl}-axis of this virtual camera is directed upwards and the viewing direction of its optical axis is towards $-Z_{gl}$. The projection model is further extended by a *near plane* at $Z_{gl} = -Z_n$ and a *far plane* at $Z_{gl} = -Z_f$ parallel to the image plane, defining the depth of the virtual camera's so-called *view frustum* (see Figure 2.12). These planes are used to limit the distance range along the optical axis at which geometry will be considered for rendering, which is required for technical reasons. This means that everything in front of the near plane and behind the far plane is cut off and will not appear in the resulting image. In contrast to a real camera where the 3D scene is directly projected into the image plane, in OpenGL it is first transformed such that everything within the view frustum is mapped to the so-called *canonical view volume* $[-1, 1]^3 \subset \mathbb{R}^3$. Assuming a symmetrical field of view, the OpenGL view frustum can be defined by specifying a vertical opening angle ρ and its width w and height h in pixels at the near plane, in addition to Z_n and Z_f. The respective transformation can be expressed in linear form as

$$\tilde{\mathbf{X}} = \begin{bmatrix} f(\rho)\frac{w}{h} & 0 & 0 & 0 \\ 0 & f(\rho) & 0 & 0 \\ 0 & 0 & -\frac{Z_f+Z_n}{Z_f-Z_n} & -\frac{2Z_fZ_n}{Z_f-Z_n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} X_{gl} \\ Y_{gl} \\ Z_{gl} \\ 1 \end{bmatrix} = \begin{bmatrix} f(\rho)\frac{w}{h}X_{gl} \\ f(\rho)Y_{gl} \\ \frac{-(Z_f+Z_n)Z_{gl}-2Z_fZ_n}{Z_f-Z_n} \\ -Z_{gl} \end{bmatrix},$$

where the focal length $f(\rho)$ is computed from the opening angle as $f(\rho) = \cot(\rho/2)$.

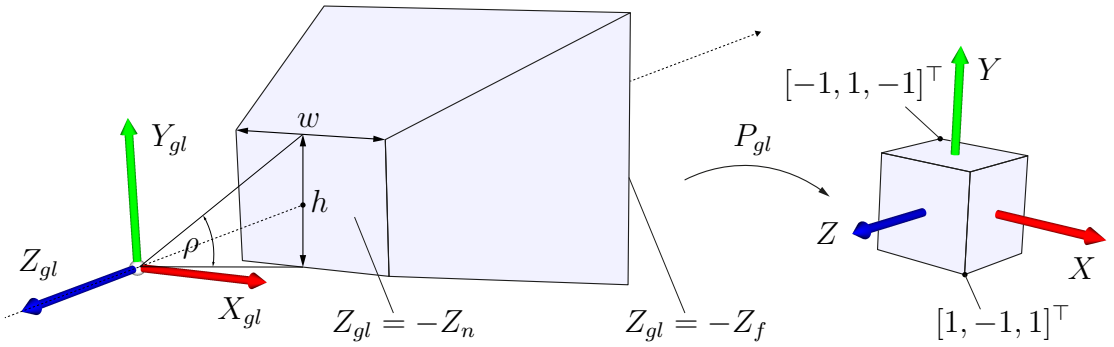


Figure 2.12: A visualization of the virtual OpenGL camera and its view frustum. By applying the perspective transformation P_{gl}, everything within this frustum is transformed into the canonical view volume.

This then leads to the definition of the 4×4 homogeneous perspective projection matrix

$$P_{gl} \doteq \begin{bmatrix} f(\rho)\frac{w}{h} & 0 & 0 & 0 \\ 0 & f(\rho) & 0 & 0 \\ 0 & 0 & -\frac{Z_f+Z_n}{Z_f-Z_n} & -\frac{2Z_fZ_n}{Z_f-Z_n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \in \mathbb{R}^{4\times4},$$

describing the linear transformation of coordinates in the virtual OpenGL camera to so-called *clipping coordinates* $\tilde{\mathbf{X}} = P_{gl}\tilde{\mathbf{X}}_{gl}$. These are internally used to efficiently determine which parts of the scene are cut off (i.e. clipped) for rendering by checking whether they are outside the canonical view volume. By subsequent perspective division and de-homogenization (see (2.29)) of the remaining coordinates the so-called *normalized device coordinates* $\mathbf{X}_{ndc} = \pi(\tilde{\mathbf{X}}) = [X_{ndc}, Y_{ndc}, Z_{ndc}]^\top$ are obtained. Finally, these coordinates are internally mapped to the screen, by the so-called *viewport transformation* as

$$\begin{bmatrix} \frac{w}{2} & 0 & 0 & \frac{w}{2} \\ 0 & \frac{h}{2} & 0 & \frac{h}{2} \\ 0 & 0 & \frac{Z_f-Z_n}{2} & \frac{Z_f+Z_n}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_{ndc} \\ Y_{ndc} \\ Z_{ndc} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{w}{2}X_{ndc} + \frac{w}{2} \\ \frac{h}{2}Y_{ndc} + \frac{h}{2} \\ \frac{Z_f-Z_n}{2}Z_{ndc} + \frac{Z_f+Z_n}{2} \\ 1 \end{bmatrix},$$

after which rasterization to pixels takes place. However, in contrast to the camera images, the 2D OpenGL screen coordinate frame for the rendered images has its

origin in the bottom left corner by default.

Now in order render a 3D model as if it was captured by the real camera, the pinhole camera model derived in Section 2.2.1 must be fused with this OpenGL transformation pipeline. Since the Y- and Z-axes of the real and this virtual camera are by default pointing in opposite directions, it is first required to include a transformation T_{glc} such that $\tilde{\mathbf{X}}_{gl} = T_{glc}\tilde{\mathbf{X}}_c$. It is immediate to see that this transformation is simply a rotation by 180° around the X-axis, that can be expressed in linear form by the 4×4 homogeneous matrix

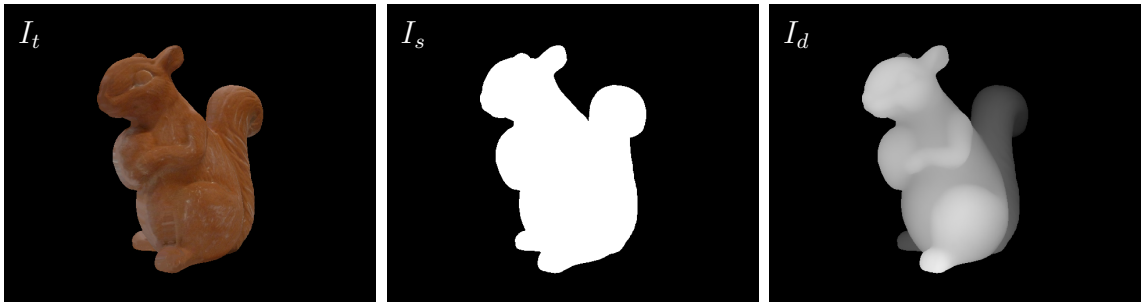
$$T_{glc} \doteq \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{SE}(3),$$

aligning the two axes of the coordinate frames without changing the chirality.

Additionally, with respect to the perspective view frustum, the opening angle and therefore the focal length are given by the intrinsic calibration and cannot be chosen freely. Hence, the known parameters f_x and f_y can directly be used in normalized form as ${}^{2f_x}/_{w}$ and ${}^{2f_y}/_{h}$, respectively, where w and h are now the width and height of the real camera's sensor in pixels. Furthermore, as previously shown, the field of view of a real camera is typically not perfectly symmetrical, meaning that its optical center does not coincide with the center of the image. This is described by the two other linear intrinsic parameters c_x and c_y, which must therefore also be considered for rendering. The perspective transformation to OpenGL clipping coordinates with respect to the intrinsic camera calibration matrix K is thus given by

$$P_{gl}(K) \doteq \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{2f_x}{w} & 0 & 1 - \frac{2c_x}{w} & 0 \\ 0 & \frac{2f_y}{h} & \frac{2c_y}{h} - 1 & 0 \\ 0 & 0 & -\frac{Z_f + Z_n}{Z_f - Z_n} & -\frac{2Z_f Z_n}{Z_f - Z_n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \in \mathbb{R}^{4\times4},$$

where $1 - {}^{2c_x}/_{w}$ and ${}^{2c_y}/_{h} - 1$ are the coordinates of the calibrated optical center normalized with respect to the canonical view volume. Note that the vertical flip applied by the matrix on the left hand side of this multiplication is necessary in order



(a) (b) (c)

Figure 2.13: An example of three different renderings using the same 3D surface model and object pose. If available the model can be rendered using texture information in order to obtain a photo-realistic view of it (a). For a synthetic silhouette projection each triangle is rendered with a constant intensity in front of a plain background, resulting in a binary image (b). In both cases a depth buffer is computed in which the intensity of each pixel corresponds to the distance of the projected surface to the camera. Here, due to the mapping of the depth range, pixels closer to the camera appear brighter in the image (c).

to align the two different 2D image coordinate frames. Given T_{glc}, $P_{gl}(K)$ as well as the objects's pose T_{cm}, the overall transformation from coordinates specified in the model frame $\mathbf{X}_m$ to clipping coordinates $\mathbf{X}$ required by OpenGL can be expressed by

$$\tilde{\mathbf{X}} = P_{gl}(K)T_{glc}T_{cm}\tilde{\mathbf{X}}_m \in \mathbb{R}^4,$$

with respect to the intrinsic parameters of the real camera and object's pose specified in relation to it.

Usually the images computed on the graphics card are directly displayed on the screen without being transferred back to the CPU. However, for the approaches presented in this thesis it is required to further process such rendered images along with the real camera images computationally without necessarily ever showing them on the screen. For this OpenGL provides the functionality of offscreen rendering to a so-called *frame buffer*. It can be used to access the intensities of the rendered images as well as the depth per pixel that is stored in the so-called *depth buffer*. In this work two types of images are being rendered, namely color images of textured models denoted by $I_t : \Omega \to [0, 255]^3 \subset \mathbb{Z}^3$ (see Figure 2.13 (a)) and silhouette images denoted by $I_s : \Omega \to [0, 255] \subset \mathbb{Z}$ (see Figure 2.13 (b)). The depth buffer

is denoted by $I_d : \Omega \to [0,1] \subset \mathbb{R}$ (see Figure 2.13 (c)), in which the intensity of each pixel corresponds to the distance of the projected surface to the camera within the range $[Z_n, Z_f]$ mapped to $[1,0]$. Thus, given the depth buffer and using (2.31), the 3D surface point $\mathbf{X}_c$ corresponding to each pixel $\mathbf{x}$ can be determined via back-projection as

$$\mathbf{X}_c(I_d, \mathbf{x}) = D(I_d, \mathbf{x}) K^{-1} \tilde{\mathbf{x}}, \quad \forall \mathbf{x} \in \Omega \text{ where } I_d(\mathbf{x}) > 0, \tag{2.33}$$

with

$$D(I_d, \mathbf{x}) \doteq \frac{2 Z_n Z_f}{Z_f + Z_n - (2 I_d(\mathbf{x}) - 1)(Z_f - Z_n)} = \frac{Z_n Z_f}{Z_f - I_d(\mathbf{x})(Z_f - Z_n)} = Z_c,$$

mapping the depth buffer value $I_d(\mathbf{x})$ back to the range $[Z_n, Z_f]$.

2.3 Nonlinear Parametric Optimization

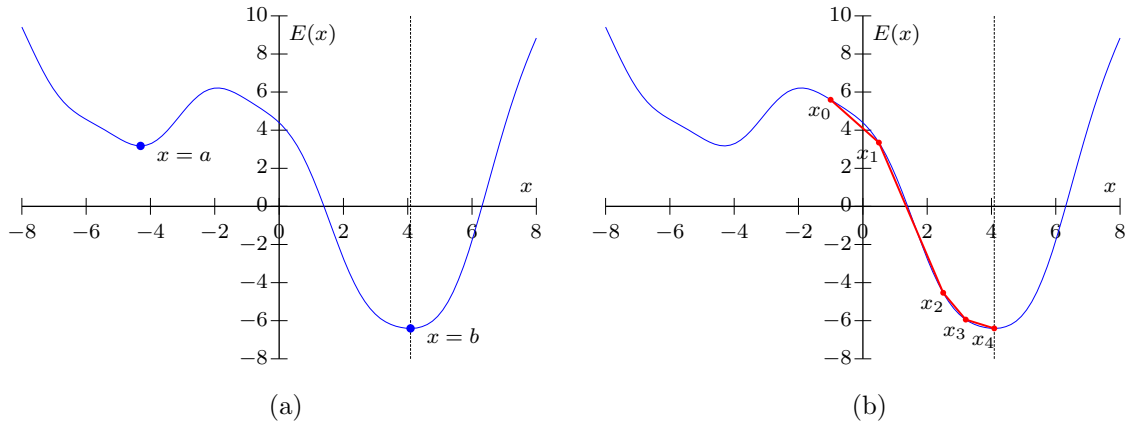
Monocular pose estimation can be seen as the optimization problem of fitting a synthetic projection of a given 3D model into measured observations in form of a 2D camera image where the only free parameters are the object's pose. Here, the quality of an estimated pose $T \in \mathbb{SE}(3)$ is in general measured by a so-called nonlinear *cost* or *energy function* $E : \mathbb{SE}(3) \mapsto \mathbb{R}$

$$E(T) = \sum_{i=0}^{n-1} f\left(I, \mathbf{x}_i(T)\right), \tag{2.34}$$

as a sum over n 2D projections $\mathbf{x}_i(T)$ each computed from a corresponding 3D model point $\mathbf{X}_{i_m}$ directly depending on T as given in (2.32). Furthermore, $f(\cdot)$ is a function that measures the fit of a single projection $\mathbf{x}_i(T)$ with respect to the observed image data I. This leads to the optimization problem

$$T_{cm} = \arg\min_T \; E(T), \quad T \in \mathbb{SE}(3),$$

where the sought rigid body transform T_{cm} is ideally the global minimum of E.



(a) (b)

Figure 2.14: An example of a non-convex 1D cost function $E(x)$ with a local minimum at $x = a$ and a global minimum $x = b$, indicated by a dashed vertical line (a). If initialized in the vicinity of b at a starting location x_0 the numerical optimization updates the parameter in iterative steps $x_1, \ldots, x_4$ towards the global minimum until eventually reaching it (b).

Assuming that E is continuously differentiable, a minimum can efficiently be determined using an iterative numerical optimization approach based on gradient information. However, a nonlinear cost function is potentially non-convex, meaning it can have multiple local minima (see Figure 2.14 (a)) and is not even guaranteed to have a distinct global minimum at all. In oder to converge to a global minimum, the optimization must be initialized somewhere in its vicinity. In this work this initialization in form of a rough pose estimate denoted by T_{cm}^0 is either obtained manually, by a preceding pose detection step, or is given by the pose estimated from the previous frame as $T_{cm}^0(t_i) = T_{cm}(t_{i-1})$ during pose tracking.

Assuming T_{cm}^0 is known, numerical minimization of E can be used in order to compute the more accurate solution T_{cm}. The difference between the initial pose T_{cm}^0 and the final refined pose T_{cm} is given by

$$T_{cm}^{\Delta} = T_{cm} \left(T_{cm}^0\right)^{-1} \in \mathbb{SE}(3),$$

with $T_{cm} = T_{cm}^{\Delta} T_{cm}^0$. However, as explained in Section 2.1.1, parameterizing this optimization directly with the 4×4 homogeneous matrix representation is not the best choice, due to the redundancy in its sixteen entries and the constraints imposed

by $\mathbb{SE}(3)$. Therefore, in this work the pose difference is modeled using rigid body motion with twist parametrization as

$$T_{cm} = \exp(\hat{\xi})T_{cm}^0, \quad \text{with} \quad \hat{\xi} = \log(T_{cm}^{\Delta}) \in \mathfrak{se}(3),$$

where $\hat{\xi}$ is parametrized by its unconstrained twist coordinates $\xi \in \mathbb{R}^6$. The projection of the 2D coordinates is then modeled accordingly as

$$\mathbf{x}(\xi) = \pi(K(\exp(\hat{\xi})T_{cm}^0\tilde{\mathbf{X}}_m)_{3\times 1}),$$

where the initial rigid body transform T_{cm}^0 is assumed to be fixed, leading to a corresponding alternative definition of the energy function

$$E(\xi) = \sum_{i=0}^{n-1} f\left(I, \mathbf{x}_i(\xi)\right), \quad E : \mathbb{R}^6 \mapsto \mathbb{R}, \tag{2.35}$$

with respect to the six twist coordinates. The general idea of iteratively minimizing a nonlinear function is very simple. Starting at a location $\xi = \xi_0 = \mathbf{0}$, ξ is successively updated to $\xi_1, \xi_2, \ldots$, such that the value of $E(\xi)$ decreases with each step i.e. $E(\xi_{t+1}) \leq E(\xi_t)$ (see Figure 2.14 (b)). These iterations are recursively related by

$$\xi_{t+1} = \xi_t + \Delta\xi_t,$$

where $\Delta\xi_t$ is the so-called *update step*. Thus, in each iteration this step should ideally be chosen, such that it moves the optimization parameters ξ towards the minimum in the direction in which the energy decreases the fastest for each dimension.

In the special case when optimizing the energy function (2.34) with respect to twist coordinates however, such an update step has a geometric interpretation, being the minimal representation of a rigid body transform. For an appropriate pose update, it must therefore be directly mapped to its corresponding group element in $\mathbb{SE}(3)$ and applied to the previous rigid body transform estimate by matrix multiplication as

$$T_{cm}^{t+1} \leftarrow \exp(\Delta\hat{\xi}_t)T_{cm}^t, \tag{2.36}$$

which sets $\xi_{t+1} = \xi_0 = \mathbf{0}$ after each iteration. Starting at the initial estimate T_{cm}^0,

the pose in the current iteration is thus computed as

$$T_{cm}^{t} = \exp(\Delta\hat{\xi}_{t-1}) \exp(\Delta\hat{\xi}_{t-2}) \dots \exp(\Delta\hat{\xi}_{0}) T_{cm}^{0},$$

being a concatenation of the piecewise optimized rigid body motion. When either the optimization has converged or after a prescribed number of iterations, the final pose result is denoted by T_{cm}. In the following sections, different popular techniques for computing the update step $\Delta\xi_t$ will be introduced, that are used within the rest of this thesis, depending on the particular cost functions.

2.3.1 Gradient Descent

Without some special knowledge about the nonlinear function E that could be used to speed up the optimization, a broadly applicable approach to iteratively minimize it is a first-order *gradient descent* (also called *method of steepest descent*). The idea is to locally approximate the cost functiom $E(\xi)$ by a first-order Taylor Expansion in each iteration as

$$E(\xi_0 + \Delta\xi) \approx E(\xi_0) + \nabla E(\xi_0)^{\top} \Delta\xi, \tag{2.37}$$

with the gradient

$$\nabla E(\xi) = \left[\frac{\partial E}{\partial \omega_1}, \frac{\partial E}{\partial \omega_2}, \frac{\partial E}{\partial \omega_3}, \frac{\partial E}{\partial v_1}, \frac{\partial E}{\partial v_2}, \frac{\partial E}{\partial v_3} \right]^{\top} \in \mathbb{R}^6,$$

being the vector of first-order partial derivatives. Accordingly, the gradient of (2.35) at $\xi_0 = \mathbf{0}$ can be written as

$$\nabla E(\xi_0) = \sum_{i=0}^{n-1} J(\mathbf{x}_i, \xi_0)^{\top} \in \mathbb{R}^6,$$

where

$$J(\mathbf{x}, \xi_0) = \left. \frac{\partial f\left(I, \mathbf{x}(\xi)\right)}{\partial \xi} \right|_{\xi_0} = \left. \frac{\partial f}{\partial \mathbf{x}(\xi)} \frac{\partial \mathbf{x}(\xi)}{\partial \xi} \right|_{\xi_0},$$

is the 1×6 Jacobian term per projected coordinate $\mathbf{x}_i$. While the first part $\partial f / \partial \mathbf{x}(\xi) \in \mathbb{R}^{1\times2}$ depends on the respective fitting function, the derivatives of the projected 2D

coordinates $\mathbf{x}(\xi)$ on the right hand side can be defined independently as

$$\frac{\partial \mathbf{x}(\xi)}{\partial \xi} = \frac{\partial \pi}{\partial K(\exp(\hat{\xi}) T_{cm}^{t} \tilde{\mathbf{X}}_m)_{3\times 1}} \frac{\partial K(\exp(\hat{\xi}) T_{cm}^{t} \tilde{\mathbf{X}}_m)_{3\times 1}}{\partial \xi} \in \mathbb{R}^{2\times 6}, \tag{2.38}$$

with respect to the twist coordinates. Assuming small motion, the computation of the derivatives of $\exp(\hat{\xi})$ (2.25) can be simplified by linearization of the matrix exponential in each iteration as

$$\exp(\hat{\xi}) \approx \mathbf{I}_{4\times 4} + \hat{\xi} = \begin{bmatrix} 1 & -\omega_3 & \omega_2 & v_1 \\ \omega_3 & 1 & -\omega_1 & v_2 \\ -\omega_2 & \omega_1 & 1 & v_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4}. \tag{2.39}$$

Given this linearization, the Jacobian matrix of (2.38) can be explicitly computed as

$$\left.\frac{\partial \mathbf{x}(\xi)}{\partial \xi}\right|_{\xi_0} = \begin{bmatrix} \frac{f_x}{Z_c} & 0 & -\frac{X_c f_x}{Z_c^2} \\ 0 & \frac{f_y}{Z_c} & -\frac{Y_c f_y}{Z_c^2} \end{bmatrix} \begin{bmatrix} 0 & Z_c & -Y_c & 1 & 0 & 0 \\ -Z_c & 0 & X_c & 0 & 1 & 0 \\ Y_c & -X_c & 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 6}, \tag{2.40}$$

with $[X_c, Y_c, Z_c]^\top = (T_{cm}^{t} \tilde{\mathbf{X}}_m)_{3\times 1}$.

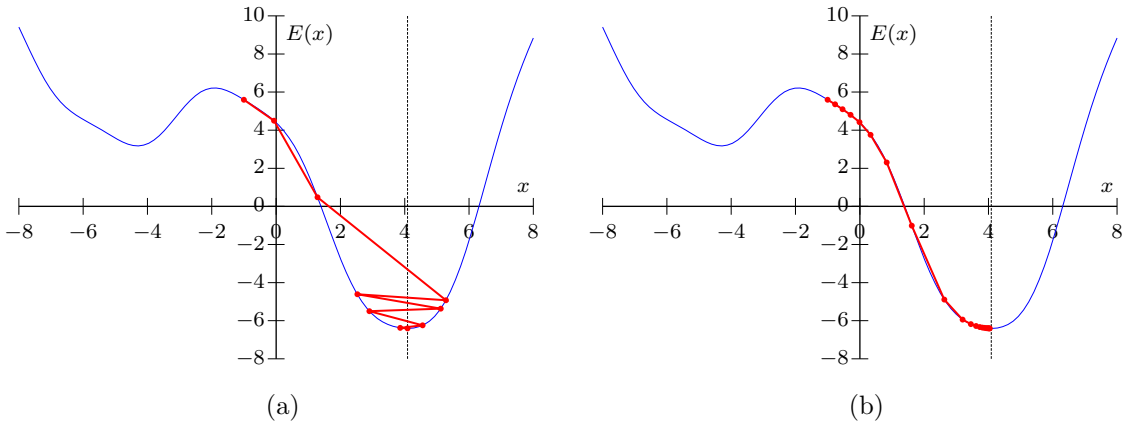
The corresponding update step within a gradient descent is obtained from the optimality condition

$$\frac{dE(\xi_0 + \Delta\xi)}{d\Delta\xi} \approx \nabla E(\xi_0)^\top = 0,$$

which implies that the derivative of the right side of (2.37) is assumed to vanish at the extremum. Furthermore, given that the gradient $\nabla E(\xi)$ of a function always yields the direction of steepest ascent, here the update step is simply chosen as its negative direction

$$\Delta\xi_t = -\lambda_t \nabla E(\xi_0) = -\lambda_t \sum_{i=0}^{n-1} J(\mathbf{x}_i, \xi_0)^\top,$$

scaled by a so-called *step size* $\lambda_t \in \mathbb{R}_+$ that can be chosen differently in each iteration.



(a) (b)

Figure 2.15: Examples of a 1D gradient descent with different step sizes. In case of the step size being too large the optimization tends to zigzag around the minimum, which potentially leads to many iterations or no convergence at all (a). To ensure convergence often a sufficiently small fixed step size is chosen in combination with running many iterations (b).

If initialized close enough to the desired minimum, the gradient descent potentially converges to it. Whether it converges or not, here highly depends on the right choice of the step size λ_t at each iteration. For this no universal scheme exists to determine an optimal fixed step size [PTVF07]. In cases where it is chosen too large the iterations tend to *zig zag* around the minimum leading to many iterations or, in the worst case, no convergence at all (see Figure 2.15 (a)). The latter can be encountered by decreasing the step size with each iteration. However, this strategy could also prevent the minimum from being reached at all. Therefore, one of the most common approaches is to run many iterations with a fixed small step size in order to ensure convergence (see Figure 2.15 (b)).

The most commonly used strategy to dynamically find an optimal step size is to perform a numerical 1D so-called *line-search* along the line in direction of the update step, such that

$$\lambda_t = \arg\min_{\lambda} E(\xi_{t-1} + \lambda \Delta\xi_t),$$

which in case of pose optimization, has to be rewritten as

$$\lambda_t = \arg\min_{\lambda} E(\exp(\lambda \Delta\xi_t) T_{cm}^t),$$

according to (2.36). This, however, requires evaluating the cost function at multiple potential step sizes. Whether such a line-search is an efficient choice with regard to the runtime of the optimization algorithm thus depends on how costly it is to evaluate the function in relation to the computation of its derivatives.

Beyond the simple gradient descent presented here, there are other more advanced first-order methods, e.g. *conjugate gradients* [PTVF07]. Depending on the energy function they can be used in order to improve the direction of the update steps and thereby the convergence rate of the iterative optimization. However, these methods also incorporate line-searches at some point and are not fundamentally different from a regular gradient descent. Since none of these alternative first-order optimization stategies are utilized in the remainder of this work, they will also not be further elaborated at this point.

2.3.2 Newton Methods

Despite the advantages of being widely applicable to differentiable multidimensional energy functions, a gradient descent generally does not have the best convergence properties among other numerical optimization algorithms. The main reason for this is that the piecewise linearization of $E(\xi)$ used within a gradient descent is a very rough approximation. In the previous section this was done because no special knowledge about the cost function was assumed to be available. However, this is not always the case. For example, if it is valid to assume that the cost function locally resembles a quadratic function around the minimum and is at least twice continuously differentiable, so-called *Newton methods* can be applied to speed up the optimization.

Similar to the idea of the gradient descent, for Newton methods the cost function E is approximated by a second-order Taylor expansion

$$E(\xi_0 + \Delta\xi) \approx E(\xi_0) + \nabla E(\xi_0)^\top \Delta\xi + \frac{1}{2}\Delta\xi^\top \nabla^2 E(\xi_0)\Delta\xi,$$

that additionally models the local curvature. Here, $\nabla^2 E(\xi)$ is the so-called *Hessian*

matrix

$$\nabla^2 E(\xi) = \begin{bmatrix} \frac{\partial^2 E}{\partial \omega_1 \partial \omega_1} & \frac{\partial^2 E}{\partial \omega_1 \partial \omega_2} & \frac{\partial^2 E}{\partial \omega_1 \partial \omega_3} & \frac{\partial^2 E}{\partial \omega_1 \partial v_1} & \frac{\partial^2 E}{\partial \omega_1 \partial v_2} & \frac{\partial E^2}{\partial \omega_1 \partial v_3} \\ \frac{\partial^2 E}{\partial \omega_2 \partial \omega_1} & \frac{\partial^2 E}{\partial \omega_2 \partial \omega_2} & \frac{\partial^2 E}{\partial \omega_2 \partial \omega_3} & \frac{\partial^2 E}{\partial \omega_2 \partial v_1} & \frac{\partial^2 E}{\partial \omega_2 \partial v_2} & \frac{\partial E^2}{\partial \omega_2 \partial v_3} \\ \frac{\partial^2 E}{\partial \omega_3 \partial \omega_1} & \frac{\partial^2 E}{\partial \omega_3 \partial \omega_2} & \frac{\partial^2 E}{\partial \omega_3 \partial \omega_3} & \frac{\partial^2 E}{\partial \omega_3 \partial v_1} & \frac{\partial^2 E}{\partial \omega_3 \partial v_2} & \frac{\partial E^2}{\partial \omega_3 \partial v_3} \\ \frac{\partial^2 E}{\partial v_1 \partial \omega_1} & \frac{\partial^2 E}{\partial v_1 \partial \omega_2} & \frac{\partial^2 E}{\partial v_1 \partial \omega_3} & \frac{\partial^2 E}{\partial v_1 \partial v_1} & \frac{\partial^2 E}{\partial v_1 \partial v_2} & \frac{\partial E^2}{\partial v_1 \partial v_3} \\ \frac{\partial^2 E}{\partial v_2 \partial \omega_1} & \frac{\partial^2 E}{\partial v_2 \partial \omega_2} & \frac{\partial^2 E}{\partial v_2 \partial \omega_3} & \frac{\partial^2 E}{\partial v_2 \partial v_1} & \frac{\partial^2 E}{\partial v_2 \partial v_2} & \frac{\partial E^2}{\partial v_2 \partial v_3} \\ \frac{\partial^2 E}{\partial v_3 \partial \omega_1} & \frac{\partial^2 E}{\partial v_3 \partial \omega_2} & \frac{\partial^2 E}{\partial v_3 \partial \omega_3} & \frac{\partial^2 E}{\partial v_3 \partial v_1} & \frac{\partial^2 E}{\partial v_3 \partial v_2} & \frac{\partial E^2}{\partial v_3 \partial v_3} \end{bmatrix} \in \mathbb{R}^{6\times 6}, \tag{2.41}$$

computed from second-order partial derivatives of E in each dimension. Due to the optimality condition

$$\frac{dE(\xi_0 + \Delta\xi)}{d\Delta\xi} \approx \nabla E(\xi_0)^\top + \nabla^2 E(\xi_0)\Delta\xi = 0,$$

the update step within a Newton iteration is given by

$$\Delta\xi_t = -\left(\nabla^2 E(\xi_0)\right)^{-1} \nabla E(\xi_0),$$

or in a more general form by

$$\Delta\xi_t = -\lambda_t \left(\nabla^2 E(\xi_0)\right)^{-1} \nabla E(\xi_0),$$

where again a step size parameter $\lambda_t \in [0, 1]$ is included. A geometric interpretation of the Newton step is that it moves the parameters towards (for $\lambda_t < 1$) or exactly to the extremum (for $\lambda_t = 1$) of a multidimensional paraboloid fitted locally to the energy function in each iteration. When initialized in a convex basin of E such a second-order optimization often converges to the minimum of E in significantly less iterations than a regular first-order gradient descent (see Figure 2.16).

However, in practice obtaining the second-order derivatives of E can be very computationally demanding. They can also become numerically unstable, as often reported in literature, since they are typically very small. For a Newton method to be successful, it is furthermore assumed that the whole optimization takes place in a convex part of the cost function. This implies that the Hessian matrix must be positive definite at each location along the path. For cost functions where this cannot be guaranteed, so-called *quasi-Newton methods* exist. The two most popular

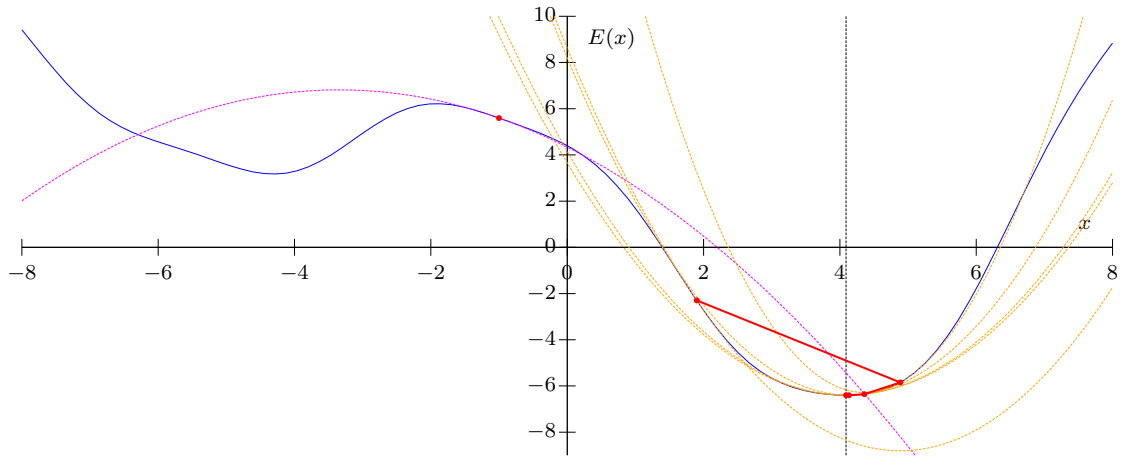


Figure 2.16: An example of a 1D Newton minimization. When initialized in the convex basin around the global minimum the parameter is successfully updated by moving it to the extremum of a parabola fitted locally to the current location (here shown in orange). When initialized in a concave part of the cost function the second-order derivative is negative, meaning that the extremum of the fitted parabola (here shown in magenta) would move the parameter away from the global minimum.

are the so-called *DFP* (Davidon-Fletcher-Powell) and the *BFGS* (Broyden-Fletcher-Goldfarb-Shanno) algorithms [PTVF07]. These approximate $\nabla^2 E(\xi_0)$ iteratively by a matrix as close as possible to the original Hessian matrix that is always positive definite. Ideally this approximation does not involve the explicit computation of any second-order derivatives. Thus, the aforementioned drawbacks related to them are avoided. However, DFP as well as BFGS also incorporate line-searches, and thus typically require a large number of function evaluations.

2.3.3 The Gauß-Newton Method

In cases where the fitting of the projected model in the image can be described in form of a quadratic cost function, meaning $f(\cdot) = r(\cdot)^2$, the energy function turns into

$$E(\xi) = \frac{1}{2} \sum_{i=0}^{n-1} r\left(I, \mathbf{x}_i(\xi)\right)^2, \quad E : \mathbb{R}^6 \mapsto \mathbb{R}, \tag{2.42}$$

where $r(\cdot)$ is called the *residuum*. For such least-squares problems, typically a *Gauß-Newton* strategy can be applied, being an efficient second-order method that does neither involve the computation of second-order derivatives nor any kind of line-search scheme.

For this, computing the gradient of (2.42) results in

$$\nabla E(\xi) = \frac{1}{2}\sum_{i=0}^{n-1}\left(\frac{\partial r\left(I, \mathbf{x}_i(\xi)\right)^2}{\partial \xi}\right)^\top = \sum_{i=0}^{n-1}\left(r\left(I, \mathbf{x}_i(\xi)\right)\frac{\partial r\left(I, \mathbf{x}_i(\xi)\right)}{\partial \xi}\right)^\top, \tag{2.43}$$

and the Hessian matrix is accordingly given by

$$\nabla^2 E(\xi) = \sum_{i=0}^{n-1}\left(\frac{\partial r\left(I, \mathbf{x}_i(\xi)\right)}{\partial \xi}\right)^\top \frac{\partial r\left(I, \mathbf{x}_i(\xi)\right)}{\partial \xi} + r\left(I, \mathbf{x}_i(\xi)\right)\frac{\partial^2 r\left(I, \mathbf{x}_i(\xi)\right)}{\partial \xi^2}. \tag{2.44}$$

The essential assumption made within the Gauß-Newton algorithm is that the residuum r is linear in the parameters ξ (in which case $\partial^2 r/\partial \xi^2 \approx 0$). This allows dropping the second-order derivative in (2.44), giving

$$\nabla^2 E(\xi) \approx \sum_{i=0}^{n-1}\left(\frac{\partial r\left(I, \mathbf{x}_i(\xi)\right)}{\partial \xi}\right)^\top \frac{\partial r\left(I, \mathbf{x}_i(\xi)\right)}{\partial \xi}, \tag{2.45}$$

which then only requires the computation of first-order derivatives and is guaranteed to be positive semi-definite. As with the previous methods, the cost function is then locally approximated by inserting (2.43) and (2.45) into a Taylor expansion, giving

$$E(\xi_0 + \Delta\xi) \approx E(\xi_0) + \sum_{i=0}^{n-1} r(I, \mathbf{x}_i) J_r(\mathbf{x}_i, \xi_0)\Delta\xi + \frac{1}{2}\sum_{i=0}^{n-1}\Delta\xi^\top J_r(\mathbf{x}_i, \xi_0)^\top J_r(\mathbf{x}_i, \xi_0)\Delta\xi,$$

with

$$J_r(\mathbf{x}, \xi_0) = \left.\frac{\partial r\left(I, \mathbf{x}(\xi)\right)}{\partial \xi}\right|_{\xi_0} = \left.\frac{\partial r}{\partial \mathbf{x}(\xi)}\frac{\partial \mathbf{x}(\xi)}{\partial \xi}\right|_{\xi_0},$$

being again the 1×6 Jacobi vector of the residuum per projected coordinate $\mathbf{x}$. Here, the computation of the derivatives $\partial \mathbf{x}(\xi)/\partial \xi$ in the Jacobi terms can again be simplified as (2.40) by exploiting the linearization of the matrix exponential (2.39). As before, due to the optimality condition, the update step is then accordingly given

by

$$\Delta\xi_t = -\left(\sum_{i=0}^{n-1} J_r(\mathbf{x}_i,\xi_0)^\top J_r(\mathbf{x}_i,\xi_0)\right)^{-1} \sum_{i=0}^{n-1} r(I,\mathbf{x}_i) J_r(\mathbf{x}_i,\xi_0)^\top,$$

which is in practice efficiently solved using a Cholesky decomposition of the summed Hessian approximation.

2.3.4 The Levenberg-Marquardt Method

Another optimization strategy that is also only applicable to least-squares problems is the so-called *Levenberg-Marquardt* method [Lev44, Mar63]. It is a damped modification of the Gauß-Newton method with a fixed step size that is assumed to be 1 and a different approximation of the Hessian matrix. Here $\nabla^2 E(\xi_0)$ is approximated either by

$$\nabla^2 E(\xi_0) \approx \left(\sum_{i=0}^{n-1} J_r(\mathbf{x}_i,\xi_0)^\top J_r(\mathbf{x}_i,\xi_0)\right) + \lambda_t \mathbf{I}_{6\times 6},$$

with the regularization term $\mathbf{I}_{6\times 6}$, as originally suggested by Levenberg [Lev44] or

$$\nabla^2 E(\xi_0) \approx \left(\sum_{i=0}^{n-1} J_r(\mathbf{x}_i,\xi_0)^\top J_r(\mathbf{x}_i,\xi_0)\right) + \lambda_t \operatorname{diag}\left(\sum_{i=0}^{n-1} J_r(\mathbf{x}_i,\xi_0)^\top J_r(\mathbf{x}_i,\xi_0)\right),$$

regularized with diag $\left(\sum_{i=0}^{n-1} J_r(\mathbf{x}_i,\xi_0)^\top J_r(\mathbf{x}_i,\xi_0)\right)$, being a more adaptive componentwise damping as suggested later by Marquardt [Mar63]. Here $\operatorname{diag}(A)$ forms the diagonal matrix from any square matrix $A \in \mathbb{R}^{m\times m}$ as

$$\operatorname{diag}\left(\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mm} \end{bmatrix}\right) \doteq \begin{bmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{mm} \end{bmatrix}.$$

In both cases $\lambda_t > 0$ now acts as a damping parameter within the approximation of the hessian matrix. It thereby creates a hybrid between the Gauß-Newton method (when $\lambda_t = 0$) and a gradient descent with step size $1/\lambda_t$ (for $\lambda_t \to \infty$). Assuming that the optimization is potentially starting in a concave part of the cost func-

tion this damping parameter λ_t is commonly initialized by a small value such that the update step is close to a gradient descent. While approaching the minimum λ_t is typically decreased in order to shift the update strategy more towards the Gauß-Newton method. Which of the above approximations works better in practice however depends on the respective cost function and must usually be evaluated empirically.

3 Point-based Pose Estimation

This chapter deals with the general concept of estimating the pose of an object from 2D-to-3D point correspondences. It starts by introducing the basic idea of point-based pose estimation (Section 3.1) and then moves on by explaining both active (Section 3.2) and passive (Section 3.3) strategies that are commonly applied in this context. Here, the main focus lies on Section 3.2, where a novel active approach is presented in detail that was developed within the course of this dissertation. The chapter concludes with a summary (Section 3.4) of the presented point-based approaches.

3.1 Outline

Point-based pose estimation from a single camera image is commonly known as the well-studied *Perspective-n-Point* (PnP) problem (see e.g. [ZKS+13, LMF09, SP08, MLF07]). In this context objects are represented by an abstract model consisting of a discrete set of n 3D Points $\mathbf{X}_{i_m}$, $i = 0, \ldots, n-1$. Compared to a dense surface model this representation is usually chosen to be relatively sparse in order to reduce the computational complexity of the corresponding pose estimation algorithm. The idea is to establish a set of 2D-to-3D correspondences $\mathbf{x}'_i \leftrightarrow \mathbf{X}_{i_m}$, where $\mathbf{x}'_i$ are 2D

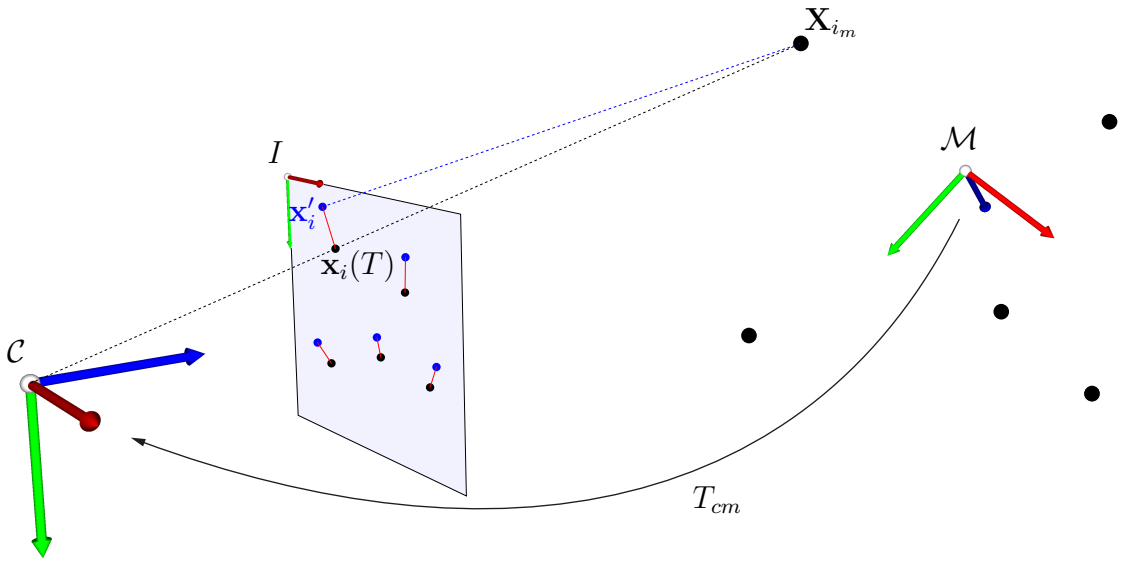


Figure 3.1: An overview of the monocular point-based pose estimation scenario. Given an hypothesis T for the pose T_{cm}, the model points are transformed into the camera coordinate frame and projected into the image plane to $\mathbf{x}_i(T)$. Here, the projection error per point (shown as red lines) is computed as the distance to each corresponding measured projection $\mathbf{x}'_i$ (shown as blue dots) in the image plane.

points extracted from the camera image, that are assumed to be the true projections of the corresponding 3D model points $\mathbf{X}_{i_m}$ (see Figure 3.1). Given this relation, the quality of a pose hypothesis $T \in \mathbb{SE}(3)$ is determined by projecting the 3D model points to 2D points $\mathbf{x}_i(T)$ in the image plane using (2.32) and comparing them to corresponding measured 2D points $\mathbf{x}'_i$. This is usually formulated as a nonlinear least-squares problem given by the energy function

$$E_p(T) = \frac{1}{2} \sum_{i=0}^{n-1} \|\mathbf{x}'_i - \mathbf{x}_i(T)\|_2^2, \tag{3.1}$$

which describes the pose fit as a sum of the squared distances between the projected and the measured 2D point coordinates in the image. Given an initial pose estimate T_{cm}^0 this energy can in general be efficiently minimized by e.g. one of the iterative methods presented in Section 2.3 with respect to twist parameters. This optimization step is usually applied as so-called *pose refinement* of the initial guess in order to obtain a more accurate solution.

In theory a 6DOF pose can be uniquely determined from a minimum number of four 2D-to-3D point correspondences. However, it has also been shown that a linear solution to the so-called *Perspective-three-Point* (P3P) problem (i.e. where $n = 3$) exists for any subset of three linearly independent 2D-to-3D point correspondences, that in general yields four possible unique pose configurations [HLON94, Gru41]. Thus in practice, a robust, simple and therefore the most widely used approach to obtain the initial pose estimate is to solve the PnP problem from a minimal set of four correspondences in two steps. First, four potential pose configurations are calculated by solving P3P, using three suitable correspondences. The remaining ambiguity is then resolved in a second and final step, where the projection error of a fourth point correspondence is computed for all four pose hypotheses, by assuming that this error is minimal for the correct pose. Nowadays P3P is a well studied problem for which a variety of robust and optimized solutions have been proposed [KR17, MZ14, KSS11, NS07, GHTC03] of which some are publicly available as part of open source software libraries[1].

Having introduced more or less generally applicable solutions to pose initialization and optimization, the remaining steps that have not yet been discussed are the extraction of the locations $\mathbf{x}'_i$ in an image and how the $\mathbf{x}'_i \leftrightarrow \mathbf{X}_{i_m}$ correspondences can be determined. In general, the point extraction step is performed by using dedicated 2D image processing methods, chosen with regard to the objects of interest, the utilized image sensor and the overall capture scenario [LF05]. Compared to this, determining the corresponding 3D point in a set of extracted 2D locations is a much harder problem for which very different solutions both software and hardware have been developed in the past. Therefore to date, a large variety of point-based pose estimation approaches have been proposed that mostly vary in terms of their strategies to solve these two core problems. Here, the respective solutions mostly depend on whether a marker is used and whether it is an active or a passive approach. While a complete overview is beyond the scope of this work, the following Sections 3.2 and 3.3 will elaborate on a novel active marker-based approach as well as popular point-based approaches for passive marker-less pose estimation, being the two extreme cases in this wide spectrum.

[1] Open source P3P implementations are available within e.g. OpenCV: https://opencv.org/ and the TooN library: https://www.edwardrosten.com/cvd/toon/html-user/

3.2 An Active Marker-based Approach

This section in most parts describes the point-based pose estimation system that was originally presented in our publication [TSSS15]. There we proposed novel a method for high-speed and robust tracking (*HSRM-Tracking*) of active markers. This algorithm robustly and accurately performs 6DOF tracking by detection of multiple markers at several hundred hertz, i.e. full frame rate of current consumer high-speed cameras. For this, we have developed two novel, nearly co-planar marker patterns that can be identified without initialization or incremental tracking. These patterns also encode a unique ID to identify and distinguish between different markers. The individual markers are calibrated semi-automatically within a few seconds using the same camera. Thus, no time-consuming and error-prone manual measurement is needed. The core novelty of HSRM-Tracking is a sophisticated pose optimization strategy that robustly resolves common pose ambiguities even at large distances to the camera by exploiting the minimal spatial structure of the markers. This allows to measure the pose of each individual marker with vastly increased accuracy in a much larger volume compared to previous purely monocular pose estimation systems.

3.2.1 Introduction and Background

Active pose estimation approaches overcome most of the limitations related to passive features (see Section 3.3), since they are less dependent on the given lighting conditions. This is commonly achieved by using active infrared light in conjunction with a corresponding infrared-pass filter attached to the camera. Thus ideally only the employed fiducials appear visible to the camera which thereby simplifies the image processing effort needed to detect and localize them in the images.

Active optical systems can be divided further into two categories. Firstly, so-called *active camera systems*, where the additional light source is attached to the camera and the fiducials are retro-reflective and secondly, so-called *active marker systems*, with light emitting devices such as light emitting diodes (LEDs) attached to the fiducials. While active camera systems utilize very lightweight targets that do not

require their own power source, active marker systems usually provide a larger total working volume and higher precision because here the light only has to travel one way instead of two.

The individual lights or retro-reflective tags each marker is composed of typically appear as mostly identical white blobs in the images (see Section 3.2.3). Therefore, one of the main tasks of active systems is to identify and distinguish between these blobs in order to obtain the required 2D-to-3D point correspondences. Active camera systems are mainly used in multi-camera stereo setups such as the commercial motion capturing systems Vicon[2] and Optitrack[3]. Here epipolar constraints and triangulation can be used to help identifing the tags. For monocular active marker systems e.g. [NF05] presented a technique for encoding individual LED lights by different amplitude modulation (i.e. blinking), used in an initialization step. After initialization the LEDs are tracked incrementally from frame to frame. Over the last few years active marker system have gained a lot of popularity of virtual reality (VR) where they are used in order to estimate the pose of head sets and hand-held controllers [SSV+17, AGHWK16]. Here these *blinking* initialization approaches have made their way into popular commercial virtual reality products such as the Oculus Rift[4] or the OSVR[5]. However, such techniques require specialized hardware that allows to synchronize the frame acquisition with the blinking of the LEDs. They also come with the downside of having to re-initialize the system after every tracking loss.

Another major choice of design for active marker approaches is how the LEDs that form a marker should be spatially arranged in order to be recognizable from as many perspectives as possible. Essentially, such markers must consist of a minimum of four LEDs, since each LED provides at most one 2D-to-3D point correspondence for solving the PnP problem. In order to guarantee that all four LEDs are visible from as many perspectives as possible, a straightforward approach is to place them within a common plane (e.g. a square or a rectangle). However in [SP06] it was shown that monocular pose estimation from such purely co-planar patterns is prone to *pose*

[2]The Vicon motion capturing system: https://www.vicon.com/
[3]The Optitrack motion capturing system: http://www.optitrack.com/
[4]The Oculus Rift virtual reality system: https://www.oculus.com/
[5]An open source virtual reality system: http://www.osvr.org/

ambiguities (see Section 3.2.5), regardless of the number of point correspondences. This can be avoided by placing one of the four LEDs outside of the plane defined by the others which thereby adds some spatial extension to the marker. In general it holds that the pose estimation robustness increases with the number of points and their distribution in all three dimensions. However, markers with a spatial structure are likely to yield self-occlusions when projected into an image, meaning it cannot be guaranteed that all LEDs are visible at the same time. In order to still obtain at least four correspondences, this problem can in principle be countered by increasing the number of LEDs. However, this in return generally complicates the identification problem. Thus an ideal LED marker design should be a compromise between the minimum number of LEDs and the least required spatial structure in order to achieve robust pose estimates from a maximum range of perspectives.

An approach that is the most related to HSRM-Tracking was presented in [FMSS14]. This system can estimate the pose of a single marker composed of four or five arbitrarily arranged LEDs at 90 Hz. Here, pose ambiguities are robustly avoided by arranging the LEDs such that they are equally well distributed in all three spatial dimensions. The LED identification is solved by a time consuming combinatorial brute force approach, that is capable of handling the resulting self-occlusions. However, it requires computing the projection error of every possible correspondence configuration based on the P3P solution and therefore does not scale well with the total number of LEDs. This strategy is used to initialize (and re-initialize) the 2D-to-3D LED correspondences. After initialization, the LEDs are tracked incrementally in conjunction with motion prediction. Here, the spatial positions of the marker LEDs are calibrated with a commercial multi-camera Vicon motion capturing system.

As mentioned before, relying on markers comes with the constraint of having to modify the objects of interest, and approaches using infrared light suffer in the presence of sunlight. On the other hand, including such specifically designed fiducials can potentially lead to an increase in accuracy and robustness. Also, active infrared light in conjunction with a corresponding filter massively reduces the complexity of the image processing steps. Hence, by accepting the aforementioned drawbacks and exploiting the advantages, the goal of HSRM-Tracking was to push the boundaries of what is possible in terms of pose accuracy and runtime performance under these conditions for the fairly large range of suitable scenarios. How this was achieved

will be explained in detail within the remainder of this chapter which is structured as follows.

Section 3.2.2 starts by introducing two proposed LED pattern designs for HSRM-Tracking markers. It is followed by Section 3.2.3 and Section 3.2.4 explaining how to detect, localize and identify the LED projections in the image in order to establish the 2D-to-3D point correspondences. Section 3.2.5 continues by giving an in-depth explanation on how the minimal spatial structure of the markers is sufficient to accurately estimate their poses and thereby robustly avoid pose ambiguities. Next, in Section 3.2.6 it is shown how to extend the LED identification approach to multiple markers by utilizing an additional property of the proposed LED patterns that allows to distinguish between them. The section concludes by presenting a light-weight monocular approach to precisely calibrate the spatial structure of the markers semi-automatically in Section 3.2.7 and a qualitative experimental performance evaluation of the overall system in Section 3.2.8.

3.2.2 Marker Design

The proposed markers consist of multiple infrared LEDs connected to a small battery pack. Here it is suggested to use SMD (surface-mounted device) LEDs which are perceptible from large viewing angles up to nearly $\pi/2$, due to their even angular light distribution. For HSRM-Tracking two different marker designs have been developed, in form of geometrically constrained patterns. The markers can either consist of seven LEDs which are arranged in a *cross-shaped* pattern (as originally suggested in [TSSS15]) or six LEDs forming a *circular* structure (see Figure 3.2).

The spatial LED coordinates are denoted relative to the marker coordinate frame and are referred to as $\mathbf{X}_{i_m}, i = 0, \ldots, n-1$, where either $n = 7$ or $n = 6$ depending on the proposed marker design. In case of the cross-shaped pattern the LED at $\mathbf{X}_{3_m} = (0, 0, 0)^\top$ defines the origin of the marker coordinate frame. Furthermore, the LEDs $\mathbf{X}_{0_m}$, $\mathbf{X}_{1_m}$ and $\mathbf{X}_{2_m}$ are ideally placed on a line along the X_m-axis in positive direction with increasing distance to the origin. The two LEDs $\mathbf{X}_{4_m}$ and $\mathbf{X}_{5_m}$ are placed on the Y_m-axis in each positive and negative direction. Hence, $\mathbf{X}_{0_m}, \ldots, \mathbf{X}_{5_m}$ all lie in the $Z_m = 0$ plane whereas $\mathbf{X}_{6_m}$ (placed in negative X_m direction at $Y_m =$

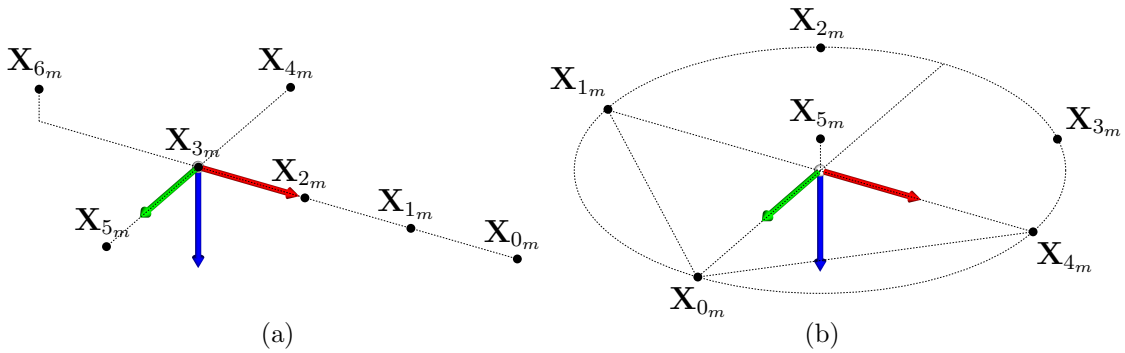


Figure 3.2: Two different marker designs proposed for HSRM-Tracking. The LEDs can either form a cross-shaped pattern of seven points (a) or a circular pattern of six points (b). In both cases all LEDs are co-planar except for the one with the highest index (i.e. $\mathbf{X}_{6_m}$ or $\mathbf{X}_{5_m}$ respectively) which is slightly elevated in negative Z_m-direction.

0) is slightly elevated in negative Z_m-direction in order to add spatial structure to the marker.

In case of the circular pattern the LEDs $\mathbf{X}_{0_m}, \ldots, \mathbf{X}_{4_m}$ lie in the $Z_m = 0$ plane along a circle centered at the origin of the marker coordinate frame. The LED $\mathbf{X}_{5_m}$ is located at that center but slightly elevated in negative Z_m-direction. Note that in both cases the LED with the highest index was chosen to be the one that is not co-planar to the others. In this circular pattern $\mathbf{X}_{0_m}$ is placed on the Y_m-axis in positive direction and $\mathbf{X}_{1_m}$ and $\mathbf{X}_{4_m}$ are placed on the X_m axis in each positive and negative direction. The remaining LEDs $\mathbf{X}_{2_m}$ and $\mathbf{X}_{3_m}$ can be placed at arbitrary positions along the circle segment with negative a Y_m-coordinate except for on the Y_m-axis (at $X_m = 0$ in order to avoid unwanted symmetries when determining correspondences, as explained in Section 3.2.4).

In practice all markers are built by mounting the LEDs according to the constraints imposed by either of those patterns. Regardless of whether this is done by a machine or a human the resulting spatial LED locations will in general contain some imprecision which is inherent to the manufacturing process. Also each $\mathbf{X}_{i_m}$ is assumed to be the spatial location of the light source. However, this location depends on the internal structure of the LED which potentially varies even for those of the same LED type. Therefore, their exact spatial positions have to be measured once

precisely in a pre-calibration step in order to obtain accurate pose estimation results from them, as explained in Section 3.2.7.

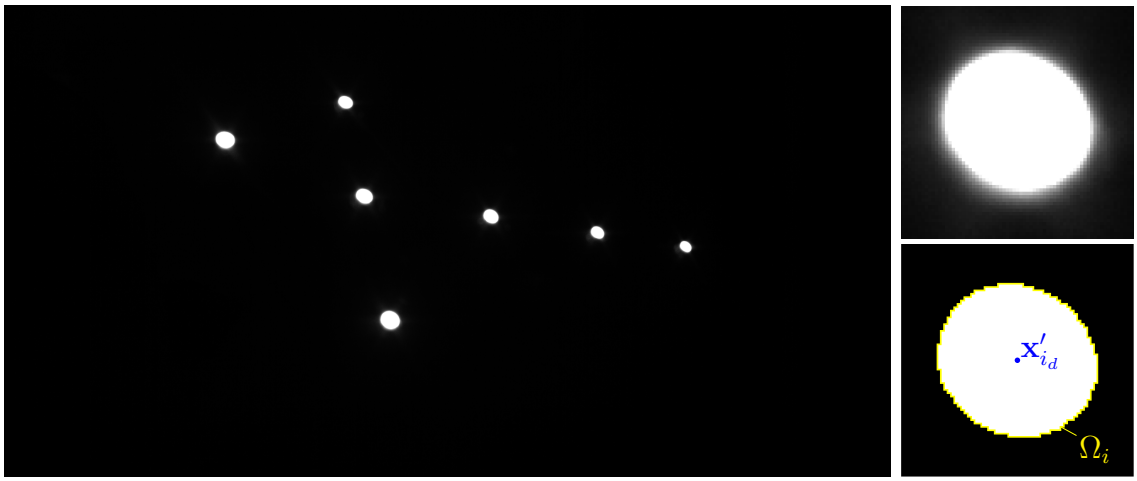
3.2.3 LED Segmentation and 2D Localization

The LEDs will appear as bright blobs in the camera frames (see Figure 3.3). These regions are saturated to a great extent and therefore significantly brighter than the rest of the image. To reduce the influence of ambient light, here a monochrome camera equipped with an infrared-pass filter is used, such that no other light hits the sensor. Thus, ideally only the projections of the LEDs remain visible in the camera image. In order to segment the LED blobs, the original image $I : \Omega \to [0, 255] \subset \mathbb{Z}$ is first transformed into a binary representation $I_b : \Omega \to \{0, 1\}$ by performing simple pixel-wise thresholding as

$$bin(I, s) \doteq I_b(\mathbf{x}) = \begin{cases} 1 & I(\mathbf{x}) > s \\ 0 & \text{otherwise} \end{cases}, \quad \forall \mathbf{x} \in \Omega,$$

where only pixels are kept with an intensity above a prescribed threshold s (see again Figure 3.3). The remaining pixels are grouped into connected regions Ω_i using a union-find algorithm. This allows to directly discard regions below a certain area size, in order to filter small blobs caused by image noise and other unwanted artifacts.

In a next step, a single 2D point $\mathbf{x}'_{i_d}$ is calculated from each remaining such blob region Ω_i that best approximates the 2D projection of the LED light source. Note that d in the index indicates that these coordinates are initially computed from the original image containing lens distortion. This is because remapping the whole image is computationally much more demanding than only removing the distortion from a set of a few extracted points. These 2D locations should furthermore be computed with sub-pixel precision and maximum accuracy since all succeeding calculations build up on them. Essentially any inaccuracy introduced in this step will have an irreversible negative influence throughout the rest of the point-based pose estimation



(a) (b)

Figure 3.3: An example camera image showing the projection of a cross-shaped LED marker (a). A magnified view of one of these LED projections shows that the pixels typically become more saturated towards the center of it (b) (top). In a binary representation pixels with an intensity above a certain threshold (in this case $s = 127$) are extracted and grouped into connected region Ω_i (b) (bottom).

procedure. In general these locations are computed as

$$\mathbf{x}'_{i_d} = \bar{\mathbf{x}}_{\Omega_i}, \quad \bar{\mathbf{x}}_{\Omega_i} \doteq [\bar{x}_{\Omega_i}, \bar{y}_{\Omega_i}]^\top,$$

where $\bar{\mathbf{x}}_{\Omega_i}$ is some weighted mean over Ω_i. A straightforward approach to compute this mean is based on so-called *region moments* of $(p+q)$th order, defined as

$$m_{i_{pq}} \doteq \sum_{\mathbf{x} \in \Omega_i} x^p y^q g(I(\mathbf{x})), \quad \mathbf{x} = [x, y]^\top,$$

where $g : \mathbb{R} \mapsto \mathbb{R}$ is an arbitrary transformation of the image intensities. Given this, the intensity weighted centroid of each blob region Ω_i is computed as

$$\bar{x}_{\Omega_i} = \frac{m_{i_{10}}}{m_{i_{00}}}, \quad \bar{y}_{\Omega_i} = \frac{m_{i_{01}}}{m_{i_{00}}},$$

where in the simplest form the moments are computed directly from the binary image, meaning $g(I(\mathbf{x})) = I_b(\mathbf{x})$. Mapping all the intensities to 1 thereby allows to

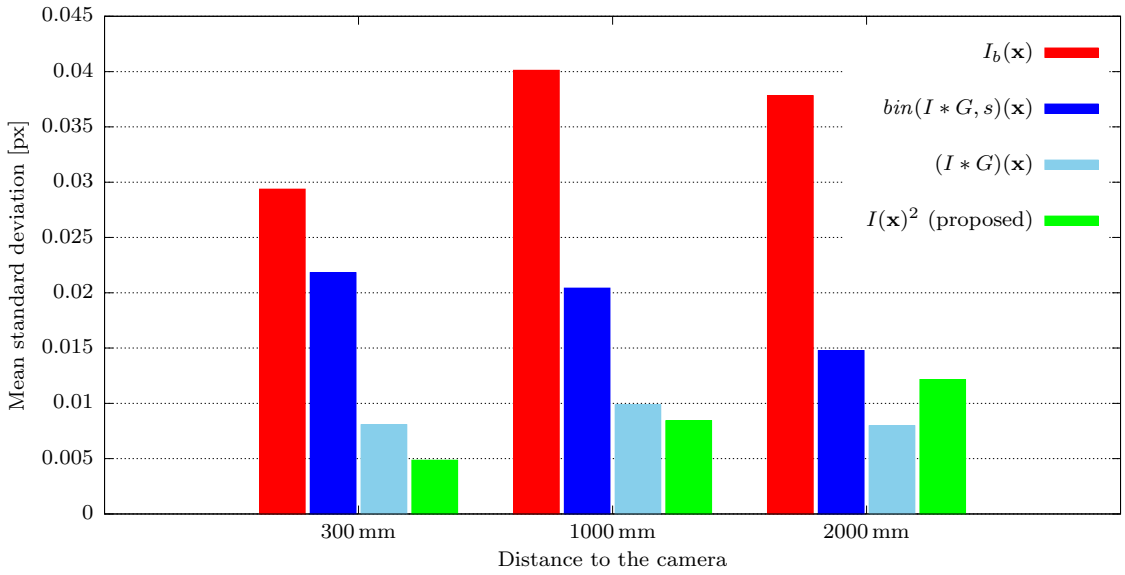


Figure 3.4: Results of the 2D LED location jitter comparison experiment. The plots show the mean standard deviation of all extracted locations $\mathbf{x}'_{i_d}$ at each distance to the camera for different centroid weighting strategies in comparison. Regardless of the method, a binarization threshold of $s = 50$ was used.

maximally simplify the required computations. However, this approach is sensitive to measurement noise of the camera sensor that causes the pixel intensities to flicker. This means that pixels randomly contribute to blob regions which ultimately results in slight jitter of the estimated 2D locations $\mathbf{x}'_{i_d}$ even when there is no actual motion of either the marker or the camera. This is commonly countered by first applying a low-pass filter G (such as Gaussian blurring) to the raw camera image as $I * G$, in order to reduce these high frequency changes. Similar to rectifying the whole image, such a convolution operation is costly compared to the overall computations necessary for pose estimation. For example, applying a 3×3 Gaussian blur kernel to a 1280×1024 px image takes about 1 ms which in this case equates to the entire runtime per frame of the proposed method (see Section 3.2.8).

In addition to smoothing the image, the sub-pixel accuracy of the estimated centroids can be increased by weighting it with the pixels' intensities, e.g. choosing $g(I(\mathbf{x})) = I(\mathbf{x})$ as done in [FMSS14]. However, since the low-pass filter will in general not eliminate the image noise completely, the resulting centroid location will still be

influenced by it. Pixels with lower intensities at the border of these blobs typically tend to flicker stronger compared to the more saturated pixels near the center. Based on this observation, in this work a simpler strategy with a similar effect is suggested that is significantly less computationally demanding. Here, in order to reduce the influence of these border pixels a quadratic weighting of the intensities $g(I(\mathbf{x})) = I(\mathbf{x})^2$ is used instead of applying a low-pass filter to the image.

The performance of the proposed segmentation approach was evaluated in an experiment where a prototype cross-shaped marker was fixed on a tripod and placed statically in front of the camera at 300 mm, 1000 mm and 2000 mm distance to it (see Figure 3.4). At each of these three locations 1000 frames were recorded at 1280×1024 px image resolution. In all images all seven 2D LED locations were then estimated using $g(I(\mathbf{x})) = I_b(\mathbf{x})$, $g(I(\mathbf{x})) = bin(I * G, s)(\mathbf{x})$, $g(I(\mathbf{x})) = (I * G)(\mathbf{x})$ and $g(I(\mathbf{x})) = I(\mathbf{x})^2$, with G being 3×3 Gaussian blur kernel. The results show that the proposed approach reduces jitter of the measured locations $\mathbf{x}'_{i_d}$ similarly to the previously suggested strategies, while having a neglectable influence on the runtime. Finally, in a last step all determined 2D LED projection centroids $\mathbf{x}'_{i_d}$ are mapped to undistorted image coordinates $\mathbf{x}'_i$ using the pre-calibrated camera intrinsics.

3.2.4 Determining 2D-to-3D correspondences

Since all LED projections look alike they cannot be distinguished by local analysis of their image intensities. Therefore, the LED patterns of the HSRM-Tracking markers were designed such that their projection is unambiguous from all perspectives under rigid body motion. This allows to determine all 2D-to-3D correspondences from a single image by analyzing the entire projected pattern $\mathbb{M} = \{\mathbf{x}'_0, \ldots, \mathbf{x}'_{n-1}\}$, i.e. the relative positions of the extracted coordinates $\mathbf{x}'_i$ with respect to it. Here, the only restriction is that all LEDs of a marker have to be visible in the respective frame.

In most parts this analysis is based on constructing lines between 2D points and computing the distance of other 2D points to them. By using homogeneous coordinates $\tilde{\mathbf{x}}'_i = [x'_i, y'_i, 1]^\top$ of extracted 2D points these computations can be simplified. Here, the cross product of two 2D points in homogeneous representation simply gives

the 2D line between them as

$$\mathbf{l} = \tilde{\mathbf{x}}_i \times \tilde{\mathbf{x}}_j = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \in \mathbb{R}^3,$$

where a, b and c are the parameters of that line implicit in form $l : ax + yb + c = 0$. Accordingly, a signed distance d of a third 2D point $\mathbf{x}$ to a line given by $\mathbf{l}$ can be computed from the dot-product as

$$\tilde{\mathbf{x}}^\top \mathbf{l} = ax + by + c = d \in \mathbb{R},$$

by again using the homogeneous representation. Additionally the cross product of two such homogeneous line vectors

$$\tilde{\mathbf{s}} = \mathbf{l}_i \times \mathbf{l}_j \in \mathbb{R}^3,$$

results in the homogeneous representation of their 2D intersection, which is then given by $\mathbf{s} = \pi(\tilde{\mathbf{s}})$.

The Cross-Shaped Marker Due to the design of the cross-shaped LED pattern the projections of at least four points, namely $\mathbf{x}'_0$, $\mathbf{x}'_1$, $\mathbf{x}'_2$ and $\mathbf{x}'_3$ must always lie on a straight line in the image, denoted by $\mathbf{l}_1$. Therefore, the algorithm starts by determining $\mathbf{l}_1$ in the set of projections $\mathbb{M}$. For this, all thirty-five unique subsets of four points from $\mathbb{M}$ are selected. Next, two out of the four points from each subset are randomly selected and the line between them is computed. The other two points of the subset are then used to calculate their absolute distance to the respective line. Assuming that these distances are small when all four points are incident to a common line (e.g. $\mathbf{l}_1$), they provide a quality measure of the co-linearity of a subset. This allows to sort the resulting lines and the corresponding subsets in descending order according to the largest of these two distances (i.e. the largest outlier).

From most perspectives this strategy will yield one dominant line with four participants, meaning that the quality of the best line is significantly better than that of all

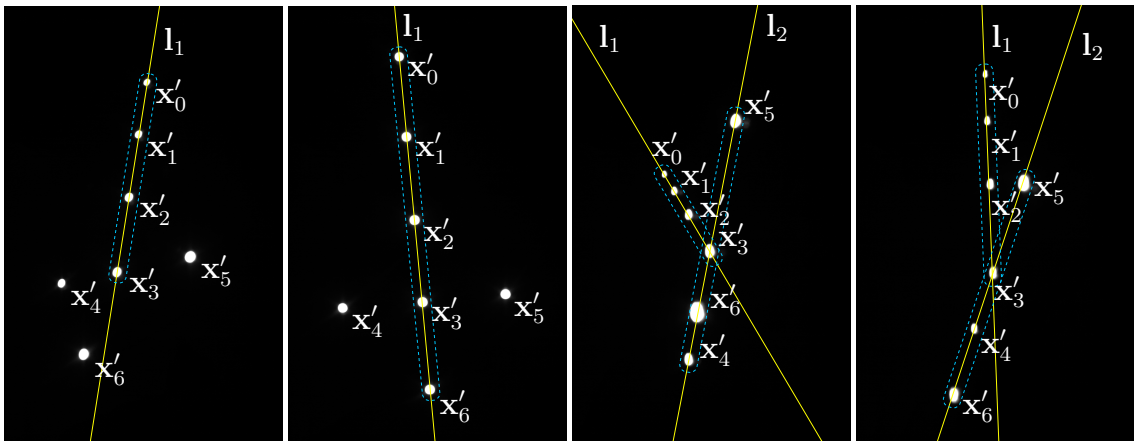


Figure 3.5: Camera images of a cross-shaped marker showing the different projection configurations distinguished during correspondence assignment. The dominant lines $\mathbf{l}_1$ and $\mathbf{l}_2$ are drawn in yellow and their participants are framed by a dashed blue line. In case of *L4* the dominant line $\mathbf{l}_1$ has four participants (a). In case of *L5* $\mathbf{l}_1$ has five participants, including $\mathbf{x}'_6$ (b). In case of *L44* two dominant lines $\mathbf{l}_1$ and $\mathbf{l}_2$ with each four participants appear in the projection pattern. Here $\mathbf{x}'_6$ can either lie in-between (c) or outside of $\mathbf{x}'_4$ and $\mathbf{x}'_5$ on $\mathbf{l}_2$ (d).

others. This case is called *L4* in the following (see Figure 3.5 (a)). However, when seen from the top, meaning that the angle between the principal axis of the camera and the Z_m-axis of the marker is small, $\mathbf{x}'_6$ will also project onto $\mathbf{l}_1$, such that it has five participants. This case is referred to as *L5* in the following (see Figure 3.5 (b)). It can be detected by checking whether the five best ranked lines are of similarly high quality and have a similar orientation. Furthermore, from certain perspectives $\mathbf{x}'_6$ together with $\mathbf{x}'_3$, $\mathbf{x}'_4$ and $\mathbf{x}'_5$ forms a second dominant line $\mathbf{l}_2$ in the image, due to the spatial structure of the marker (see Figure 3.5 (c) and (d)). In this case, called *L44*, the two top ranked lines will be of similar and significantly better quality than all others but have distinctly different orientations. For large outer plane rotation of the marker, meaning that the angle between the principal axis and the Z_m-axis is close to 90°, $\mathbf{l}_1$ can also have more than five participants when grouping the top ranked lines by quality and angle. If this or any other degenerate cases are detected, they are directly discarded since the projected pattern does not provide sufficient information in order to reliably identify correspondences. Otherwise, detecting ei-

ther *L4*, *L5* or *L44* allows to identify the individual correspondences in a next step. This is done with separate strategies depending on the respective case, as follows:

***L4*:** The point on $\mathbf{l}_1$ with the largest distance to the centroid of the three points not lying on $\mathbf{l}_1$ must be $\mathbf{x}'_0$. Next, $\mathbf{x}'_1, \mathbf{x}'_2$ and $\mathbf{x}'_3$ can be assigned ascendingly according to their distance to $\mathbf{x}'_0$. One of the three remaining LED projections (either $\mathbf{x}'_4$ or $\mathbf{x}'_5$) will lie solely on one side of $\mathbf{l}_{03} = \tilde{\mathbf{x}}'_0 \times \tilde{\mathbf{x}}'_3$. It can be detected and assigned by calculating the three signed distances to $\mathbf{l}_{03}$. Since the direction of $\mathbf{l}_{03}$ is known, the sign of this distance for $\mathbf{x}'_4$ and $\mathbf{x}'_5$ is unambiguous. Assuming for example this assignment yields $\mathbf{x}'_5$, then $\mathbf{x}'_4$ and $\mathbf{x}'_6$ are assigned in ascending order according to their distance to the line $\mathbf{l}_{35} = \tilde{\mathbf{x}}'_3 \times \tilde{\mathbf{x}}'_5$.

***L5*:** The two points that do not belong to $\mathbf{l}_1$ define a second line $\mathbf{l}_2$. The point closest to the intersection of $\mathbf{l}_1$ and $\mathbf{l}_2$ must be $\mathbf{x}'_3$. The point that lies solely on one side of $\mathbf{l}_2$ is $\mathbf{x}'_6$. The points $\mathbf{x}'_0, \mathbf{x}'_1, \mathbf{x}'_2$ are assigned in descending order according to their absolute distance to $\mathbf{l}_2$. Finally $\mathbf{x}'_4$ and $\mathbf{x}'_5$ are assigned based on their signed distance to the line $\mathbf{l}_{03} = \tilde{\mathbf{x}}'_0 \times \tilde{\mathbf{x}}'_3$.

***L44*:** First it has to be determined which of the two dominant lines is $\mathbf{l}_1$. It is identified based on the fact that all points incident to $\mathbf{l}_1$ must lie on the same side of line $\mathbf{l}_2$, except $\mathbf{x}'_3$ which is always the closest point to the intersection of $\mathbf{l}_1$ and $\mathbf{l}_2$. Next, $\mathbf{x}'_0$, $\mathbf{x}'_1$ and $\mathbf{x}'_2$ can be enumerated in descending order according to their distance to $\mathbf{x}'_3$. Now the LED projection that lies solely on one side of $\mathbf{l}_1$ is either $\mathbf{x}'_4$ or $\mathbf{x}'_5$. Here, the correct assignment is determined by the signed distance to $\mathbf{l}_1$ as in *L4*. Assuming that this assignment yields $\mathbf{x}'_5$, there are two possibilities (*L44a* and *L44b*) to assign $\mathbf{x}'_6$ and $\mathbf{x}'_4$, which can not be distinguished robustly by 2D criteria only (see Figures 3.5 (c) and (d)). It is therefore resolved with regard to the spatial structure of the marker. At this point four out of six LEDs have already been identified. This allows to select two of these known correspondences (e.g. $\mathbf{x}'_5 \leftrightarrow \mathbf{X}_{5_m}$ and $\mathbf{x}'_0 \leftrightarrow \mathbf{X}_{0_m}$) and combine them with the two possible assignment configurations $\mathbf{x}'_4 \leftrightarrow \mathbf{X}_{6_m}$, $\mathbf{x}'_6 \leftrightarrow \mathbf{X}_{4_m}$ and $\mathbf{x}'_4 \leftrightarrow \mathbf{X}_{4_m}$, $\mathbf{x}'_6 \leftrightarrow \mathbf{X}_{6_m}$ in order to compute the respective pose hypotheses from them by solving P3P. Finally the correct assignment is determined by computing the average projection error across all

LEDs using (3.1) for both solutions. The true configuration is then identified based on the assumption that it will yield the smaller error.

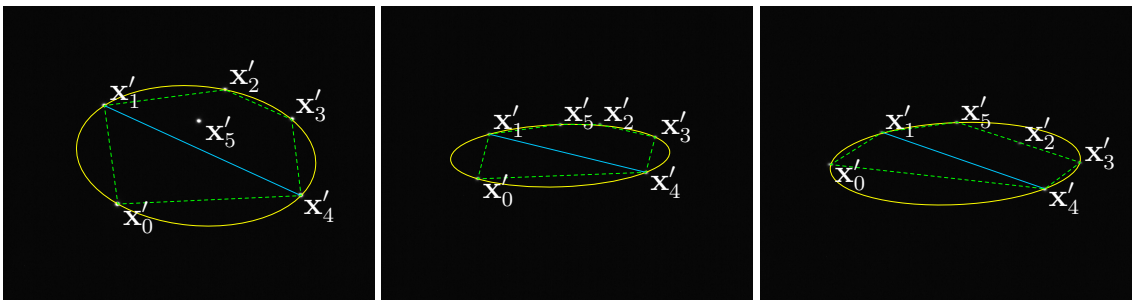
The Circular Marker The correspondence assignment in case of the circular marker is based on the fact that five of the six LEDs, namely $\mathbf{X}_{0_m}, \ldots, \mathbf{X}_{4_m}$ are arranged on a circle. Therefore $\mathbf{x}'_0, \ldots, \mathbf{x}'_4$ will always project onto an ellipse or another circle (being a special case of an ellipse) in the image. Furthermore, from most perspectives $\mathbf{x}'_5$ will lie within the convex hull of $\mathbf{x}'_0, \ldots, \mathbf{x}'_4$ (see Figure 3.6). Thus, the assignment procedure starts by extracting the convex hull of the projected pattern $\mathbb{M}$ in order to separate $\mathbf{x}'_0, \ldots, \mathbf{x}'_4$ from $\mathbf{x}'_5$ which can then be assigned directly. Here, the convex hull can be computed e.g. using the algorithm described in [Skl82] which determines the set of participants in clockwise or counter clockwise order.

Next, an ellipse is fitted in the participating points of the convex hull. Here, an ellipse can be represented implicitly by the second-order polynomial

$$ax^2 + bxy + cy^2 + dx + ey + f = 0,$$

that describes general conic sections. Any conic section can be parametrized by a minimum number of five 2D points incident to it. Thus given at least five points, the parameter vector $[a, b, c, d, e, f]^\top$ of the corresponding polynomial can be obtained by resolving a linear equation system involving their coordinates. However, doing so will potentially give any conic section (e.g. an hyperbola) and not necessarily an ellipse unless all five points are perfectly incident to one, which is unlikely in case of the noisy LED locations extracted from the image. Therefore in order to enforce the resulting conic to be an ellipse, the non-linear constraint $b^2 - 4ac < 0$ has to be incorporated (see e.g. [FPF96] for a detailed solution).

Once the ellipse is determined, the candidates for $\mathbf{x}'_1$ and $\mathbf{x}'_4$ are selected based on the fact that $\mathbf{X}_{1_m}$ and $\mathbf{X}_{4_m}$ are the only two LEDs that are placed exactly opposite to another on the circle in the pattern. For this reason $\mathbf{X}_{2_m}$ and $\mathbf{X}_{3_m}$ can not be placed on the Y_m-axis, as stated in Section 3.2.2. It can therefore be assumed that the center of the ellipse is incident or close to the line $\mathbf{l}_{14} = \tilde{\mathbf{x}}'_1 \times \tilde{\mathbf{x}}'_4$ in the projection. Thus $\mathbf{x}'_1$ and $\mathbf{x}'_4$ are determined by constructing the lines from all ten point pairs on



(a) (b) (c)

Figure 3.6: Camera images of a circular marker showing different projection cases for correspondence assignment. The convex hull is depicted by a dashed green line, the ellipse fitted into the hull points by a yellow line and $\mathbf{l}_{14}$ by a blue line. From most perspectives $\mathbf{x}'_5$ will lie within the convex hull (a). For large outer plane rotation of the marker $\mathbf{x}'_5$ lies on the ellipse and adds to the convex hull (b). For even larger outer plane rotation $\mathbf{x}'_5$ is part of the convex hull instead of another point (e.g. $\mathbf{x}'_2$) that then lies within it (c).

the ellipse, computing the distance of the ellipse center to it and choosing the pair where this is minimal. Note that at this point $\mathbf{x}'_1$ and $\mathbf{x}'_4$ can not yet be distinguished. However $\mathbf{x}'_0$ is identified by computing the three singed distances of the candidates for $\mathbf{x}'_0$, $\mathbf{x}'_2$ and $\mathbf{x}'_3$ to $\mathbf{l}_{14}$, knowing that it lies solely on on side of this line. Afterwards the four remaining correspondences $\mathbf{x}'_1$, $\mathbf{x}'_2$, $\mathbf{x}'_3$ and $\mathbf{x}'_4$ on the ellipse can simply be assigned ascendingly according to their clockwise ordering within the convex hull.

Whenever $\mathbf{x}'_5$ lies within the convex hull of $\mathbb{M}$ the above strategy can be used to assign all six correspondences. This is the case for most perspectives. However, for large outer plane rotation of the marker $\mathbf{x}'_5$ can also lie on the ellipse or even outside of it due to the slight elevation of $\mathbf{X}_{5_m}$ (see Figure 3.6). In both cases $\mathbf{x}'_5$ is then part of the convex hull. In the first case it adds to the hull, meaning it has six participants, whereas in the latter case it can either add to it or replace another point. Therefore, cases where the hull computation yields six participants are directly discarded since they can not provide sufficient information to identify the LEDs.

The scenario where $\mathbf{x}'_5$ replaces another point (i.e. the size of the convex hull is still five) can not be handled by the proposed approach and will lead to mis-assignment.

It should therefore be avoided by construction. This can for example be achieved by either mounting the LEDs $\mathbf{X}_{0_m} \dots \mathbf{X}_{3_m}$ slightly tilted such that they are not visible from the critical angles or mounting all of them on a non-translucent, flat, cone-shaped surface that will lead to desired occlusions.

3.2.5 Robust Pose Estimation

Once all 2D-to-3D correspondences in an image are known, the pose T_{cm} of the associated marker with respect to the camera can be determined. Here, a first estimate T_{cm}^0 is computed from a subset of four correspondences based on the P3P solution [KR17, GHTC03]. In case of the cross-shaped marker these correspondences are chosen as $\mathbf{x}'_0 \leftrightarrow \mathbf{X}_{0_m}$, $\mathbf{x}'_4 \leftrightarrow \mathbf{X}_{4_m}$, $\mathbf{x}'_5 \leftrightarrow \mathbf{X}_{5_m}$ and $\mathbf{x}'_6 \leftrightarrow \mathbf{X}_{6_m}$. Respectively, for the circular marker $\mathbf{x}'_0 \leftrightarrow \mathbf{X}_{0_m}$, $\mathbf{x}'_1 \leftrightarrow \mathbf{X}_{1_m}$, $\mathbf{x}'_4 \leftrightarrow \mathbf{X}_{4_m}$ and $\mathbf{x}'_5 \leftrightarrow \mathbf{X}_{5_m}$ are selected. This initial pose estimate T_{cm}^0 is then refined iteratively by minimizing the projection error across all correspondences

$$E_p(\xi) = \frac{1}{2} \sum_{i=0}^{n-1} \|\mathbf{x}'_i - \mathbf{x}_i(\xi)\|_2^2, \tag{3.2}$$

with respect to twist coordinates ξ, using a Levenberg-Marquardt optimization (see Section 2.3.4). By denoting the per correspondence 2D residuum vector as

$$\mathbf{r}_p(\mathbf{x}_i, \xi) = \mathbf{x}'_i - \mathbf{x}_i(\xi) \in \mathbb{R}^2,$$

the above equation (3.2) may be rewritten as

$$E_p(\xi) = \frac{1}{2} \sum_{i=0}^{n-1} \|\mathbf{r}_p(\mathbf{x}_i, \xi)\|_2^2. \tag{3.3}$$

The gradient of (3.3) then is given by

$$\nabla E_p(\xi) = \sum_{i=0}^{n-1} \left(\mathbf{r}_p(\mathbf{x}_i, \xi)^\top \frac{\partial \mathbf{r}_p(\mathbf{x}_i, \xi)}{\partial \xi} \right)^\top,$$

and accordingly the Hessian matrix is computed as

$$\nabla^2 E_p(\xi) \approx \left(\sum_{i=0}^{n-1} \left(\frac{\partial \mathbf{r}_p(\mathbf{x}_i, \xi)}{\partial \xi} \right)^\top \left(\frac{\partial \mathbf{r}_p(\mathbf{x}_i, \xi)}{\partial \xi} \right) \right) + \lambda_t \operatorname{diag} \left(\sum_{i=0}^{n-1} \left(\frac{\partial \mathbf{r}_p(\mathbf{x}_i, \xi)}{\partial \xi} \right)^\top \left(\frac{\partial \mathbf{r}_p(\mathbf{x}_i, \xi)}{\partial \xi} \right) \right),$$

assuming $\mathbf{r}_p(\mathbf{x}_i, \xi)$ is linear in ξ. By then denoting the 2×6 Jacobian of $\mathbf{r}_p(\mathbf{x}_i, \xi)$ at the current pose, i.e. $\xi_0 = \mathbf{0}$, as

$$J_p(\mathbf{x}_i, \xi_0) = \left. \frac{\partial \mathbf{r}_p(\mathbf{x}_i, \xi)}{\partial \xi} \right|_{\xi_0} = \left. \frac{\partial \mathbf{x}_i' - \mathbf{x}_i(\xi)}{\partial \xi} \right|_{\xi_0} = \left. \frac{\partial \mathbf{x}(\xi)}{\partial \xi} \right|_{\xi_0} \in \mathbb{R}^{2 \times 6},$$

that can be computed according to (2.40), the update step is given by

$$\Delta \xi_t = - \left(\left(\sum_{i=0}^{n-1} J_p(\mathbf{x}_i, \xi_0)^\top J_p(\mathbf{x}_i, \xi_0) \right) + \lambda_t \operatorname{diag} \left(\sum_{i=0}^{n-1} J_p(\mathbf{x}_i, \xi_0)^\top J_p(\mathbf{x}_i, \xi_0) \right) \right)^{-1} \sum_{i=0}^{n-1} J_p(\mathbf{x}_i, \xi_0)^\top \mathbf{r}_p(\mathbf{x}_i, \xi_0).$$

The iterative optimization of T_{cm}^0 results in the marker pose T_{cm_1}. As explained in [SP06], there exist ambiguities when estimating the pose from co-planar points only, which cause the pose to flip under certain configurations. This is because in general the projection error E_p has two distinct local minima when all points are lying in a common plane (see Figure 3.7). Unfortunately, also the single additional point $\mathbf{X}_{n-1_m}$ of the proposed markers that lies out of this plane is not sufficient to robustly eliminate the second minimum when looking at the overall projection error. Thus depending on the quality of the pose initialization the iterative refinement of T_{cm}^0 can end up in either of these minima. This becomes more problematic with increasing distance to the camera, since the accuracy of the linear P3P solution is significantly affected by the uncertainty of the four incorporated extracted 2D coordinates, due to image noise and discretization errors. It has furthermore been shown that these ambiguities are more likely to occur at small angles for outer plane rotation, i.e. when the marker XY-plane is almost parallel to the image plane.

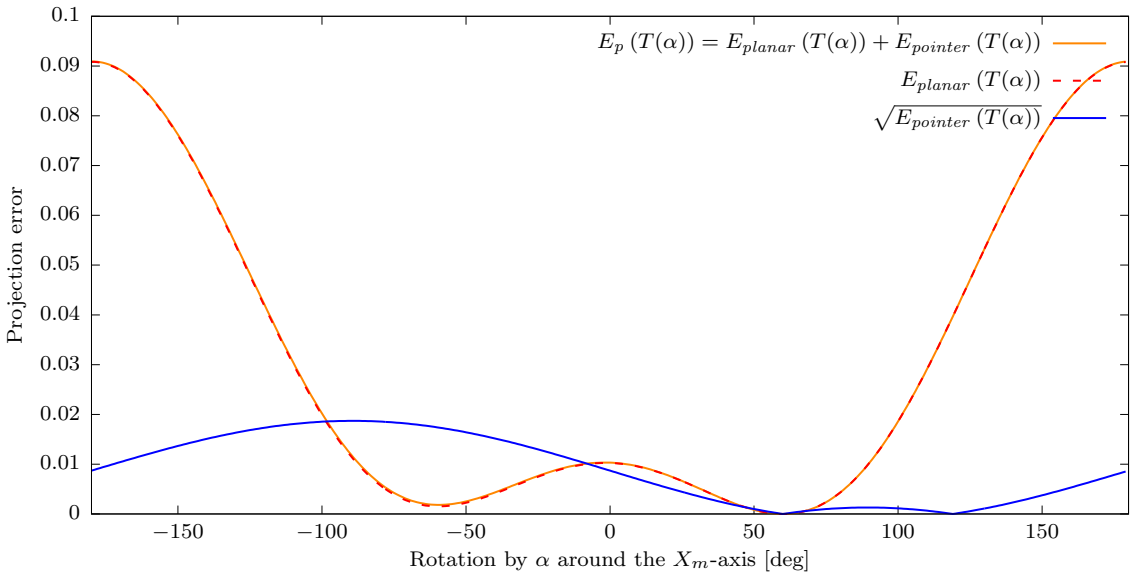


Figure 3.7: Results of a synthetic projection error experiment where a model consisting of four co-planar points $\mathbf{X}_{0_m} = [1, 1, 0]^\top$, $\mathbf{X}_{1_m} = [1, -1, 0]^\top$, $\mathbf{X}_{2_m} = [-1, -1, 0]^\top$, $\mathbf{X}_{3_m} = [-1, 1, 0]^\top$ as well as a fifth point $\mathbf{X}_{4_m} = [0, 0, -0.1]^\top$ that is slightly elevated in the style of the proposed markers. This model was initially projected into a virtual image plane using $K = \mathbf{I}_{3\times3}$ and a ground truth pose T_{cm} composed of $R_{cm} = R_x(\alpha)$, with $\alpha = \pi/3$ (60°) and $\mathbf{t}_{cm} = [0, 0, 10]^\top$ in order to obtain $\mathbf{x}'_0, \ldots, \mathbf{x}'_4$. The plot shows the different projection errors for all pose hypotheses $T(\alpha)$ composed of $R = R_x(\alpha)$, with $\alpha \in [-\pi; \pi]$ and $\mathbf{t} = \mathbf{t}_{cm}$. Note that $\sqrt{E_{pointer}\,(T(\alpha))}$ is plotted for visibility purposes.

Based on this insight, [SP06] presented an approach resolve this ambiguity for planar structures. Given the pose T_{cm_1}, that belongs to one of the minima, a strategy is derived for computing the other solution T_{cm_2} at the second potential minimum directly from it, by only using the correspondences of co-planar points. It was then originally suggested to resolve this ambiguity by choosing the final pose as

$$T_{cm} = \underset{T\in\{T_{cm_1}, T_{cm_2}\}}{\arg\min} E_p(T),$$

assuming that the correct solution will yield the smaller overall projection error. When using purely co-planar markers, this strategy has been shown to be very effective and is still state of the art. However, as experiments showed, it only

robustly resolves the ambiguities at relatively close distances to the camera and becomes unstable at larger distances again (see Figure 3.8). While in theory the minimum at the correct pose is always smaller than at the other local minimum, this criterion is again prone to measurement noise and discretization errors and thus gets corrupted at far distances.

Hence, here a slightly different strategy is proposed with respect to the introduced nearly co-planar markers. It robustly overcomes this problem in the entire distance range wherein the LEDs are still sufficiently visible to the camera. This is essentially achieved by using the single elevated LED $\mathbf{X}_{n-1_m}$ as a *pointer* to the correct solution. For this, the overall projection error E_p (3.1) is split into an error computed from only the co-planar correspondences

$$E_{planar}(T) = \frac{1}{2}\sum_{i=0}^{n-2} \|\mathbf{x}'_i - \mathbf{x}_i(T)\|^2, \tag{3.4}$$

and a second error

$$E_{pointer}(T) = \frac{1}{2}\|\mathbf{x}'_{n-1} - \mathbf{x}_{n-1}(T)\|^2, \tag{3.5}$$

computed from only the single out-of-plane correspondence $\mathbf{x}'_{n-1} \leftrightarrow \mathbf{X}_{n-1_m}$. Given T_{cm_1} from initial refinement by using all correspondences, the second potential solution T_{cm_2} is then calculated using only the co-planar correspondences, following the strategy of [SP06]. Inspired by [YWX+12] the correct solution is then chosen based on the insight that $E_{pointer}$ exhibits significantly different characteristics compared to the overall projection error E_p. Synthetic experiments showed that $E_{pointer}$ and E_p share the sought minimum while being contrary at the delusive minimum (see again Figure 3.7). Thus, here the correct pose is chosen as

$$T_{cm} = \underset{T \in \{T_{cm_1}, T_{cm_2}\}}{\arg\min} E_{pointer}(T),$$

which reliably distinguishes the solutions, robust to measurement noise. In cases where T_{cm_2} was selected, it is once more refined in a final optimization using all correspondences in order to further increase its accuracy. The performance of this pose estimation strategy was evaluated in an experiment, where the camera was fixed on a static tripod and a prototype cross-shaped marker was attached to a

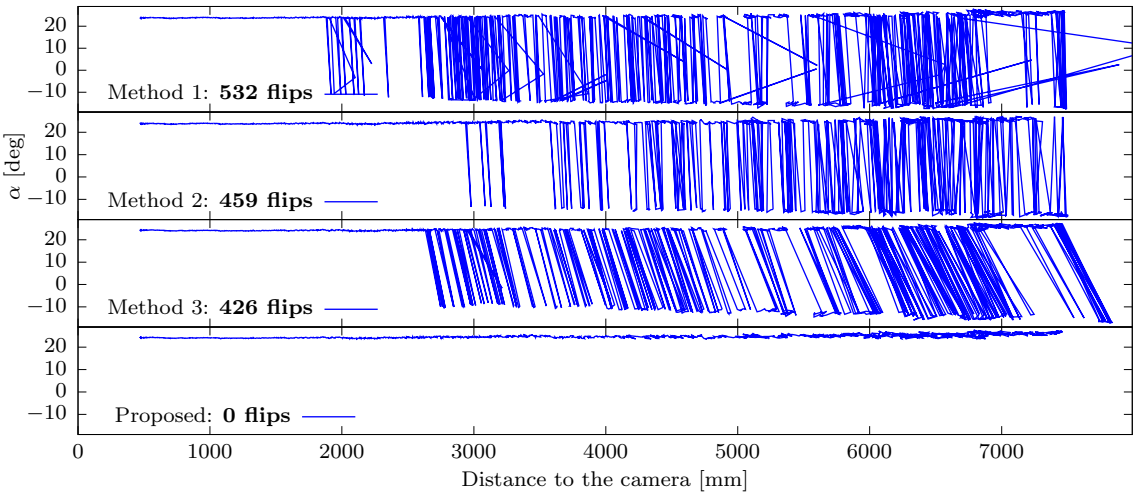


Figure 3.8: Results of the pose flip comparison experiment. The plots show α in relation to t_z obtained from the estimated pose T_{cm}. Given the setup of the experiment, this combination serves as the best visualization for pose flips. The values are plotted in chronological order from frame 0 to frame 2289. Here, negative values of α indicate the occurrence of a pose flip.

wheeled tripod. The marker itself was tilted with an outer plane rotation angle $\alpha \approx 20°$ around the X_m-axis, which according to [SP06] is one of the most critical configurations for the pose ambiguity problem. Starting at about 50 cm distance to the camera a sequence consisting of 2290 frames at 2048×1088 px resolution was recorded while manually moving the marker up to 7.5 m away. This pre-recorded sequence was then processed for pose estimation using the proposed as well as the following three other methods (see Figure 3.8):

Method 1 is the simplest approach where the pose is estimated as $T_{cm} = T_{cm_1}$ by only optimizing E_{planar} without considering the other potential solution.

Method 2 also only considers the six co-planar correspondences but makes use of the improvement of [SP06] by choosing

$$T_{cm} = \underset{T \in \{T_{cm_1}, T_{cm_2}\}}{\arg\min} E_{planar}(T).$$

Method 3 is similar to method 1 where the pose is estimated as $T_{cm} = T_{cm_1}$ without

considering the other potential solution, but optimizes E_p using all correspondences.

All four methods were initialized with the same T_{cm}^0, obtained from P3P. For a comparing evaluation, the total number of pose flips was monitored for each method. Here, a flip could simply be identified by checking for a negative sign of the estimated α. The results show that method 1 performs worst with the most pose flip occurrences in total. Here, the pose starts to flip at about 2 meters distance to the camera. The results for method 2 demonstrate how the strategy of [SP06] reduces the number of pose flips when using a purely co-planar marker. It can be seen that in this case the flips first occur at a distance of about 3 meters to the camera and that their frequency rises with increasing distance. Although method 3 performs better than method 2 regarding the total number of flips, the first flips already occur about 0.5 meters closer to the camera. It can also be seen that the marker pose is estimated further away from the camera due to the elevation of $\mathbf{X}_{(n-1)_m}$ when the minimization ends up in the false minimum. The proposed method did not flip once in the whole sequence, which demonstrates the effectiveness of the pointer strategy even at large distances despite its relatively small elevation of about 1 cm.

3.2.6 Extension to Using Multiple Markers

The HSRM-Tracking algorithm can easily be extended to so-called *multi-marker tracking* (by detection). Here, the poses of multiple markers that are simultaneously visible in the same image have to be estimated (see Figure 3.9 (a)). This requires to first separate and identify the individual markers, based on their projection.

Regardless of the number of markers, the image processing steps are exactly the same as described in Section 3.2.3 until the centeroids of all potential LED blobs are determined. Separation of the individual markers is then performed by the well-known k-means algorithm [Llo82] with an adapted initialization scheme in order to compute clusters of the extracted projections. Here, the number of expected clusters is set to $k = \lceil b/n \rceil$, where b is the total number of visible blobs in the respective frame. That way, false positive detections e.g. caused by reflections of sunlight or

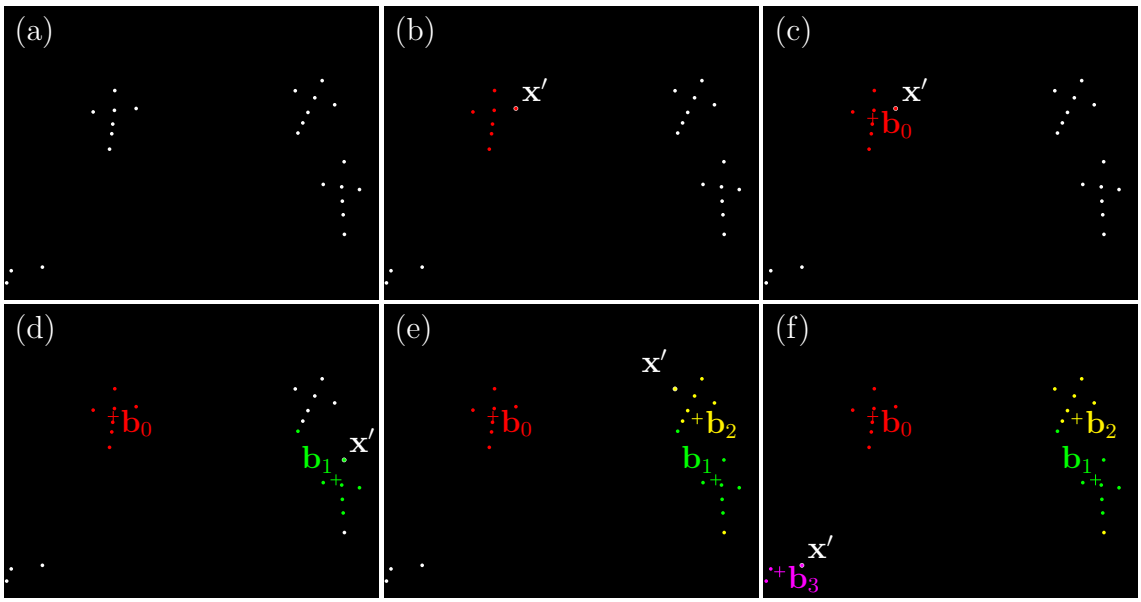


Figure 3.9: An example of the proposed k-means initialization for multi-marker tracking (by detection). Here, three cross-shaped markers are fully visible in the original camera image while only three LED projections of a fourth marker appear in the bottom left corner, meaning $b = 24$ and thus $k = \lceil 24/7 \rceil = 4$ (a). The procedure starts by randomly selecting an unassigned point $\mathbf{x}'$ (indicated by a white outline) and assigning its $n-1 = 6$ nearest neighbors (shown in red) to it (b). Next the first cluster center $\mathbf{b}_0$ (depicted as a red cross) is initialized to the barycenter of these seven points assigned to it (c). This process is then repeated until all blobs have been assigned to a cluster (d) - (e). In this case $\mathbf{b}_3$ belongs to the *trash cluster* that can be filtered by its size. The resulting false assignment of the blobs corresponding to $\mathbf{b}_1$ and $\mathbf{b}_2$ is then robustly resolved by the successive k-means algorithm within a few iterations.

markers that are not fully visible, in many cases will end up in a separate smaller cluster and thus can be filtered easily.

In order to speed up and assure correct convergence of k-means, the cluster centers $\mathbf{b}_j$, $j = 0, \ldots, k-1$, are initialized by exploiting the constraint that each marker cluster must have exactly n members (see Figure 3.9). This is done by repeating the following procedure k times, i.e. for $j = 0, \ldots, k-1$ do

1. Randomly choose a point $\mathbf{x}'$ from all unassigned LED projections (see Figure 3.9 (b)).

2. From the set of unassigned LED projections, assign the n-1 (or less, if not more are left) nearest neighbors to this $\mathbf{x}'$ (see also Figure 3.9 (b)).

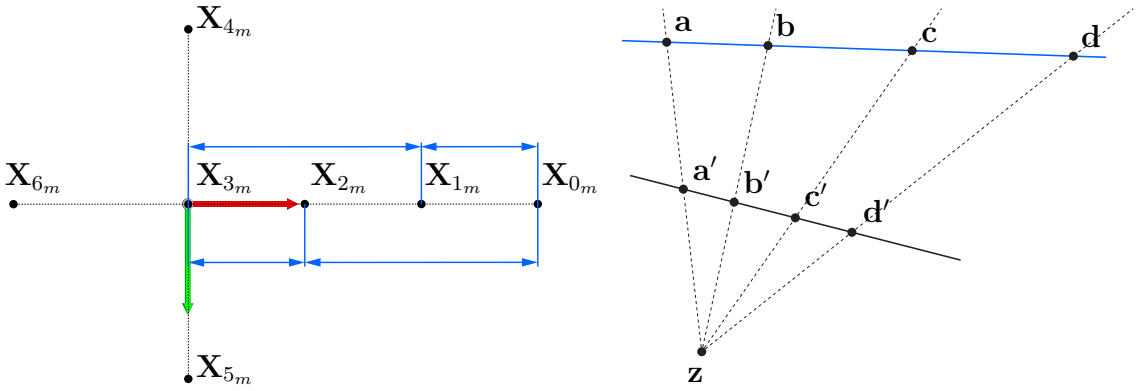
3. Set the location of the current center $\mathbf{b}_j$ to the barycenter of $\mathbf{x}'$ and its assigned neighbors (see Figure 3.9 (c)).

This initialization guarantees that the centers are already well distributed in the 2D image plane, such that k-means itself usually only needs one or two iterations for convergence.

For each 2D projection within a cluster the corresponding LED indices $i = 0, \ldots, n-1$ then can be determined independently for each cluster (i.e. marker) as previously described in Section 3.2.4. Although k-means is a rather simple algorithmic choice for separating the markers and likely to fail if the projections of the markers are too close together, it still works fine for many application scenarios. For example when tracking a human arm and there are two markers attached to the upper and the lower arm, critical configurations are unlikely to occur.

Having separated the marker projections and assigned the correct LED indices, the only thing left to do before the poses can be estimated is to identify the corresponding marker that belongs to each cluster of projections. This is required since the spatial geometry of each marker is different and thus has to be known in order to perform point-based pose estimation from the correct 2D-to-3D correspondences. Here another property of the proposed marker design is utilized. Both the cross-shaped as well as the circular LED pattern encode a unique ID for each marker. This ID is computed based on the so-called *cross-ratio*, being a real number associated with a subset of the LEDs that is invariant under perspective projection.

In case of the cross-shaped marker, the cross ratio is computed from the four co-linear LEDs $\mathbf{X}_{0_m}$, $\mathbf{X}_{1_m}$, $\mathbf{X}_{2_m}$ and $\mathbf{X}_{3_m}$, where the line between $\mathbf{X}_{0_m}$ and $\mathbf{X}_{3_m}$ is divided by both $\mathbf{X}_{1_m}$ and $\mathbf{X}_{2_m}$ into two sections each defining a division ratio (see Figure 3.10 (a)). In general the cross-ratio of four such co-linear points describes the ratio of these two division ratios and is therefore also often called *double-ratio*. Thus, given four points $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \mathbb{R}^n$ on one line, where $\mathbf{b}$ and $\mathbf{c}$ are located between $\mathbf{a}$



(a) (b)

Figure 3.10: A visualization of the cross-ratio marker ID in case of the cross-shaped marker computed from the four co-linear points $\mathbf{X}_{0_m}$, $\mathbf{X}_{1_m}$, $\mathbf{X}_{2_m}$ and $\mathbf{X}_{3_m}$. Here the sections of the two corresponding division ratios are depicted by blue arrows (a). A 2D example shows how the cross ratio of any four co-linear points remains the same when projected onto another line i.e. $\mathcal{X}_|(\mathbf{a},\mathbf{d};\mathbf{c},\mathbf{b}) = \mathcal{X}_|(\mathbf{a}',\mathbf{d}';\mathbf{c}',\mathbf{b}')$ with $\mathbf{z}$ being the center of projection (b).

and $\mathbf{d}$ (see Figure 3.10 (b)), one possible definition of the cross-ratio is

$$\mathcal{X}_|(\mathbf{a},\mathbf{d};\mathbf{c},\mathbf{b}) \doteq \frac{\|\mathbf{a}-\mathbf{c}\|_2}{\|\mathbf{d}-\mathbf{c}\|_2} : \frac{\|\mathbf{a}-\mathbf{b}\|_2}{\|\mathbf{d}-\mathbf{b}\|_2} = \frac{\|\mathbf{a}-\mathbf{c}\|_2 \cdot \|\mathbf{d}-\mathbf{b}\|_2}{\|\mathbf{d}-\mathbf{c}\|_2 \cdot \|\mathbf{a}-\mathbf{b}\|_2}.$$

Note that there are in total six different unique definitions of the cross-ratio from four co-linear points which are however all directly related to one another.

A cross-shaped marker is then identified based on the assumption that

$$\mathcal{X}_|(\mathbf{x}'_0,\mathbf{x}'_3;\mathbf{x}'_2,\mathbf{x}'_1) \approx \mathcal{X}_|(\mathbf{X}_{0_m},\mathbf{X}_{3_m};\mathbf{X}_{2_m},\mathbf{X}_{1_m}),$$

despite the inaccuracy of each $\mathbf{x}'_i$, since the cross-ratio is projective invariant i.e. it is preserved under projective transformations of the corresponding points. This means, that by varying the relative positions of $\mathbf{X}_{1_m}$ and $\mathbf{X}_{2_m}$ along the line between $\mathbf{X}_{0_m}$ and $\mathbf{X}_{3_m}$ new unique marker IDs can be constructed.

In case of the circular marker where no four points are co-linear, the cross-ratio is computed from the five co-planar LEDs $\mathbf{X}_{0_m}$, $\mathbf{X}_{1_m}$, $\mathbf{X}_{2_m}$, $\mathbf{X}_{3_m}$ and $\mathbf{X}_{4_m}$ on the circle.

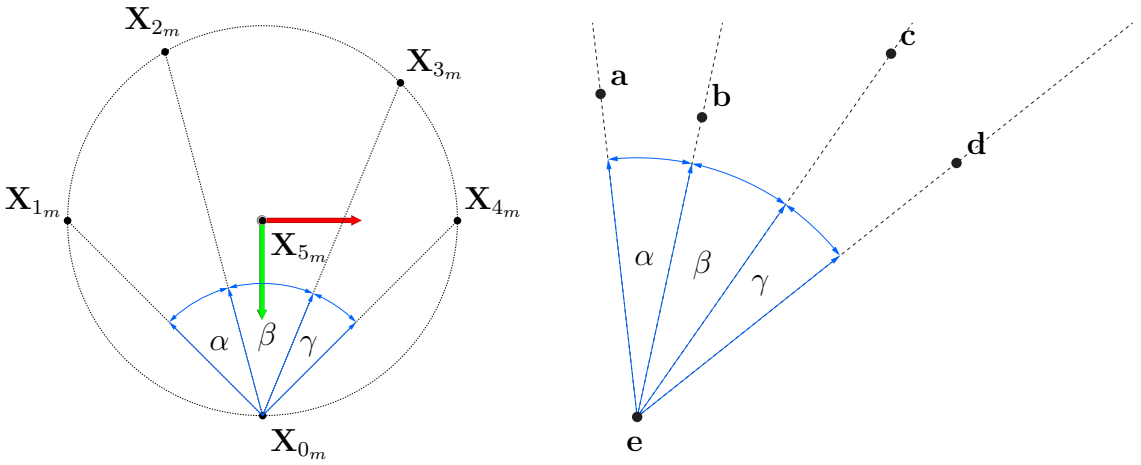


Figure 3.11: A visualization of the cross-ratio marker ID in case of the circular marker computed from the five co-planar points $\mathbf{X}_{0_m}$, $\mathbf{X}_{1_m}$, $\mathbf{X}_{2_m}$, $\mathbf{X}_{3_m}$ and $\mathbf{X}_{4_m}$. Here the relevant angles α, β and γ between four lines through $\mathbf{X}_{0_m}$ and each $\mathbf{X}_{1_m}, \ldots, \mathbf{X}_{4_m}$ are depicted by blue arrows (a). A general 2D example shows that a cross ratio $\mathcal{X}_{\bigcirc}(\mathbf{a}-\mathbf{e}, \mathbf{d}-\mathbf{e}; \mathbf{b}-\mathbf{e}, \mathbf{c}-\mathbf{e})$ can be computed from any four points $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$, each on a line intersecting in a fifth point $\mathbf{e}$, co-planar to the others (b).

Here the angles between the four direction vectors $\mathbf{X}_{1_m} - \mathbf{X}_{0_m}, \ldots, \mathbf{X}_{4_m} - \mathbf{X}_{0_m}$ of the corresponding lines that are co-punctal $\mathbf{X}_{0_m}$ are utilized (see Figure 3.11 (a)). In general the angles between the direction vector of four co-planar lines in $\mathbb{R}^n$ that intersect in a single point $\mathbf{e} \in \mathbb{R}^n$ also define a cross-ratio. Assuming the four points $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \mathbb{R}^n$ are each located on one of these lines and are not equal to $\mathbf{e}$, the corresponding direction vectors can be obtained as $\mathbf{a}-\mathbf{e}, \mathbf{b}-\mathbf{e}, \mathbf{c}-\mathbf{e}, \mathbf{d}-\mathbf{e} \in \mathbb{R}^n$. Assuming that $\measuredangle(\mathbf{a}-\mathbf{e}, \mathbf{b}-\mathbf{e}) < \measuredangle(\mathbf{a}-\mathbf{e}, \mathbf{c}-\mathbf{e}) < \measuredangle(\mathbf{a}-\mathbf{e}, \mathbf{d}-\mathbf{e})$, where

$$\measuredangle(\mathbf{u}, \mathbf{v}) \doteq \cos^{-1}\left(\frac{\mathbf{u}^\top \mathbf{v}}{\|\mathbf{u}\|_2 \cdot \|\mathbf{v}\|_2}\right),$$

is the angle between two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$, the cross-ratio used here is defined as

$$\mathcal{X}_{\bigcirc}(\mathbf{a}-\mathbf{e}, \mathbf{d}-\mathbf{e}; \mathbf{b}-\mathbf{e}, \mathbf{c}-\mathbf{e}) \doteq \frac{\sin(\alpha+\beta)}{\sin(\beta)} : \frac{\sin(\alpha+\beta+\gamma)}{\sin(\beta+\gamma)} = \frac{\sin(\alpha+\beta) \cdot \sin(\beta+\gamma)}{\sin(\beta) \cdot \sin(\alpha+\beta+\gamma)},$$

with $\alpha = \measuredangle(\mathbf{a}-\mathbf{e}, \mathbf{b}-\mathbf{e})$, $\beta = \measuredangle(\mathbf{b}-\mathbf{e}, \mathbf{c}-\mathbf{e})$ and $\gamma = \measuredangle(\mathbf{c}-\mathbf{e}, \mathbf{d}-\mathbf{e})$, as shown in Figure 3.11 (b)).

A circular marker is then identified based on the assumption that

$$\mathcal{X}_{\bigcirc}(\mathbf{x}'_1 - \mathbf{x}'_0, \mathbf{x}'_4 - \mathbf{x}'_0; \mathbf{x}'_2 - \mathbf{x}'_0, \mathbf{x}'_3 - \mathbf{x}'_0) \approx \mathcal{X}_{\bigcirc}(\mathbf{X}_{1_m} - \mathbf{X}_{0_m}, \mathbf{X}_{4_m} - \mathbf{X}_{0_m}; \mathbf{X}_{2_m} - \mathbf{X}_{0_m}, \mathbf{X}_{3_m} - \mathbf{X}_{0_m}),$$

again due to the projective invariance of the cross-ratio and despite the inaccuracy of each $\mathbf{x}'_i$. Accordingly, in this case new unique marker IDs can be constructed by varying the relative positions of $\mathbf{X}_{2_m}$ and $\mathbf{X}_{3_m}$ along the semicircle segment between $\mathbf{X}_{1_m}$ and $\mathbf{X}_{4_m}$ (again, except for the middle).

All markers to track have to be calibrated beforehand, so that the correspondences between LED projection clusters and markers can be found by simply comparing the cross-ratio of each cluster with the cross-ratio of each known marker. The next section describes a semi-automatic calibration procedure developed specifically for this purpose.

3.2.7 Semi-Automatic Marker Calibration

Before the pose of a marker can be estimated from its 2D projection, its 3D geometry i.e. the absolute spatial locations of its LEDs relative to the marker coordinate frame $\mathcal{M}$ have to be known. The more precisely these positions are measured, the more accurately the pose can be determined. Measuring the positions of the LEDs of a marker manually, however, is error-prone, inconvenient and should therefore be reduced to a minimum.

Thus in this work, a semi-automatic marker calibration method is proposed that uses a so-called *multi-view stereo* approach. It only requires a collection of different images of the corresponding marker as well as a single manually measured distance between two of its LEDs in order to predefine its absolute scale. In case of the cross-shaped marker this scale is measured as the distance d_{03} between the LEDs at $\mathbf{X}_{0_m}$ and $\mathbf{X}_{3_m}$ and for the circular marker d_{14} between the LEDs at $\mathbf{X}_{1_m}$ and $\mathbf{X}_{4_m}$ using a sliding caliper. The collection of m images $I(t_j)$ of the marker at different times t_j, with $j = 0, \ldots, m-1$ can be captured using the same camera as later for pose estimation or a similar calibrated monochrome camera that is also equipped with an

infrared filter. Here, it is important that these images are captured from as different as possible viewpoints and rotations relative to the marker such that they cover a large variety of perspectives in order to maximize their entropy (similar to those of the chessboard pattern used for camera calibration, e.g. Figure 2.10). They should furthermore be captured at a rather short distance between marker and camera in order to minimize the influence of discretization errors related to the extracted 2D locations $\mathbf{x'}_i$. Lastly it is required that only the single marker to be calibrated is in the field of view of the camera which must be fully visible at all times. For this, these images can for example be obtained manually while holding the marker into the camera at different locations and orientations. In practice around 20 calibration images are sufficient, given that they meet the previously mentioned requirements, which means that the whole process can take far less than a minute depending on the experience of the user.

The calibration process starts by extracting the set of assigned LED projections $\mathbb{M}(t_j) = \{\mathbf{x}'_0(t_j), \ldots, \mathbf{x}'_{n-1}(t_j)\}$ from each individual image $I(t_j)$. Here it is exploited that this assignment can be done even without knowing the 3D structure of the observed marker (as explained in Section 3.2.3 and Section 3.2.4). The set of all these projections is denoted by $\mathbb{M}^* = \{\mathbb{M}(t_0), \ldots, \mathbb{M}(t_{m-1})\}$ in the following. Note that in case of a cross-shaped marker, perspectives yielding case *L44* (see Figure 3.5) must be avoided during calibration since they are the only ones that cannot be assigned correctly without knowing the spatial marker geometry in advance. However, these usually rarely occurring frames can be detected and are omitted automatically. Based on the extracted and assigned 2D coordinates the successive complete calibration scheme is then split into an initialization and a refinement step, as explained in the following.

Initialization The initial guesses for the 3D LED locations $\mathbf{X}^0_{i_m}$ are obtained based on the *epipolar geometry*, which describes the relative geometric relationship of two images capturing overlapping parts of the same scene. In this scenario, it is assumed that all images have been captured by the same camera of which all intrinsic parameters are known. This means in particular, that from every image point $\mathbf{x}$ its corresponding ideal image coordinates $\mathbf{x}_c$ can directly be obtained as $\tilde{\mathbf{x}}'_c = K^{-1}\tilde{\mathbf{x}}'$ according to (2.30). Thus, the special case of the epipolar geometry of two so-called

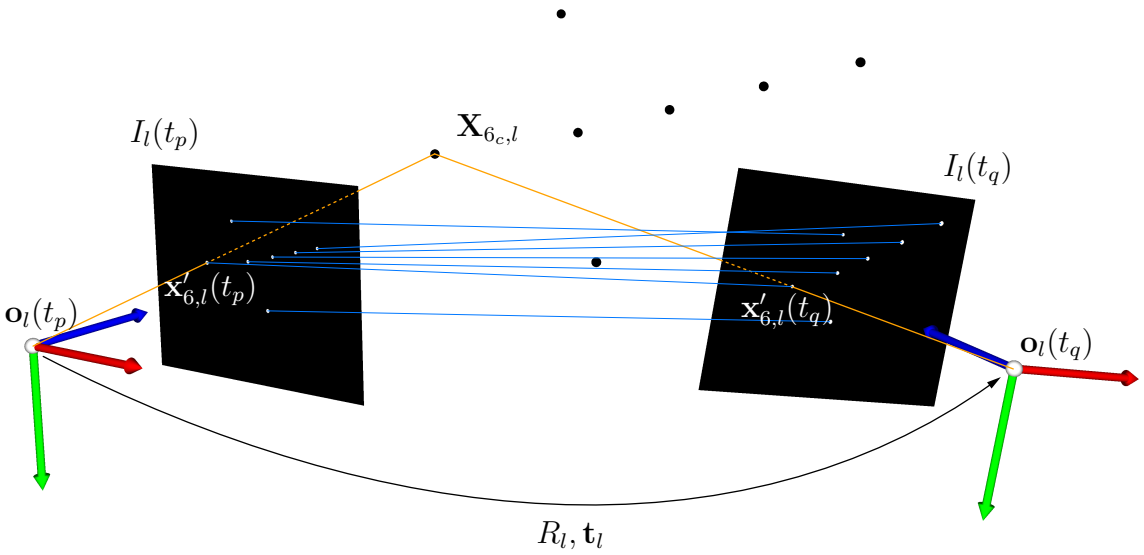


Figure 3.12: A visualization of the geometric relationship between each two calibration images of a pair $(I_l(t_p), I_l(t_q))$ capturing a cross shaped marker. Given the rigid body transform $R_l, \mathbf{t}_l$ between these two calibrated views, in this example the spatial location of the LED $\mathbf{X}_{6_c,l}$ is reconstructed via triangulation. Here the corresponding projection rays are depicted in orange.

calibrated views can be considered, which simplifies the related computations. In the context of multiple-view geometry it is a well-known fact, that given a minimum of five 2D-to-2D ideal point correspondences among two such images, the relative rigid body camera motion that occurred between the two respective calibrated views can be recovered up to scale without any prior spatial knowledge about the scene itself [HZ06, MSKS05, Kru13]. Furthermore, knowing the relative pose between the two cameras allows to reconstruct the 3D location of the corresponding scene points via triangulation from their 2D projections in both views (see Figure 3.12).

Now the main idea of the proposed initialization scheme is to perform the above triangulation procedure for all image pairs in the set of collected marker images in order to use all available information. All these individual spatial measurements are eventually fused into a single initial marker reconstruction that already provides high accuracy for the successive refinement step. The initialization starts by selecting all possible pairs of previously extracted projection sets $(\mathbb{M}_l(t_p), \mathbb{M}_l(t_q)) \in \mathbb{M}^* \times \mathbb{M}^*$,

with $p, q = 0, \ldots, m-1$, $p < q$ and $l = 0, \ldots, \binom{m}{2} - 1$, corresponding to the image pair $(I_l(t_p), I_l(t_q))$. These provide n already correctly assigned 2D-to-2D point correspondences $\tilde{\mathbf{x}}'_{i,l}(t_p) \leftrightarrow \tilde{\mathbf{x}}'_{i,l}(t_q)$ between the respective views that can next be used to recover the relative camera motion for each pair.

Each two views or cameras are related by a rigid body transform, namely

$$T_{cc,l}(t_p, t_q) = T_{cm,l}(t_q) T_{cm,l}(t_p)^{-1},$$

which will be referred to as

$$T_l \doteq \begin{bmatrix} R_l & \mathbf{t}_l \\ \mathbf{0}^\top & 1 \end{bmatrix} = T_{cc,l}(t_p, t_q),$$

in the following. At this point neither $T_{cm,l}(t_q)$ nor $T_{cm,l}(t_p)$ are known and only the relative transform between them is considered. Therefore, one of them, here $T_{cm,l}(t_p) = \mathbf{I}_{4\times4}$, is typically set to be the identity.

For each pair each unknown 3D point $\mathbf{X}_{i_c,l}$ that projects to corresponding $\tilde{\mathbf{x}}'_{i,l}(t_p)$ and $\tilde{\mathbf{x}}'_{i,l}(t_q)$ forms a triangle together with the two optical centers of the cameras $\mathbf{o}_l(t_p)$ and $\mathbf{o}_l(t_q)$ defining an *epipolar plane*. Now assuming that all 2D locations were perfectly extracted, meaning $\tilde{\mathbf{x}}'_i(t_j) = \tilde{\mathbf{x}}_i(t_j)$, then $\tilde{\mathbf{x}}'_{i_c}(t_j) = K^{-1}\tilde{\mathbf{x}}'_i(t_j)$ gives the vector from the optical center in direction of $\mathbf{X}_{i_c}(t_j)$ for each camera, i.e. $Z_{i_c}(t_j)\tilde{\mathbf{x}}'_{i_c}(t_j) = \mathbf{X}_{i_c}(t_j)$. Based on this, the three vectors $R_l\tilde{\mathbf{x}}'_{i_c,l}(t_p)$, $\mathbf{t}_l$ and $\tilde{\mathbf{x}}'_{i_c,l}(t_q)$, being the direction vectors of the three sides of the aforementioned triangle, must also ideally be within their corresponding epipolar plane. This relationship can be expressed by the tri-product

$$\langle \tilde{\mathbf{x}}'_{i_c,l}(t_q), \mathbf{t}_l \times R_l\tilde{\mathbf{x}}'_{i_c,l}(t_p) \rangle = \tilde{\mathbf{x}}'_{i_c,l}(t_q)^\top \hat{\mathbf{t}}_l R_l \tilde{\mathbf{x}}'_{i_c,l}(t_p) = 0,$$

which is usually referred to as the *epipolar constraint*. The matrix resulting from the unknowns $\hat{\mathbf{t}}_l \in \mathfrak{se}(3)$ and $R_l \in \mathbb{SO}(3)$ is called the *essential matrix*

$$E_l = \hat{\mathbf{t}}_l R_l \in \mathbb{E},$$

being an element of the so-called *essential space* denoted by

$$\mathbb{E} \doteq \left\{ E = \hat{\mathbf{t}} R \;\middle|\; R \in \mathbb{SO}(3), \mathbf{t} \in \mathbb{R}^3 \right\} \subset \mathbb{R}^{3\times 3}.$$

Given this, the epipolar constraint can be rewritten as

$$\tilde{\mathbf{x}}'_{i,l}(t_q)^\top E_l \tilde{\mathbf{x}}'_{i,l}(t_p) = 0. \tag{3.6}$$

An essential matrix can be resolved from at least five epipolar constraints (3.6) each given by a normalized ideal 2D-to-2D point correspondence, using the thus so-called *5-point algorithm* [Nis04]. It will give the best solution robust to noise, which is important, because in practice the 2D locations cannot be perfectly determined, i.e. $\tilde{\mathbf{x}}'_i(t_j) \approx \tilde{\mathbf{x}}_i(t_j)$. Given an essential matrix $E_l = \hat{\mathbf{t}}_l R_l$, the corresponding rotation matrix R_l and the translation vector $\mathbf{t}_l$ can be recovered up to scale [Nis04]. Since the scale of the observed scene is unknown, the length of the baseline between the two cameras is commonly set to 1, i.e. $\|\mathbf{t}_l\|_2 = 1$.

In the next step, for each image pair the set of n 3D LED locations $\mathbf{X}_{i_c,l}(t_p)$ is reconstructed via triangulation. For this, the unknown $Z_{i_c,l}(t_p)$ values are determined by finding the best solution to

$$\mathbf{x}'_{i_c,l}(t_q) - \pi\left(R_l Z_{i_c,l}(t_p) \tilde{\mathbf{x}}'_{i_c,l}(t_p) + \mathbf{t}_l\right) = 0,$$

as described in detail in e.g. [HZ06, MSKS05]. The resulting 3D locations are all estimated with respect to the individual relative camera motion T_l and thus all scaled and oriented differently for each pair. In all previous steps the assumption was made that for each pair it holds $T_{cm,l}(t_p) = \mathbf{I}_{4\times 4}$ and thus $\mathbf{X}_{i_c,l}(t_p) = \mathbf{X}_{i_m,l}(t_p)$. This implies that the marker coordinate frame is equal to the camera coordinate frame, which however cannot be true. Therefore, to fuse these individual marker reconstructions in order to obtain the final estimate of the initialization, they are aligned and registered into the common marker coordinate frame.

Thus, a transformation

$$h_l : \mathbb{R}^3 \to \mathbb{R}^3; \quad \mathbf{X}_{i_c,l} \mapsto h_l\left(\mathbf{X}_{i_c,l}, T_{mc,l}, s_l\right) = \mathbf{X}_{i_m,l},$$

must be specified for each pair individually that maps the triangulated 3D locations from each camera coordinate frame $\mathcal{C}_l$ to a common marker coordinate frame $\mathcal{M}$. Here, each h_l must apply not only rigid body transform $T_{mc,l} \in \mathbb{SE}(3)$ but also a scaling $s_l \in \mathbb{R}_+$ to each point. In the following it will be shown how to obtain the required transformation parameters by construction. This is done separately for both the cross-shaped and the circular marker, since the proposed approach depends on the marker type.

In case of the cross-shaped marker each h_l is supposed to transform the corresponding point set such that $h_l\left(\mathbf{X}_{3_c,l}, T_{mc,l}, s_l\right) = [0,0,0]^\top$, $h_l\left(\mathbf{X}_{0_c,l}, T_{mc,l}, s_l\right) = [d_{03},0,0]^\top$ and

$$\frac{h_l\left(\mathbf{X}_{5_c,l}, T_{mc,l}, s_l\right)}{\|h_l\left(\mathbf{X}_{5_c,l}, T_{mc,l}, s_l\right)\|_2} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

holds, according to the marker coordinate frame presented in Section 3.2.2. Here, the scale factor s_l can be computed from the previously manually measured reference distance d_{03} as

$$s_l = \frac{d_{03}}{\|\mathbf{X}_{0_c,l} - \mathbf{X}_{3_c,l}\|_2},$$

that scales the coordinates equally to the prescribed absolute units of the marker. For $T_{mc,l}$ both the translation vector $\mathbf{t}_{mc,l}$ as well as the rotation matrix $R_{mc,l}$ have to be constructed. Here the translation vector is simply given by

$$\mathbf{t}_{mc,l} = -\mathbf{X}_{3_c,l},$$

since $\mathbf{X}_{3_m,l}$ lies in the origin of the marker coordinate frame. Finally, the rotation matrix $R_{mc,l} = [\mathbf{r}_{1,l}, \mathbf{r}_{2,l}, \mathbf{r}_{3,l}]^\top$ is constructed as

$$\mathbf{r}_{1,l} = \frac{\mathbf{X}_{0_c,l} + \mathbf{t}_{mc,l}}{\|\mathbf{X}_{0_c,l} + \mathbf{t}_{mc,l}\|_2}, \quad \mathbf{r}_{3,l} = \frac{\mathbf{r}_{1,l} \times (\mathbf{X}_{5_c,l} + \mathbf{t}_{mc,l})}{\|\mathbf{r}_{1,l} \times (\mathbf{X}_{5_c,l} + \mathbf{t}_{mc,l})\|_2}, \quad \mathbf{r}_{2,l} = \frac{\mathbf{r}_{3,l} \times \mathbf{r}_{1,l}}{\|\mathbf{r}_{3,l} \times \mathbf{r}_{1,l}\|_2},$$

with $\mathbf{r}_{1,l}, \mathbf{r}_{2,l}, \mathbf{r}_{3,l} \in \mathbb{R}^3$.

Similar to this, in case of the circular marker each h_l has to transform the point sets

such that $h_l\left(\left(\mathbf{X}_{1_c,l}+\mathbf{X}_{4_c,l}\right)/2, T_{mc,l}, s_l\right) = [0,0,0]^\top$, $h_l\left(\mathbf{X}_{4_c,l}, T_{mc,l}, s_l\right) = [d_{14}/2, 0, 0]^\top$ and

$$\frac{h_l\left(\mathbf{X}_{0_c,l}, T_{mc,l}, s_l\right)}{\|h_l\left(\mathbf{X}_{0_c,l}, T_{mc,l}, s_l\right)\|_2} = \begin{bmatrix}0\\1\\0\end{bmatrix},$$

given the manually measured distance d_{14}. The scaling factor is now computed as

$$s_l = \frac{d_{14}}{\|\mathbf{X}_{1_c,l} - \mathbf{X}_{4_c,l}\|_2},$$

the translation vector is given by

$$\mathbf{t}_{mc,l} = -\left(\frac{\mathbf{X}_{1_c,l} + \mathbf{X}_{4_c,l}}{2}\right),$$

and the column vectors of the transposed rotation matrix $R_{mc,l} = [\mathbf{r}_{1,l}, \mathbf{r}_{2,l}, \mathbf{r}_{3,l}]^\top$ are constructed as

$$\mathbf{r}_{1,l} = \frac{\mathbf{X}_{4_c,l} + \mathbf{t}_{mc,l}}{\|\mathbf{X}_{4_c,l} + \mathbf{t}_{mc,l}\|_2}, \quad \mathbf{r}_{3,l} = \frac{\mathbf{r}_{1,l} \times (\mathbf{X}_{0_c,l} + \mathbf{t}_{mc,l})}{\|\mathbf{r}_{1,l} \times (\mathbf{X}_{0_c,l} + \mathbf{t}_{mc,l})\|_2}, \quad \mathbf{r}_{2,l} = \frac{\mathbf{r}_{3,l} \times \mathbf{r}_{1,l}}{\|\mathbf{r}_{3,l} \times \mathbf{r}_{1,l}\|_2},$$

accordingly.

Now regardless of the marker type, having obtained $T_{mc,l}$ and s_l as described above h_l can then be defined as

$$h_l\left(\mathbf{X}_{i_c,l}, T_{mc,l}, s_l\right) \doteq s_l(T_{mc,l}\tilde{\mathbf{X}}_{i_c,l})_{3\times 1} = \mathbf{X}_{i_m,l},$$

which is applied to all point sets of all image pairs. This results in a set of 3D points $\mathbf{X}_{i_m,l}$ distributed closely around the respective LED location, since in practice each reconstruction is afflicted by an individual measurement error inherited from the previous steps. In order to fuse these point sets into a single location per LED the median values $X_{i_m}^{med}$, $Y_{i_m}^{med}$ and $Z_{i_m}^{med}$ are calculated across all $\mathbf{X}_{i_m,l}$ for each dimension independently. Here computing the median is preferred over the mean, since it is more robust to outliers. The final position estimates of the initialization are then eventually given by the points $\mathbf{X}_{i_m}^0 = [X_{i_m}^{med}, Y_{i_m}^{med}, Z_{i_m}^{med}]^\top$.

Refinement In order the increase the accuracy and reliability of the pose estimation the spatial position measurements $\mathbf{X}_{i_m}^0$ coming from the initialization step are further improved by a subsequent iterative optimization step. Here, the basic idea is to always use the currently estimated model coordinates in order to compute the camera poses for all images relative to them. This set of camera poses is then jointly used in order to obtain a refined model estimate in each iteration (see Figure 3.13). This is done based on the assumption that the more accurate LED locations will in return always lead to more precise camera poses in the next iteration allowing to further refine the spatial model reconstruction. In the context of multi-view stereo this problem is commonly known as *bundle adjustment*. For this, popular previous approaches suggest to jointly update all camera poses and spatial positions by minimizing the 2D projection errors of all points across all images [LA09].

However, here a custom interleaved and decoupled optimization of the camera poses and the marker geometry similar to [LFS13] was developed for this specialized task of marker calibration. This approach allows to include geometric marker constraints as well as the proposed robust pose estimation. Here, the spatial coordinates of the LEDs are updated in the 3D marker space as opposed to the 2D image space. The resulting decoupled optimization procedure is repeated iteratively until convergence as described in detail in the following.

During refinement, each camera pose and each LED position is treated independently. In every iteration k, first all $m-1$ camera poses are computed as the inverse of the respective marker poses

$$T_{mc}^k(t_j) = T_{cm}^k(t_j)^{-1} \in \mathbb{SE}(3),$$

which can each be estimated from a single projection set $\mathbb{M}(t_j)$ based on the given current model positions $\mathbf{X}_{i_m}^{k-1}$, as described in Section 3.2.5. Afterwards, all camera poses $T_{mc}^k(t_j)$ are used to "re-triangulate" each LED location by computing the 3D point $\mathbf{X}_{i_m}^k$ closest to all 3D projection rays

$$\mathbf{o}_m^k(t_j) + \lambda \mathbf{v}_{i_m}^k(t_j),$$

from each camera center $\mathbf{o}_m^k(t_j) = \mathbf{t}_{mc}^k(t_j)$ in direction of each corresponding pro-

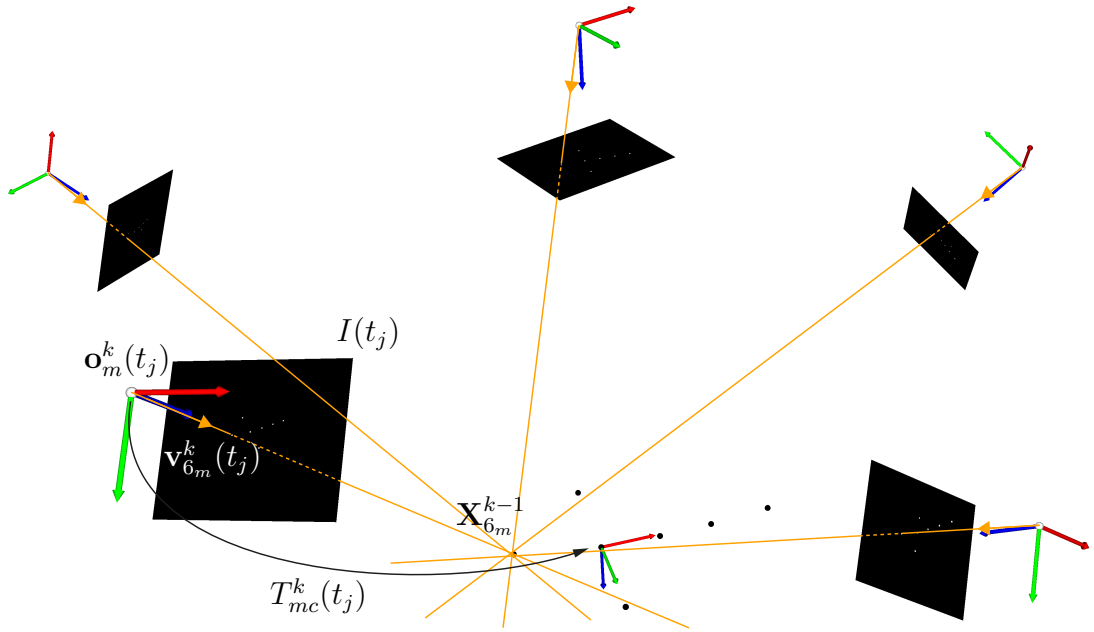


Figure 3.13: A visualization of the iterative marker calibration refinement method with a cross-shaped marker. Given the geometry estimate from the last iteration $\mathbf{X}_{6_m}^{k-1}$, new camera poses $T_{mc}^k(t_j)$ to all calibration images $I(t_j)$ are computed. In this example the projection rays $\mathbf{o}_m^k(t_j) + \lambda \mathbf{v}_{6_m}^k(t_j)$ are shown, that are then used in order to compute the new refined location of $\mathbf{X}_{6_m}^k$ in this iteration. They are depicted in orange with their direction vectors $\mathbf{v}_{6_m}^k(t_j)$ visualized by small arrows.

jection $\mathbf{v}_{i_m}^k(t_j) = R_{mc}^k(t_j)\tilde{\mathbf{x}}_{i_c}(t_j)/\|\tilde{\mathbf{x}}_{i_c}(t_j)\|_2$ in the marker coordinate system. This is done based on the fact, that the distance d of any 3D point $\mathbf{X}$ to such 3D projection ray can be computed as

$$\|\mathbf{v}_{i_m}^k(t_j) \times (\mathbf{o}^k(t_j) - \mathbf{X})\|_2 = d.$$

Now since every $\mathbf{X}_{i_m}^k$ is supposed to minimize the distance to all rays in its corresponding bundle regardless of the actual distance value, this leads to the per camera pose constraint

$$\mathbf{v}_{i_m}^k(t_j) \times (\mathbf{o}^k(t_j) - \mathbf{X}_{i_m}^k) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \tag{3.7}$$

of a linear least-squares problem

$$\sum_{j=0}^{m-1} \|\mathbf{v}_{i_m}^k(t_j) \times (\mathbf{o}^k(t_j) - \mathbf{X}_{i_m}^k)\|_2^2 = 0,$$

for each LED $\mathbf{X}_{i_m}$. By a simple rearrangement equation (3.7) can be written as

$$\hat{\mathbf{v}}_{i_m}^k(t_j)\mathbf{X}_{i_m}^k = \hat{\mathbf{v}}_{i_m}^k(t_j)\mathbf{o}^k(t_j), \quad \text{with} \quad \hat{\mathbf{v}}_{i_m}^k(t_j) \in \mathfrak{se}(3),$$

which provides three equations to a linear system

$$A_i^k \mathbf{X}_{i_m}^k = \mathbf{b}_i^k, \tag{3.8}$$

where

$$A_i^k = \begin{bmatrix} \hat{\mathbf{v}}_{i_m}^k(t_0) \\ \vdots \\ \hat{\mathbf{v}}_{i_m}^k(t_j) \\ \vdots \\ \hat{\mathbf{v}}_{i_m}^k(t_{m-1}) \end{bmatrix} \in \mathbb{R}^{3(m-1)\times 3} \quad \text{and} \quad \mathbf{b}_i^k = \begin{bmatrix} \hat{\mathbf{v}}_{i_m}^k(t_0)\mathbf{o}^k(t_0) \\ \vdots \\ \hat{\mathbf{v}}_{i_m}^k(t_j)\mathbf{o}^k(t_j) \\ \vdots \\ \hat{\mathbf{v}}_{i_m}^k(t_{m-1})\mathbf{o}^k(t_{m-1}) \end{bmatrix} \in \mathbb{R}^{3(m-1)}.$$

In every iteration the system (3.8) is constructed for each LED individually and solved for $\mathbf{X}_{i_m}^k$ using QR decomposition, in order to obtain the new refined location. These will then be used for better pose estimation in the next iteration. In order to preserve the absolute scale of the marker, either $\mathbf{X}_{0_m}^0$ and $\mathbf{X}_{3_m}^0$, in case of a cross-shaped marker, or $\mathbf{X}_{1_m}^0$ and $\mathbf{X}_{4_m}^0$, in case of a circular marker, do not get updated and therefore remain unchanged by the refinement.

The whole process is repeated until the change of the spatial LED locations converges. Here, the error per iteration is measured by

$$e^k = \frac{1}{n} \sum_{I=0}^{n-1} \|\mathbf{X}_{i_m}^k - \mathbf{X}_{i_m}^{k-1}\|_2,$$

and convergence is determined by looking at the change of this error $\Delta e = |e^k - e^{k-1}|$. This means that the algorithm stops if Δe is smaller than a prescribed bound ϵ. In

practice $\epsilon = 0.0001$ was used as a suitable threshold for convergence.

The reliability of the overall marker calibration strategy and the improvement obtained by the refinement step were evaluated experimentally. For this, both a cross-shaped as well as a circular prototype marker of similar size were calibrated repeatedly and the individual sets of LED reconstructions compared to one another afterwards. Since the spatial positions of the LEDs remain the same for each calibration when using the same marker, the standard deviation of the measured spatial positions should ideally be zero across the different calibrations. The results of this experiment are shown in Table 3.1.

	Cross-Shaped Marker					Circular Marker			
init.	$\mathbf{X}^0_{1m}$	$\mathbf{X}^0_{2m}$	$\mathbf{X}^0_{4m}$	$\mathbf{X}^0_{5m}$	$\mathbf{X}^0_{6m}$	$\mathbf{X}^0_{0m}$	$\mathbf{X}^0_{2m}$	$\mathbf{X}^0_{3m}$	$\mathbf{X}^0_{5m}$
X^0_{im}	$76.02 \pm .03$	$37.95 \pm .03$	$-0.08 \pm .14$	$0.25 \pm .15$	$-37.92 \pm .04$	$0.13 \pm .42$	$\mathbf{-28.77 \pm .63}$	$28.12 \pm .48$	$-0.42 \pm .13$
Y^0_{im}	$-0.11 \pm .01$	$-0.04 \pm .01$	$-37.89 \pm .17$	$38.2 \pm .18$	$0.4 \pm .05$	$78.92 \pm .42$	$-73.58 \pm .40$	$-73.17 \pm .35$	$0.03 \pm .07$
Z^0_{im}	$0.08 \pm .02$	$0.21 \pm .02$	$\mathbf{0.21 \pm .40}$	$0.00 \pm .00$	$\mathbf{-11.05 \pm .40}$	$0.00 \pm .00$	$-0.29 \pm .05$	0.38 ± 0.07	$\mathbf{-4.29 \pm 3.68}$
ref.	$\mathbf{X}_{1m}$	$\mathbf{X}_{2m}$	$\mathbf{X}_{4m}$	$\mathbf{X}_{5m}$	$\mathbf{X}_{6m}$	$\mathbf{X}_{0m}$	$\mathbf{X}_{2m}$	$\mathbf{X}_{3m}$	$\mathbf{X}_{5m}$
X_{im}	$76.02 \pm .01$	$37.94 \pm .01$	$-0.09 \pm .04$	$0.2 \pm .04$	$-37.96 \pm .03$	$-0.05 \pm .07$	$\mathbf{-28.51 \pm .06}$	$28.33 \pm .06$	$-0.28 \pm .05$
Y_{im}	$-0.12 \pm .01$	$-0.05 \pm .01$	$-37.93 \pm .09$	$38.14 \pm .09$	$0.38 \pm .02$	$79.08 \pm .11$	$-73.76 \pm .11$	$-73.34 \pm .10$	$-0.04 \pm .03$
Z_{im}	$0.15 \pm .01$	$0.28 \pm .01$	$\mathbf{0.22 \pm .04}$	$0.01 \pm .03$	$\mathbf{-11.34 \pm .03}$	$-0.01 \pm .08$	$-0.28 \pm .08$	$0.37 \pm .08$	$\mathbf{-10.41 \pm .02}$

Table 3.1: The results of the marker calibration experiment where a cross-shaped marker with $d_{03} = 114.4\,\text{mm}$ and a circular marker with $d_{14} = 157.3\,\text{mm}$ were calibrated 30 times using the same camera. Here the mean spatial coordinates as well as their standard deviation in each dimension are shown for each concerned LED of both prototype markers before and after refinement. Cases where the refinement step has reduced the deviation of a coordinate by more than an order of magnitude are highlighted in **boldface**.

The experimental results show that the spatial positions after the initialization are already accurate to several tenths of a millimeter, while the refinement step is then able to reduce the standard deviation by an order of magnitude in most cases. The proposed refinement method was furthermore compared to an established implementation of full sparse bundle adjustment (SBA) [LA09]. It could be shown that the proposed method reduces the standard deviation of the calibration initialization about an order of magnitude while SBA only reduces it by a factor of 2 to 3. To further analyze this difference, experiments have been conducted where the initial estimates of $\mathbf{X}^0_{im}$ were randomly distorted by $\pm 5\,\text{mm}$ for each calibration. Here the

proposed refinement method was still able to robustly reduce the standard deviation to several hundredths of a millimeter while SBA varies about an order of magnitude more. This shows the reliability of our refinement method even for poor initial estimations.

3.2.8 Evaluation

For further experimental evaluation, a single-core C++ implementation of HSRM-Tracking was run on a commodity quad core laptop CPU @ 2.6 GHz. This section starts by presenting a brief runtime performance analysis followed by an extensive comparison of the two different proposed marker patterns with regard to measurement accuracy and reliability of the pose estimation method. These experiments were mainly conducted using a 1280×1024 px USB 3.0 camera[6] with a fixed-focus 6 mm lens and the same two prototype markers that were used to obtain the values in Table 3.1.

Runtime Analysis The performance of the implementation was measured by profiling the computation times for the main processing steps over a thousand frames while moving each marker arbitrarily in front of the camera. Figure 3.14 shows a plot of these timings subdivided into the 2D image processing steps until obtaining the 2D LED locations (Section 3.2.3) and the marker pose estimation including correspondence assignment (Section 3.2.4 and 3.2.5). It shows that regardless of the marker pattern, the average computation time is robustly around (mostly below) 1 ms per frame on a single core. Given a suitable camera, this enables tracking by detection of a single marker at frequencies of up to 1000 Hz. Here, the runtime of the pose estimation is independent of the image resolution. Thus, each additional marker would only increase the runtime by approximately 0.5 – 0.6 ms if not processed in parallel on a multi-core CPU.

The computation time of all the 2D image processing steps mostly depends on the image resolution and is nearly independent of the number of visible markers and the utilized LED pattern. This influence was evaluated in another experiment where

[6] A XIMEA XiQ MQ013MG-CM camera: https://www.ximea.com/

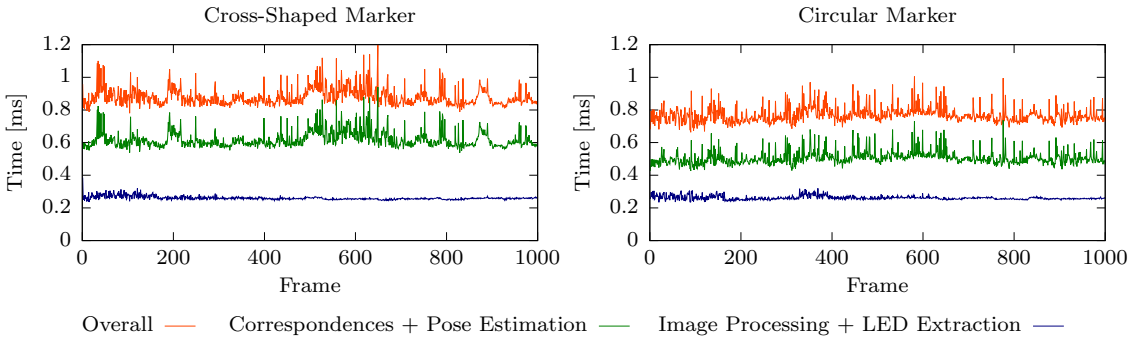


Figure 3.14: Computation times of HSRM-Tracking for both marker designs. All measurements were averaged over 100 runs using two prerecorded sequences of each 1000 frames at full 1280×1024 px resolution. In each sequence a single marker was held in hand and translated and rotated arbitrarily in front of the camera.

cameras with different image sensor sizes were used to measure the average processing time, excluding correspondence assignment and pose estimation (see Table 3.2).

Image resolution	640×480 px	1280×1024 px	2048×1088 px	2048×2048 px
Processing time	≈ 0.1 ms	≈ 0.3 ms	≈ 0.4 ms	≈ 0.9 ms

Table 3.2: Timings of the 2D image processing step for different image resolutions.

It can be seen that the runtime of the image processing steps depends roughly linearly on the number of pixels. This means, that even at a large resolution of four megapixels, the pose of a single marker can still be estimated at more than 500 Hz.

Accuracy Evaluation In the following, a qualitative evaluation of the pose estimation accuracy and reliability is given using both marker patterns in direct comparison. In general, determining the absolute measurement accuracy of HSRM-Tracking is difficult for two main reasons. Firstly, obtaining ground truth pose information for comparison would require another pose estimation system with perfect accuracy that can be trusted as reference. However, if such a system existed then there would be no more need for the proposed solution, which makes ground truth comparison a so-called *chicken-egg-problem*. In [FMSS14] for example a Vicon motion capturing system was used to provide reference measurements. Altough such a system is afflicted by inaccuracies itself and thus cannot be fully trusted, it was not available to

the author of this work and too expensive to buy, only to conduct these experiments. Secondly, the accuracy of a live HSRM-Tracking system does not only depend on the proposed marker patterns and algorithm but also on the quality of the camera calibration, the image sensor and its resolution, the lens and the LEDs as well as the manufacturing of the markers.

Here, it was therefore chosen to perform the accuracy evaluation by conducting several simulated experiments. This allows to exclude the influences of these factors that have not been subject of this thesis and at the same time solve the ground truth data problem. For this, known 3D marker geometry $\mathbf{X}_{i_m}$ is virtually projected into the image plane using different prescribed ground truth 6DOF poses in order to generate a set of "perfect" 2D LED locations $\mathbf{x}'_i = \mathbf{x}_i$ using (2.32). These projections are then used as input for the proposed pose estimation method along with the set of 3D LED locations. Since everything is known in this scenario, the resulting pose estimates would also be perfect up to numerical restrictions. However, this strategy also allows to purely evaluate the influence of uncertainties introduced by the proposed image processing algorithm on the pose accuracy with respect to the marker pattern, when including them in the simulation.

For this, on the one hand, the uncertainty of the extracted 2D LED locations is simulated by adding 2D Gaussian noise to the generated 2D locations such that $\mathbf{x}'_i \neq \mathbf{x}_i$ based on the observations presented in Figure 3.4. There it can be seen that the standard deviation of this 2D noise depends roughly linearly on the markers distance to the camera i.e. the t_z-translation entry of the pose. Thus in the following experiments the standard deviation of this 2D Gaussian noise is computed as $\sigma = 0.003568295 + t_z \cdot 0.00000429485$, being a linear approximation of the measured results in Figure 3.4.

On the other hand, the uncertainty of the proposed marker calibration is also included in the simulation. The results of Table 3.1 show that despite the high precision of the calibration results, the measured 3D LED locations will not be perfectly known and thus also have a negative influence on the pose estimation accuracy. Therefore, all experiments were conducted once with the ground truth LED locations (being the mean measured coordinates of Table 3.1), once using the ground truth locations plus the standard deviation and once minus the determined standard

deviation shown in Table 3.1. These three configurations for the marker geometry, along with the noisy 2D projections (always generated from the ground truth geometry) were used as the input for the pose estimation. Here, the projections of the LEDs are computed with the intrinsic matrix

$$K = \begin{bmatrix} 1265.32 & 0 & 625.87 \\ 0 & 1265.86 & 513.11 \\ 0 & 0 & 1 \end{bmatrix},$$

which corresponds to the real camera used to calibrate the two markers beforehand.

In order to determine the reliability of the measurements, at each pose configuration a thousand poses were estimated, each afflicted by individual 2D noise. These measurements were then used to compute the mean and the standard deviation for all six degrees of freedom represented in form of the translation vector components t_x, t_y and t_z as well as the three Euler angles α, β and γ. In case of monocular pose estimation the translation along the optical axis t_z as well as the two outer image plane rotation angles α and β are the most critical. These can typically be estimated with about an order of magnitude less accuracy than the other three in-plane motion parameters t_x, t_y and γ. For this reason and because covering the entire 6DOF pose space comprehensively is cumbersome and redundant, this evaluation is restricted to the three most meaningful and exemplary experiments focussing on the critical outer image plane motion.

In the first experiment the absolute measurement precision of the translation t_z along the optical axis of the camera is evaluated. Starting at $t_z = 300\,\text{mm}$ the marker was virtually moved in $1\,\text{mm}$ steps up to $t_z = 2000\,\text{mm}$ and evaluated at each location in between. The absolute accuracy was then determined by comparing the mean measured t_z to the prescribed ground truth value for all three LED geometry configurations and each marker as seen in Figure 3.15.

The results show that given a perfect camera calibration, HSRM-Tracking is able to measure the absolute position of either marker robustly with an accuracy of about $\pm 1\,\text{mm}$ (including the standard deviation of the pose parameters as well as the worst marker calibration error) even at 2 meters distance to the camera. It can furthermore be seen that the depth estimation in case of the circular marker

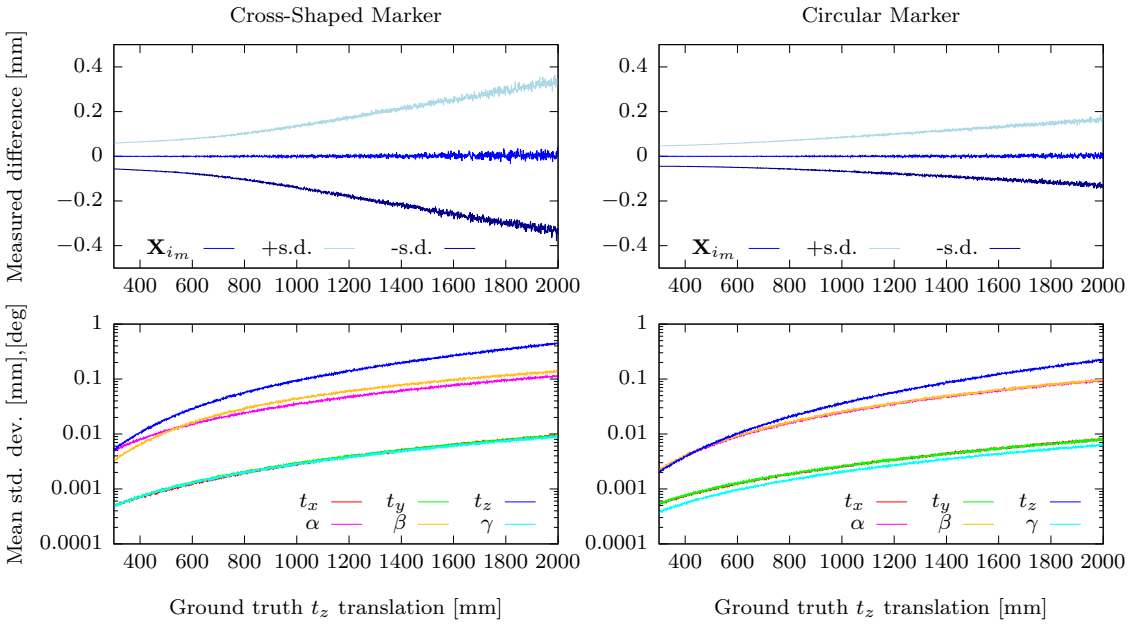


Figure 3.15: Results of the simulated depth estimation accuracy experiment where both markers were translated along the optical axis of the camera by changing t_z of the prescribed ground truth marker pose. Top: The difference between the mean estimated value for t_z and the ground truth. These plots show the results for using the ground truth LED coordinates $\mathbf{X}_{im}$ as well as those distorted by adding ("+s.d.") and subtracting ("-s.d.") the standard deviation of the marker calibration in each dimension (see Table 3.1). Bottom: The corresponding measured standard deviations of all six pose parameters at each evaluated distance.

is about twice as accurate compared to the cross-shaped alternative, with much lower variance. This can be explained by looking at the distribution of the LEDs in each pattern. The cross-shaped marker consists of seven LEDs that are mostly distributed along the X_m axis of the marker. The four used to compute its cross-ratio are furthermore co-linear which means two of them do not contribute to increase the pose estimation accuracy. The circular marker pattern consists of only six LED, however not three of them are co-linear and they are overall more evenly distributed within the X_mY_m-plane, meaning they all provide unique information to the pose estimation. Regardless of the pattern, these plots also show that the uncertainty of t_z, α and β is as expected about an order of magnitude higher than of t_x, t_y and γ.

Next, the rotational accuracy of the pose estimation is analyzed. For this, the marker

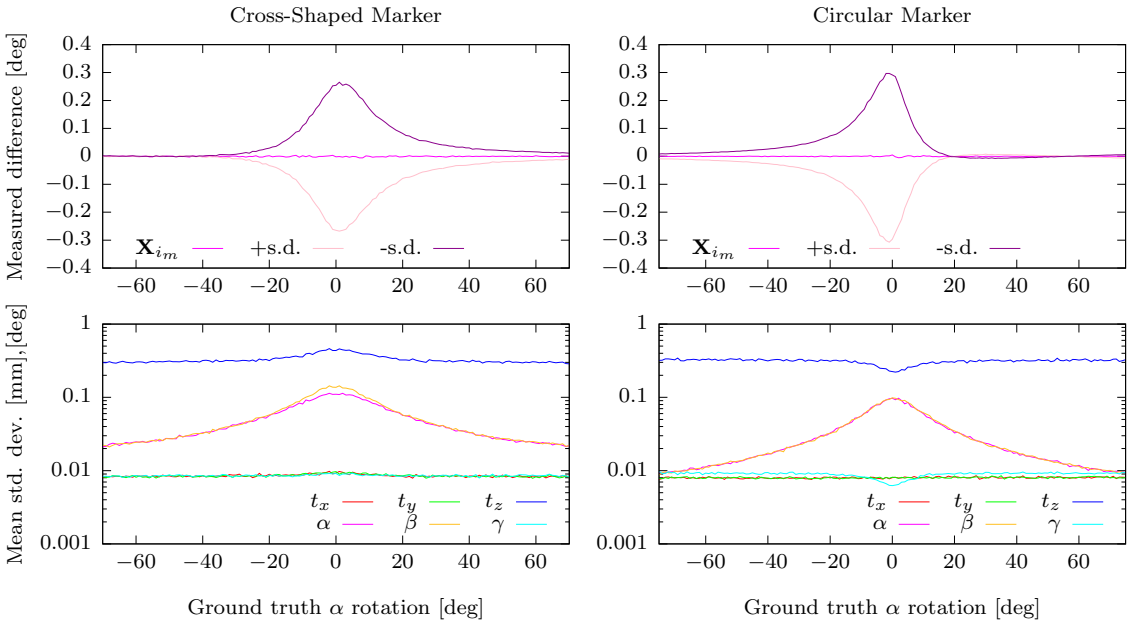


Figure 3.16: Results of the simulated rotational accuracy experiment where both markers were rotated around their X_m-axis by the angle α at $t_z = 2000$ mm.

was placed at a distance of $t_z = 2000\,\text{mm}$ to the camera and rotated once only around its X_m-axis and once only around its Y_m-axis in 1° steps within angle ranges where no self-occlusions of the projected LED pattern would occur in practice. Thus, the cross-shaped marker was rotated by $\alpha \in [-70°, 70°]$ around its X_m-axis and $\beta \in [-60°, 80°]$ around its Y_m-axis. Accordingly, due to its more symmetric shape, the circular marker could be evaluated for $\alpha \in [-75°, 75°]$ and also $\beta \in [-75°, 75°]$. In analogy to the previous t_z translation experiment, the absolute accuracy was then determined by computing the difference between the mean measured α and β and the prescribed ground truth angles for all three LED geometry configurations, as shown in Figure 3.16 and Figure 3.17.

The resulting plots show that for both markers the rotational measurement error is robustly within ±0.5° even at such a large distance to the camera. It can furthermore be seen that in case of the cross-shaped marker the largest angular error around the Y_m-axis is approximately three times lower than around the X_m-axis. This is expected because the LEDs within this pattern are more distributed in X_m than in

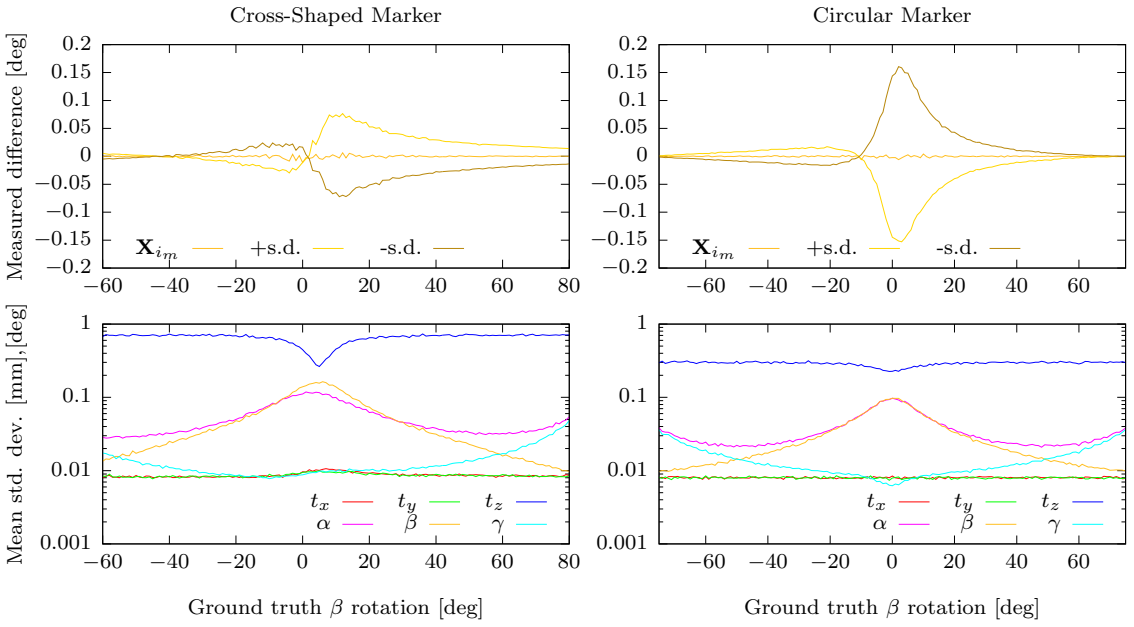


Figure 3.17: Results of the simulated rotational accuracy experiment where both markers were rotated around their Y_m-axis by the angle β at $t_z = 2000$ mm.

Y_m direction and thus provide more information for estimating the rotation around the Y_m-axis. Therefore, in case of the circular marker, where the LEDs are better distributed in the $X_m Y_m$-plane, the largest magnitude of the two angular errors is more alike.

Lastly, the variation of the 2D cross-ratio that is computed from the noisy 2D projections $\mathbf{x}'_i$ is evaluated with respect to perspective changes for both markers. This is important because in case of multi-marker tracking the cross-ratio measured in the image plane is used to identify individual markers by comparing it to the expected cross-ratio given by the corresponding known spatial LED coordinates (see Section 3.2.6). Thus, the maximal variation of the 2D cross-ratio indicates a lower bound for how similarly two markers can be constructed, in order to still be distinguishable in the same image. Here, for example the utilized cross-shaped marker has a true cross-ratio of $\mathcal{X}_|(\mathbf{X}_{0_m}, \mathbf{X}_{3_m}; \mathbf{X}_{2_m}, \mathbf{X}_{1_m}) = 3.991$ and the circular marker of $\mathcal{X}_\bigcirc(\mathbf{X}_{1_m} - \mathbf{X}_{0_m}, \mathbf{X}_{4_m} - \mathbf{X}_{0_m}; \mathbf{X}_{2_m} - \mathbf{X}_{0_m}, \mathbf{X}_{3_m} - \mathbf{X}_{0_m}) = 1.882$.

The projected 2D cross-ratio is mostly affected by outer image plane rotation of

the marker. For this evaluation, it was thus computed from every noise afflicted pattern projection used within both previous rotational accuracy experiments for each marker. For the cross-shaped marker, this resulted in a maximal difference to the expected cross-ratio of 0.158 and an overall standard deviation of 0.022. Compared to this, in case of the circular marker this maximal difference was only 0.065 whereas here the standard deviation of the measured cross-ratio was 0.015.

3.3 Natural Feature-based Approaches

When no marker and no additional light sources are used, the image points $\mathbf{x}'_i$ have to be computed passively from so-called *natural features* visible in the scene. The two most popular and commonly used natural features are so-called *point* and *edge features*. The following sections will briefly explain how these can be computed from camera images and how they can be used for pose estimation from explicit 2D-to-3D correspondences. It will also be given an overview of related methods that have successfully used these features in the past as well as of the limitations and constraints connected to them. Due to the large amount of research that has been conducted over the years in this context, in this work, object pose estimation from natural features was considered "sufficiently studied" for the fairly limited scenarios and objects it is suitable for. Since here the focus lies on scenarios where natural features typically fail, they have not been further investigated and are therefore mostly included for completeness.

3.3.1 Point Features

A large variety of object pose estimation methods based on natural point features have been proposed in the past, e.g. [VLF04], [Ros06], [HBN07], [PLW08], [KLW10], [HCH10], [ÖCLF10], [WRM+10] or [PSK+11] to enumerate only some of them. These features are commonly extracted in two steps. The first is called the *feature detection* step where the 2D locations of potential *keypoints* or *interest points* are computed in the image (see Figure 3.18). Here, the intensities in the image patch surrounding a useful keypoint should yield a unique texture, that can be easily

(a) (b) (c)

Figure 3.18: Natural point feature detection on a well textured object. Here a color image of a tea packing is used as an example (a). Since most feature detectors are designed for single channel intensity images, it is first converted to a grey scale version. The detected FAST feature points [RD06] are depicted in blue (b). Also the local image patch (marked by a green outline in (b)) around one of the detected points is shown in a detailed view, in order to give a typical example for an image corner (c).

recognized in other frames. Such points are typically located at image corners, which are defined as image locations with two different dominant local intensity gradient directions (e.g. the intersection of two edges). But they can also be located at an isolated point of a local intensity maximum or minimum. Mathematically these points can be identified by checking if both eigenvalues of the local Hessian matrix $\nabla^2 I(\mathbf{x}')$ of the image at a potential location $\mathbf{x}'$ are large. Here, the Hessian matrix (or a suitable approximation thereof) is commonly computed as

$$\nabla^2 I(\mathbf{x}'_i) = \frac{1}{|\mathcal{P}(\mathbf{x}')|} \sum_{\mathbf{x} \in \mathcal{P}(\mathbf{x}')} \begin{bmatrix} \frac{\partial^2 I(\mathbf{x})}{\partial x \partial x} & \frac{\partial^2 I(\mathbf{x})}{\partial x \partial y} \\ \frac{\partial^2 I(\mathbf{y})}{\partial x \partial x} & \frac{\partial^2 I(\mathbf{y})}{\partial x \partial y} \end{bmatrix} \in \mathbb{R}^{2\times 2},$$

by averaging second-order derivatives over a local image patch $\mathcal{P}(\mathbf{x}') \subset \Omega$ centered at $\mathbf{x}'$, where $|\mathcal{P}(\mathbf{x}')|$ gives the number of considered (pixel) locations within $\mathcal{P}(\mathbf{x}')$. Some of the most popular keypoint detectors are the Harris-Stephens detector [HS88], the Shi-Tomasi detector [ST94] and more recently the so-called *FAST features* [RD06].

The second step is called *feature description*, where a high dimensional so-called *feature vector* is computed for each previously detected keypoint. These feature vectors are designed to be ideally unique fingerprints of the interest points which

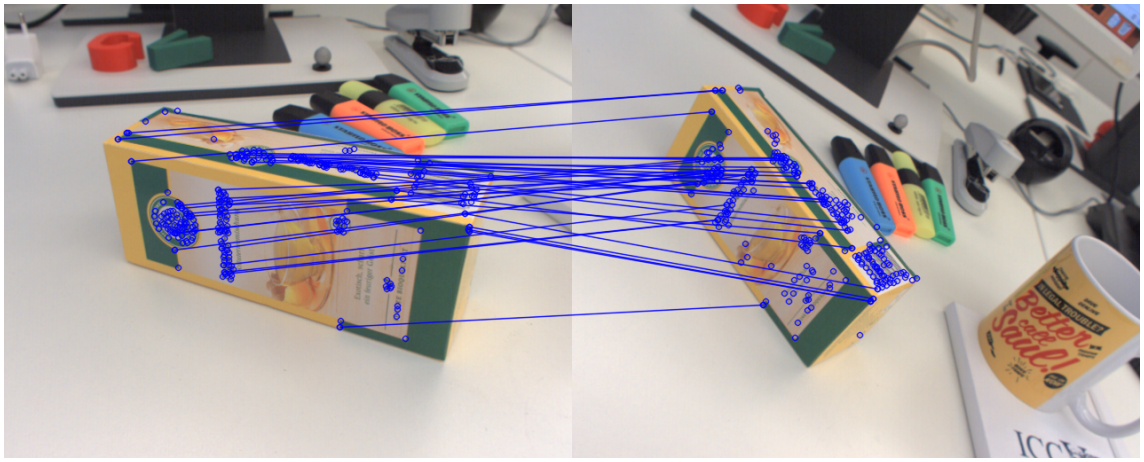


Figure 3.19: An example of matching corresponding feature points across two images of the same object under different perspectives. Here feature matches between the image of Figure 3.18 (left) and another image of the same tea box (right) are depicted by blue lines that connect corresponding feature points. Matched and unmatched detections are shown as blue circles in both images.

allow to re-identify them in other images. They are typically computed from the intensities in again a local image patch (usually a little larger than $\mathcal{P}$) centered around the keypoint location. This description should be invariant to rotation and translation of the camera, scale (i.e. distance to the camera) and illumination in order to be robustly recognizable in other images under different perspectives and illumination conditions (see Figure 3.19). One of the first and probably still the most popular feature descriptor is called *SIFT* (scale invariant feature transform) [Low99, Low04], which was originally proposed for the task of object pose detection. It marked the beginning of a diverse and still ongoing series of succeeding speeded-up, simplified, modified and improved feature descriptors, such as *SURF* [BTG06], *BRIEF* [CLSF10], *ORB* [RRKB11], *BRISK* [LCS11], *FREAK* [AOV12], *KAZE* [ABD12] and *LIFT* [YTLF16] to again only name some of more popular ones.

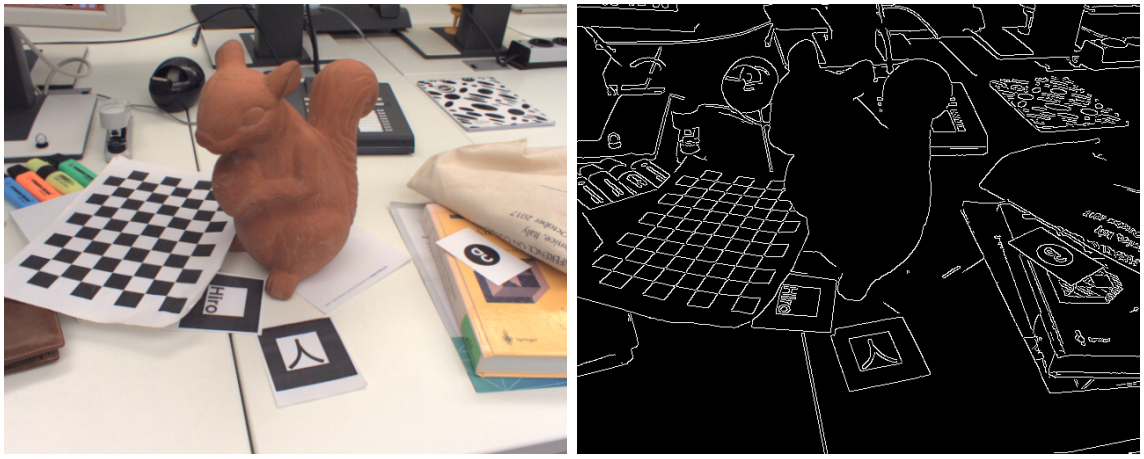
Given such a discriminative descriptor, corresponding feature points in two different images of the same object or scene are then commonly computed by performing a nearest neighbor search in the space of all feature vectors. This step is usually called *feature matching*. In case of object pose estimation these descriptors get attached

(a) (b)

Figure 3.20: In case of textureless or weakly-textured objects (here the squirrel figure) very few feature points get detected which are located typically at their contour. Hence the corresponding feature descriptors are built in large parts from the intensities in the local background region, which immediately change whenever the object or the camera move. Therefore, they cannot be matched robustly (a). When the image becomes blurred due to motion blur significantly less (or sometimes no) features are detected resulting unstable and inaccurate matching and consequently pose estimation (b).

to the respective 3D model points on the object's surface, which then enables to determine the required 2D-to-3D point correspondences between an image and the model. Here, the feature matching step is often directly coupled with the pose estimation itself by incorporating a RANSAC (random sample and consensus) [FB81] approach in order to detect reject outliers, i.e. false correspondences.

Regardless of the employed descriptor, using natural point features requires the objects' surfaces to be sufficiently textured since their detection is based on strong intensity gradients. In practice this constraint however significantly limits the variety of qualified objects (see Figure 3.20). This is why the performance of related methods is usually demonstrated for things like book covers, paintings, postcards, packings or in general any objects with text or graphics printed on them. The other main drawbacks of such features are that they struggle with motion blur, defocus of the camera or low lighting conditions, causing the intensity gradients to flatten or disappear (see again Figure 3.20). Furthermore, the number of visible and distinguishable feature points directly depends on the size of the projected object region in the image. Methods relying on natural point features therefore quickly become unstable with increasing distance of the object to the camera, causing it to appear



(a) (b)

Figure 3.21: An image of a textureless squirrel figure in a cluttered scene (a). The result of the Canny edge detector [Can86] from the given image (b).

smaller in the image and the fine detail corner structures to vanish.

3.3.2 Edge Features

Another popular category of visual pose estimation methods based on natural features use an edge representation of the camera image [CC12, PT11, CMC03, DC02, MBC01, Har93]. These are assumed to yield a reduced abstraction of the object's contour and the inner structure of the object (especially in case of angular objects). Thus, such edge-based approaches work especially well with weakly textured or textureless objects, in contrast to the previously discussed point features. Such an edge representation is usually a binary transformation of the original image, where each pixel is either considered part of an edge or not (see Figure 3.21). In the simplest form this can be decided by checking whether the magnitude of the local gradient per pixel $\|\nabla I(\mathbf{x})\|_2$ is greater than a predefined threshold. A widely used and more advanced approach is the Canny edge detector [Can86].

Related methods usually represent the object by either again a set of 3D points or a set of 3D line segments. Accordingly, a synthetic projection of this model yields either a set of 2D points or 2D lines that can be compared to the computed edge

transformation of the camera image. For pose estimation then either point-to-point, point-to-line, or line-to-line correspondences are established in the image plane. In all cases this again leads to a respective quadratic cost function to be minimized in order to obtain the object's pose.

As for the approaches based on natural point features, relying on intensity edges introduces some significant drawbacks as well. Their detection also struggles with blurring effects and large distances to the camera, for the same reasons as before. In addition edge-based methods are usually prone to local minima in cases of cluttered backgrounds (see again Figure 3.21). There have been some methods proposed that combine point and line features for improved robustness [AD02, VLF04, RD05, PM06], but the main problems connected to them persist.

3.4 Summary

In this chapter, different approaches to point-based pose estimation have been described. Firstly, an active marker-based tracking by detection approach, called HSRM-Tracking, was presented that was developed within the course of this dissertation. The proposed method provides an easy to use *pick up and play* and low-cost visual 6DOF pose measurement tool, that can currently be considered state of the art for monocular pose estimation in terms of accuracy and runtime performance.

The approach can be used with two different novel nearly co-planar infrared LED marker designs that allow to identify each LED projection from a single image. Both markers encode a unique ID based on a geometric cross-ratio and can thus also be used for identifying and tracking multiple markers simultaneously. Here not only the cross-shaped marker that was originally proposed in [TSSS15] but also a new circular marker have been introduced and evaluated in direct comparison. The experimental results showed that the pose and the cross-ratio ID of the circular marker can be estimated with even higher accuracy than that of the original cross-shaped marker due to its overall better distribution of the LEDs.

It has furthermore been shown how a single non co-planar point is sufficient to robustly resolve the common pose ambiguity that all purely co-planar markers suffer

from. In order to guarantee high measurement pose accuracy for HSRM-Tracking a novel semi-automatic marker calibration algorithm has been developed and presented. It can be used to reconstruct the 3D locations of the marker LEDs with high precision, based on a small set of images and a single manual measurement of the absolute scale.

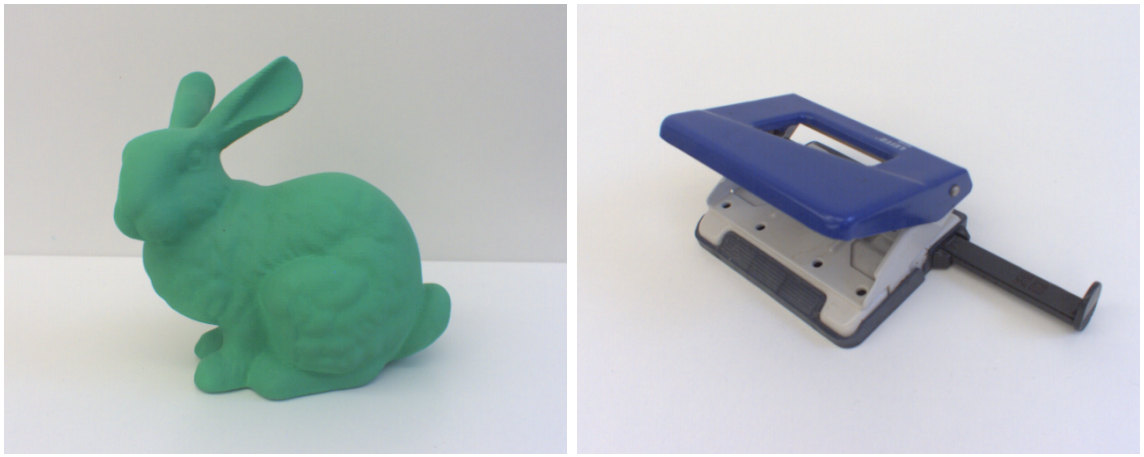
In addition to his, another category of point-based methods has been introduced, that utilizes so-called natural features to enable passive pose estimation. Here it has been explained how both such point and edge features can in principle be detected and matched across different images. However, it was also discussed how these methods suffer from significant drawbacks with respect to their robustness towards several common conditions, which make them not an ideal choice for object pose estimation.

4

Region-based Pose Estimation

This chapter deals with the general concept of estimating the pose of an object based on its 2D silhouette i.e. the projected object region in the image. Most parts of the following sections describe the novel approach that was developed within the course of this dissertation and originally presented in our publications [TSS16, TSS17] and [TSSC18]. There we proposed an improved region-based pose estimation method that is superior to comparable previous solutions, in cases of cluttered backgrounds, heterogeneous objects and partial occlusions, due to a novel statistical image segmentation model. The second main contribution we developed is a new numerical optimization scheme derived for the utilized region-based cost function that significantly improved the convergence properties and thus the tracking robustness and accuracy, especially in cases of fast rotational motion and scale changes. The overall system can furthermore be considered state of the art in terms of runtime performance, since it is currently the only region-based solution capable of tracking the 6DOF poses of multiple objects in real-time. Lastly, the third contribution is a new complex dataset dedicated to the task of monocular object pose tracking which was made available to the community. To our knowledge, it is the first to address the common and important scenario in which both the camera and the objects are moving simultaneously in cluttered scenes.

The following chapter starts by introducing the basic idea of region-based pose es-



(a) (b)

Figure 4.1: Two examples for weakly-textured objects. Here the 3D printed bunny figure is completely green i.e. homogeneous (a) while the hole puncher's surface is partially blue, grey and black i.e. heterogenous (b).

timation (Section 4.1) and then moves on to give an overview of related methods and how the proposed solution builds upon and extends them (Section 4.2). Next, the utilized cost function is introduced (Section 4.3) and corresponding statistical image segmentation models (Section 4.4) as well as an efficient optimization strategy (Section 4.5) are derived. Based on this, the resulting hybrid pose tracking and detection approach is explained (Section 4.6) followed by an extension to multiple objects (Section 4.7). Furthermore, an extensive evaluation of the proposed approach in comparison to other methods is given (Section 4.8), including a popular public dataset for pose detection as well as our own pose tracking dataset. The chapter concludes with a summary discussing advantages and drawbacks of our region-based solution (Section 4.9).

4.1 Outline

As introduced in Section 3.3 for estimating the pose of rigid objects without the use of artificial markers or active lighting, natural image features such as corners and edges have been popular. However, as also previously explained, these are prone to

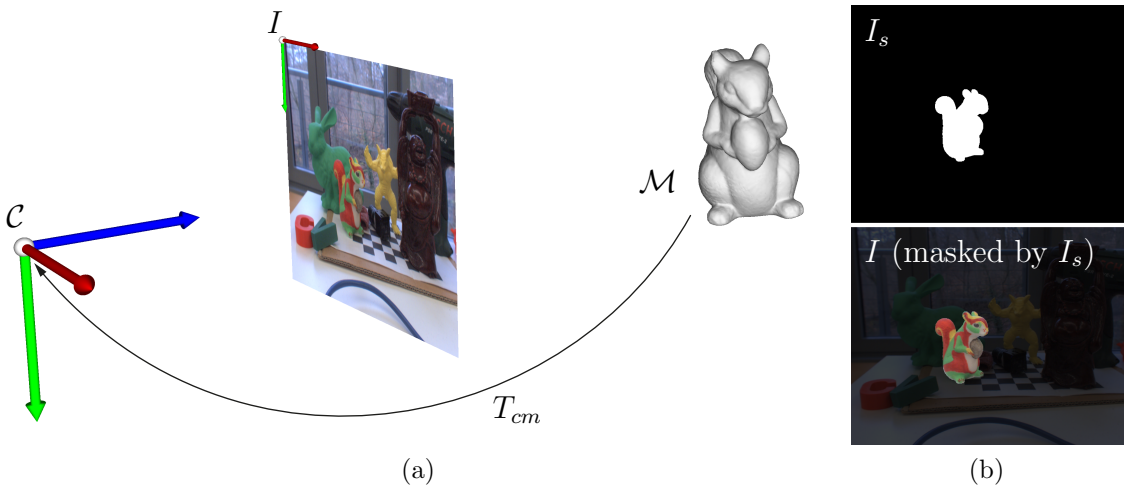


Figure 4.2: An overview of the region-based pose estimation setting. The estimated object pose T_{cm} is used to project the 3D model into the image plane and align it with the object region extracted in the current camera image I (a). The resulting silhouette rendering I_s provides a pixel-wise segmentation mask of I that is then used to update the segmentation models (b).

fail under a variety of common conditions. Here the most significant drawback is that approaches relying on these features require the objects' surfaces to be sufficiently textured. This significantly limits the variety of suitable objects in human made environments. In the following, such weakly-textured or textureless objects are referred to as either *homogeneous* or *heterogeneous*. Here, homogeneous objects are those with a single dominant surface color (see e.g. Figure 4.1 (a)) whereas heterogeneous objects are multi-colored without displaying useful local gradient features (see e.g. Figure 4.1 (b)) .

Since the appearance of such objects is characterized by their silhouette, in recent years so-called *region-based* approaches have been introduced and gained popularity [RBW07, DSYT08, BRGC10, SRBW12, PR12, HH16, TSS17]. They are potentially suitable for a large variety of objects in complex scenarios regardless of local intensity gradients i.e. their texture and thus overcome most of the problems connected to the mentioned features while also being completely passive. These approaches assume discriminative image statistics between the object and the background image region. Based on a suitable statistical appearance model as well as a

3D shape prior (usually in form of a 3D model of the object), here pose estimation essentially works by aligning two silhouettes. Here, the target is the extracted object's silhouette in the current image using a segmentation model while the other is rendered synthetically from the shape prior parametrized by the sought pose. The discrepancy between these two shapes is then minimized by changing the pose parameters used for the synthetic projection. In return, given the object's pose in the current frame, the rendered silhouette provides an accurate pixel-wise segmentation mask that is typically used for updating the foreground and background statistics of the segmentation model, in order to dynamically adapt to scene changes (see Figure 4.2). Therefore, if the pose is given in the first frame of an image sequence, it allows to initialize the statistical model. Pose tracking then is performed recursively in an interleaved manner by first estimating the pose based on that in the previous frame and then updating the segmentation model afterwards using the mask information in the current frame. Here, the initial pose is typically obtained by pose detection or manual alignment, resulting in a tracking and detection hybrid system, as explained in Section 1.2 of the introduction.

4.2 Background and Related Work

When only using a single regular RGB camera, region-based methods relying on statistical level-set segmentation [CRD07] are currently achieving state of the art performance for the task of 6DOF object pose tracking. Early region-based approaches were not real-time capable [RBW07, BRGC10, SRBW12] but already showed the vast potential of the general strategy by presenting promisingly robust results in many complex scenarios. In these works image segmentation was based on level-sets together with pixel-wise likelihoods used to explicitly extract the object's contour in each camera frame. Here, pose estimation was based on an ICP approach by solving a linear system set up from 2D-to-3D point correspondences between the extracted contour and the 3D model. These correspondences are re-established after each iteration in the 2D image plane to the evolving synthetic contour projection.

In [PR12] the authors presented PWP3D, the first region-based approach that achieved real-time frame rates (20–25 Hz) by relying heavily on GPGPU accelera-

tion. Here, pose estimation is performed using a pixel-wise gradient-based optimization scheme similar to the variational approach suggested in [DSYT08]. But instead of separately integrating over the foreground and background region, PWP3D uses a level-set pose embedding in a cost function similar to the very early methods above in order to simplify computations and make it real-time capable. Additionally, based on the idea presented in [BR08] the previously proposed pixel-wise likelihoods were exchanged for pixel-wise posterior probabilities which have been shown to provide a wider basin of convergence.

There have been several successive works recently that build on the general concept of PWP3D. These mostly address two main potential improvements of the original algorithm. The first being the first-order gradient descent used for pose optimization, involving four different fixed step sizes that have to be adjusted experimentally for each object individually. It also uses a fixed number of iterations to maintain real-time performance and thus suffers from commonly related convergence issues, as explained in Section 2.3.1. This optimization was replaced in [TSS16] with a second-order so-called Gauß-Newton-like optimization where the Hessian matrix is approximated from first-order derivatives. Furthermore, PWP3D originally parametrized the pose with a unit quaternion for rotation and a decoupled translation vector. This was also substituted by linearized twists. This strategy overall vastly enhanced the convergence properties resulting in significantly increased robustness towards fast rotations and scale changes, without the need for choosing step sizes. Furthermore, by performing this optimization in a hierarchical coarse-to-fine manner the overall runtime of the proposed mainly CPU-based implementation (it only uses OpenGL for rendering) could be reduced to achieve frame rates of 50–100 Hz for a single object on a commodity laptop. However, in [TSS16] the optimization strategy was discovered from empirical studies and thus not properly derived analytically.

This Gauß-Newton-like optimization was shortly after also adopted in [KTIN17] which directly builds upon [TSS16]. Here, an extended cost function with respect to the depth modality of an RGB-D device is derived in order to improve on both the robustness of the pose estimation as well as the object segmentation in cluttered environments. To obtain the object pose it was suggested to combine the Gauß-Newton-like approach for the RGB-based term with a straightforward Gauß-Newton strategy for the depth-based term in a joint optimization.

Another CPU-based approach was presented in [PKMR15] that achieves around 30 Hz on a mobile phone by using a hierarchical Levenberg-Marquardt optimization strategy for the translation parameters and approximating the level-set related computations. However, the main speed-up was enabled by including the phone's gyroscope to obtain the rotation estimate that is only corrected for drift every tenth frame by a single gradient descent step. Due to this sensor fusion the method presented in [PKMR15] can technically not be considered as a monocular solution and is furthermore restricted to application scenarios in which the phone moves around a static object.

The second main disadvantage of the original PWP3D method is the rather simple segmentation model based on global foreground and background color histograms which is prone to fail in cluttered scenes. Therefore, the authors of [ZWS+14] introduce a boundary constraint for improvement, which is, however, not real-time capable. In [HH16], based on the idea presented in [LT08], a localized segmentation model was proposed that also relies on pixel-wise posteriors but uses multiple local color histograms to better capture spatial color variations of the objects. However, in [HH16] this approach was neither evaluated for pose tracking in video sequences nor shown to be real-time capable. A segmentation strategy similar to the local color histograms used in [HH16] was presented within a contour edge-based approach in [SPP+14] and further improved in [WWZ+15]. Although it performs well in cluttered scenes, the approach struggles with motion blur and defocus and is limited to slow movement for real-time use.

All of the aforementioned methods are strictly designed for tracking and do not provide a solution for pose detection. While in case of well-textured objects, again computing and matching natural feature descriptors can be used to obtain an initial pose from a single image, this problem has been tackled in several other ways in the past in order to provide a more generally applicable solution to it. For real-time applications, pose detection based on 2D template matching [HLI+10, HHC+11b, HLI+12, HCI+12, RT13, KHKH16] has been popular and yielding the best results for many years. Here, the templates are projections of the model at varying perspectives. Probably the most popular and still the most generic template-based method for real-time use is LINE-2D, introduced in [HCI+12] and improved in [RT13]. Here, both the input image and the templates are transformed into so-called *gradient re-*

sponse maps by computing the dominant orientations of RGB intensity gradients.

LINE-2D and similar approaches are usually demonstrated in scenarios where the objects are assumed to be standing or lying on a surface. This allows to only include the upper hemisphere for outer image plane rotations and a small range of in-plane rotation during template generation, instead of the full shpere that is needed in case of e.g. handheld objects. In addition, the pose accuracy of these methods is constrained to the resolution of the templates and they do not incorporate a solution for pose refinement or tracking.

Another more recent approach to pose detection using RGB images is presented in [BMK+16] as an improvement of [BKM+14]. It is based on learning so-called *object coordinates* using random forests. Although this approach outperforms the previously suggested template matching strategies, its runtime performance is far away from real-time capable.

Also very recently and in parallel to this work, the first monocular pose detection methods [CRV+15, PAF+16, MAFK17, RL17, KMT+17] using so-called *deep learning* [GBC16] strategies have been developed. Based on deep convolutional neural networks (CNNs), specifically trained for either one or multiple objects, these approaches achieve state of the art results in public datasets compared to *classical methods* and even outperform those using RGB-D data. Initially, the two main drawbacks related to these deep learning methods were, however, that they require a couple of thousand (in parts manually) labeled training images for each new object as well as a powerful GPU to be real-time capable. While in the most recent work [KMT+17] it has been shown that the cumbersome manual labeling step can be efficiently replaced and fully automated by using only synthetically rendered images of the object for training, the strong hardware dependence currently remains.

In the latest work on region-based pose estimation [TSS17] presented a real-time capable implementation of [HH16] in combination with the ideas of [TSS16] and was the first to extend region-based pose tracking by a strategy for pose detection. The core novelty of this approach is to attach local color histograms to the object's surface. This allows to enforce temporal consistency within each of them which improves the robustness of pose tracking in case of dynamic occlusion, motion of both the camera as well as the object and light changes in cluttered scenes superior

to other methods. It has also been shown that the resulting temporally consistent, local color histograms (called *tclc-histograms*) form a novel object descriptor that can be used for pose detection within a template matching strategy, including the full sphere for outer plane and 360° for in-plane rotation. Here, a unique similarity measure is used for both template matching and pose optimization. This has not been previously addressed by other level-set-based pose estimation approaches. For any new object, the descriptors can be trained online within a couple of seconds moving a handheld object in front of a camera. During this training stage, it is already capable to recover from accidental tracking loss.

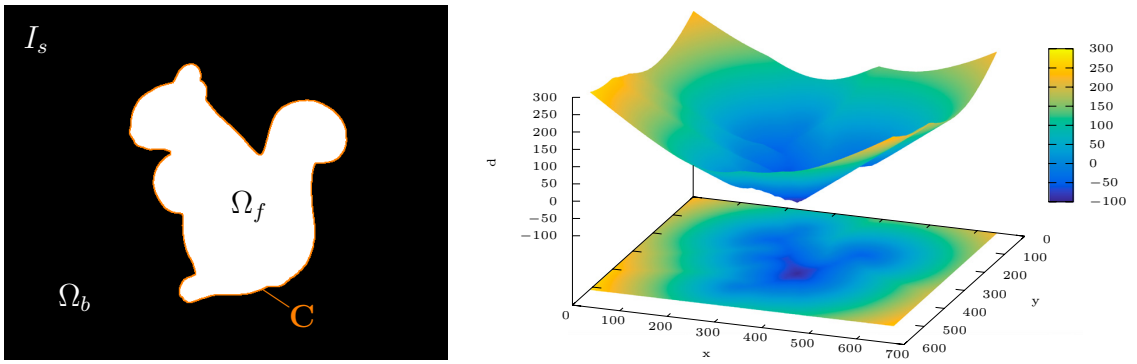
4.3 A Region-based Cost Function

For region-based pose estimation each object is represented by a given 3D model in form of a triangle mesh with corner vertices $\mathbf{X}_{i_m}$, $i = 1, \ldots, n$ (as shown in Section 2.2.4). The approaches described in the following rely on RGB color images denoted by $I : \Omega \to [0, 255]^3 \subset \mathbb{Z}^3$, assuming 8-bit quantization per intensity channel. Projecting a 3D model into the image plane using e.g. OpenGL rendering, results in a binary silhouette mask denoted by $I_s : \Omega \to \{0, 1\}$ that yields a contour $\mathbf{C}$ splitting the image into a foreground region $\Omega_f \subset \Omega$ and a background region $\Omega_b = \Omega \setminus \Omega_f$ (see Figure 4.3). The approach presented here is essentially based on statistical 2D image segmentation [CRD07]. As commonly done in this context, the object's silhouette is implicitly represented by a so-called *shape-kernel* $\Phi(\mathbf{x})$. It is a level-set embedding of the object's 2D shape, with $\Phi(\mathbf{x}) > 0, \forall \mathbf{x} \in \Omega_b$ and $\Phi(\mathbf{x}) \leq 0, \forall \mathbf{x} \in \Omega_f$ such that the zero-level line $\mathbf{C} = \{\mathbf{x} | \Phi(\mathbf{x}) = 0\}$ gives its contour i.e. the boundary between Ω_f and Ω_b (see Section 4.3.1 for details).

For shape matching and segmentation, the probabilistic formulation

$$P(\Phi|I) = \prod_{\mathbf{x} \in \Omega} \big(H_e(\Phi(\mathbf{x})) P_f(\mathbf{x}) + (1 - H_e(\Phi(\mathbf{x}))) P_b(\mathbf{x}) \big), \tag{4.1}$$

is adopted, which was originally derived in [BR08] and later also used within the PWP3D method [PR12]. It describes the posterior probability of the shape kernel Φ given an image I, with H_e being a smoothed Heaviside step function (for details



(a) (b)

Figure 4.3: An example visualization of a rendered silhouette mask I_s of the 3D squirrel model (a) and its corresponding signed distance transform Φ (b). Here the contour $\mathbf{C}$ between Ω_f and Ω_b in I_s is depicted by an orange outline.

see Section 4.3.2). Here, $P_f(\mathbf{x})$ and $P_b(\mathbf{x})$ represent the per pixel foreground and background region membership probability, based on underlying statistical appearance models (see Section 4.4). In the general context of 2D region-based image segmentation, the closed curve $\mathbf{C}$ would be evolved in an unconstrained manner such that it maximizes $P(\Phi|I)$ and thus the discrepancy between the foreground and background appearance model statistics. In the 6DOF pose estimation scenario, however, the evolution of the object's contour $\mathbf{C}$ is constrained by the known shape prior in form of a 3D model. Therefore the shape kernel only depends on the pose parameters, e.g. $\Phi(\mathbf{x}(\xi))$. It can thus also be seen as a level-set embedding of the pose. Assuming pixel-wise independence and taking the negative log, (4.1) turns into the region-based cost function

$$E_{rb}(\xi) = -\sum_{\mathbf{x}\in\Omega} \log\big(H_e(\Phi(\mathbf{x}(\xi)))P_f(\mathbf{x}) + (1 - H_e(\Phi(\mathbf{x}(\xi))))P_b(\mathbf{x})\big), \tag{4.2}$$

to be minimized with respect to e.g. twist coordinates ξ for pose estimation based on 2D-to-3D shape matching.

4.3.1 Level-Set Pose Embedding

In this work, the level-set embedding $\Phi(\mathbf{x})$ is explicitly computed as

$$\Phi(\mathbf{x}) \doteq \begin{cases} -d(\mathbf{x}, \mathbf{C}) & \forall \mathbf{x} \in \Omega_f \\ d(\mathbf{x}, \mathbf{C}) & \forall \mathbf{x} \in \Omega_b \end{cases},$$

where

$$d(\mathbf{x}, \mathbf{C}) \doteq \min_{\mathbf{c} \in \mathbf{C}} \|\mathbf{c} - \mathbf{x}\|_2,$$

being the 2D Euclidian signed distance function of the binary silhouette mask I_s (see Figure 4.3). In practice $\Phi(\mathbf{x})$ can be computed efficiently using the two-pass algorithm presented in [FH12]. In this work it was implemented with help of CPU multi-threading, where the first pass runs in parallel for each row and the second pass for each column of pixels. In addition to the distance value, also the 2D coordinates of the closest contour point $\mathbf{c} \in \mathbf{C}$ to every pixel $\mathbf{x} \in \Omega_b$ is stored. This is required for obtaining the corresponding 3D surface point as needed to compute the derivates of $\Phi(\mathbf{x})$ (4.18) with respect to a background pixel (see Section 4.5.2).

4.3.2 Heaviside Step Function

The 1D transition between the foreground and the background region is described using the Heaviside step function H_e. In order to be differentiable with regard to gradient-based optimization, typically a smoothed version of H_e is utilized. In this work it is defined explicitly as

$$H_e\left(\Phi(\mathbf{x})\right) \doteq \frac{1}{\pi}\left(-\operatorname{atan}(s \cdot \Phi(\mathbf{x})) + \frac{\pi}{2}\right), \tag{4.3}$$

with s determining the pitch of the smoothed transition. Its derivative is accordingly given by

$$\frac{\partial H_e\left(\Phi(\mathbf{x})\right)}{\partial \Phi(\mathbf{x})} \doteq \delta_e\left(\Phi(\mathbf{x})\right) = \frac{s}{\pi \Phi(\mathbf{x})^2 s^2 + \pi}, \tag{4.4}$$

being the corresponding smoothed Dirac delta function $\delta_e(x)$. In the remainder of this chapter $s = 1.2$ is used (see Figure 4.4), which was determined as most suitable by extensive empirical experiments.

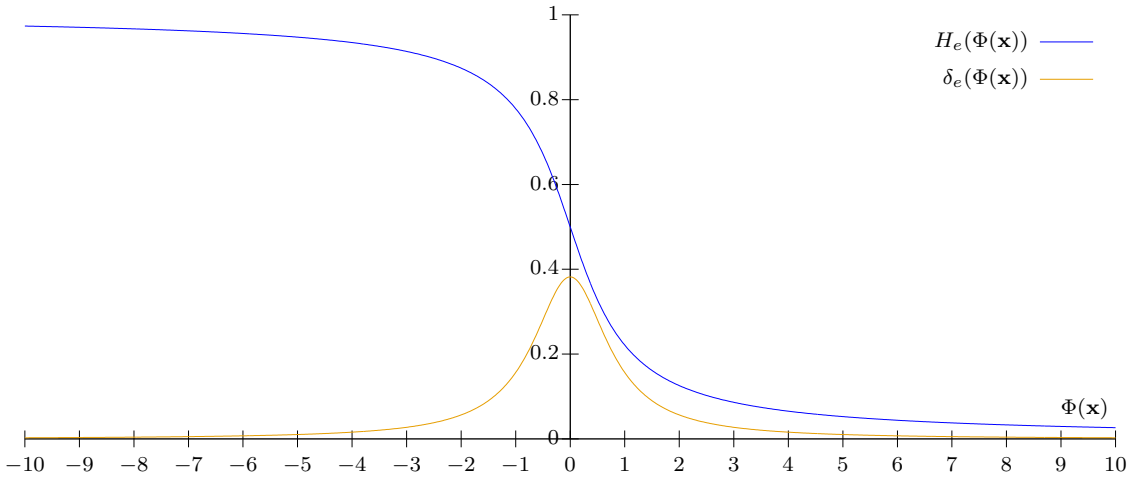


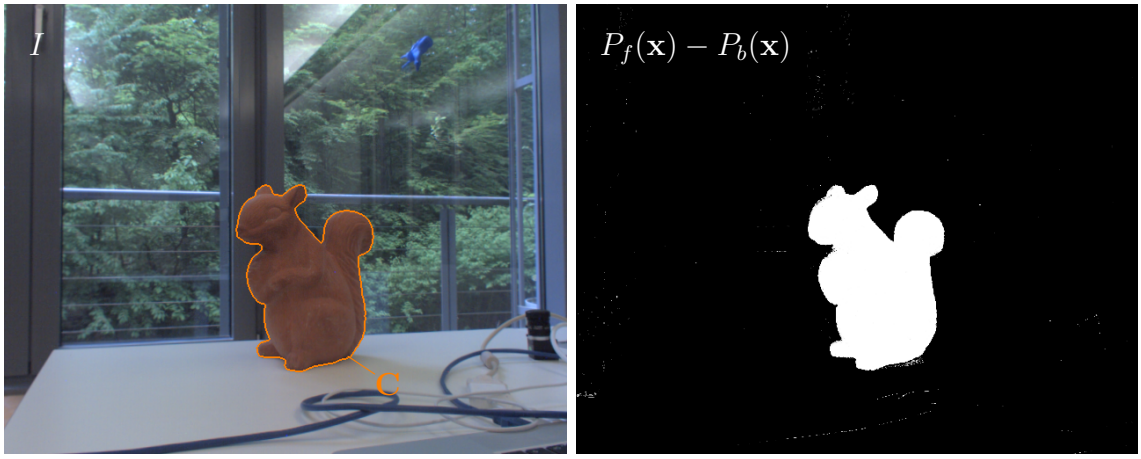
Figure 4.4: A combined plot of the smoothed Heaviside step function H_e and the corresponding smoothed Dirac delta function δ_e for $s = 1.2$ as utilized within this work.

4.4 Statistical Segmentation Models

In the past, different appearance models have been proposed to compute $P_f(\mathbf{x})$ and $P_b(\mathbf{x})$ within cost functions of the form (4.2). The following sections give an overview of the most popular and successful strategies. These are presented in chronological order to illustrate the evolution towards the current state of the art appearance model that was developed within this dissertation (see Section 4.4.3).

4.4.1 A Global Model

Initially [BR08] proposed a global appearance model based on the color distribution in both Ω_f and Ω_b that was adopted in e.g. [PR12] and [TSS16]. Here, each region has its own model denoted by $P(\mathbf{y}|M_f)$ for the foreground i.e. the object and $P(\mathbf{y}|M_b)$ for the background, where $\mathbf{y} = I(\mathbf{x})$. Each of them is represented with a global color histogram. In this work such histograms are computed using the RGB color model with a quantization of 32 bins per channel. Based on this, the region membership probabilities are computed in form of pixel-wise posteriors as



(a) (b)

Figure 4.5: An ideal example for image segmentation using global color histograms. Here a homogeneous object is placed in a scene that does not contain its primary color in the background. The contour $\mathbf{C}$ (depicted in orange) that corresponds to its estimated pose splits the image I into a foreground and a background region (a). This allows to compute pixel-wise posteriors $P_f(\mathbf{x})$ and $P_b(\mathbf{x})$ using (4.5) that provide a segmentation of the object in the image (b).

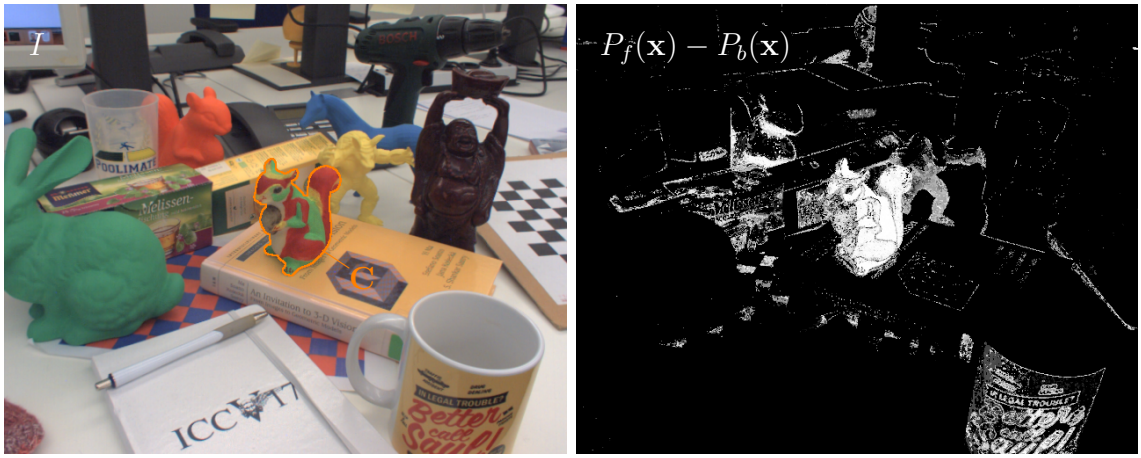
$$
\begin{aligned}
P_f(\mathbf{x}) &= P(M_f|\mathbf{y}) = \frac{P(\mathbf{y}|M_f)}{\eta_f P(\mathbf{y}|M_f) + \eta_b P(\mathbf{y}|M_b)}, \\
P_b(\mathbf{x}) &= P(M_b|\mathbf{y}) = \frac{P(\mathbf{y}|M_b)}{\eta_f P(\mathbf{y}|M_f) + \eta_b P(\mathbf{y}|M_b)},
\end{aligned}
\tag{4.5}
$$

with

$$
\eta_f = \sum_{\mathbf{x}\in\Omega} H_e(\Phi(\mathbf{x})), \quad \eta_b = \sum_{\mathbf{x}\in\Omega} 1 - H_e(\Phi(\mathbf{x})).
$$

Using this global model, a segmentation of a previously unseen image can now be obtained by simply computing the pixel-wise difference $P_f(\mathbf{x}) - P_b(\mathbf{x})$ (see Figure 4.5)

This model is also used within PWP3D [PR12] where it is further suggested to keep the appearance models temporally consistent, in order to adapt to scene changes while tracking the object. Having successfully estimated the current live pose $T(t_l)$ allows to render a corresponding silhouette mask denoted by $I_s(t_l)$. Ideally, this mask provides an exact segmentation of the object region the current camera frame $I(t_l)$ and can thus be used in order to compute up-to-date color histograms $P^{t_l}(\mathbf{y}|M_f)$ and



(a) (b)

Figure 4.6: A critical example case for image segmentation based on global color histograms, where a heterogenous object is placed in a cluttered scene that contains all colors of the objects itself in the background. Here the contour $\mathbf{C}$ of the object was assumed to be known (a). Even when computing the global color histograms from this perfect image splitting, the corresponding pixel-wise posteriors do not provide a sufficient object segmentation for pose estimation (b).

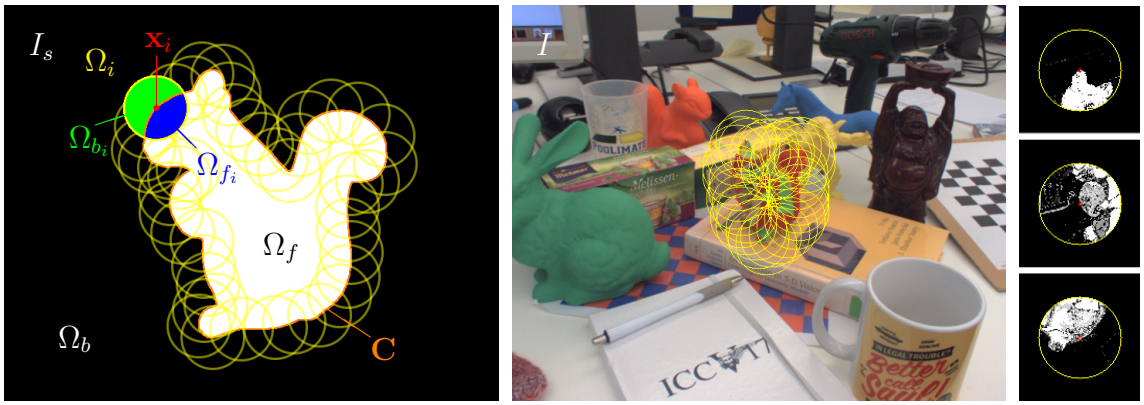
$P^{t_l}(\mathbf{y}|M_b)$. Instead of always using the latest color distribution for the appearance models, [PR12] suggested to recursively adjust the histograms by

$$\begin{aligned} P(\mathbf{y}|M_f) &= (1-\alpha_f)P^{t_{l-1}}(\mathbf{y}|M_f) + \alpha_f P^{t_l}(\mathbf{y}|M_f), \\ P(\mathbf{y}|M_b) &= (1-\alpha_b)P^{t_{l-1}}(\mathbf{y}|M_b) + \alpha_b P^{t_l}(\mathbf{y}|M_b), \end{aligned} \tag{4.6}$$

to prevent them from being corrupted by occlusions or pose estimation inaccuracies. Here, α_f and α_b are the foreground and background learning rates. For example, in [PR12, TSS16] learning rates of $\alpha_f = 0.05$ and $\alpha_b = 0.02$ are suggested for this update.

4.4.2 A Localized Model

The appearance model of the previous Section 4.4 is based on the assumption that the global color distribution is sufficiently descriptive in order to distinguish between



(a) (b) (c)

Figure 4.7: An illustration of the localized segmentation model of [HH16] where multiple local regions Ω_i are defined along the contour $\mathbf{C}$ (a). Applying this model to the cluttered scene of Figure 4.6 allows to compute local pixel-wise posteriors $P_{f_i}(\mathbf{x})$ and $P_{b_i}(\mathbf{x})$ using (4.9) based on the individual local color histograms corresponding to Ω_{f_i} and Ω_{b_i} (b). Here the local segmentation result $P_{f_i}(\mathbf{x}) - P_{b_i}(\mathbf{x})$ is visualized for three different local regions (at the top, left and bottom) in order to illustrate the improvement over the global result, which is sufficient for pose estimation (c).

the foreground and the background region. Therefore, it has been shown to work particularly well with homogeneous objects of a distinct color that is not dominantly present in the rest of the scene. However for homogeneous objects and in case of cluttered scenes this global model is prone to fail (see Figure 4.6).

Hence, in [HH16] a localized appearance model was proposed for the PWP3D approach that better captures spatial variations of the object's color distribution. The idea is to build a segmentation model from multiple overlapping circular image regions along the object's contour as originally introduced in [LT08]. Each such local region is denoted by $\Omega_i = \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{x}_i\|_2 < r\}$ with radius r, centered at pixel $\mathbf{x}_i \in \mathbf{C}$. Now, I_s splits each Ω_i into a foreground region $\Omega_{f_i} \subset \Omega_i$ and a background region $\Omega_{b_i} = \Omega_i \setminus \Omega_{f_i}$ (see Figure 4.7). This allows to compute local foreground and background color histograms for each region. Given these, in [HH16] the shape matching cost is computed separately for each Ω_i by

$$E_i = \sum_{\mathbf{x}\in\Omega} \log\big(H_e(\Phi(\mathbf{x}))P_{f_i}(\mathbf{x}) + (1 - H_e(\Phi(\mathbf{x})))P_{b_i}(\mathbf{x})\big)\mathbf{B}_i(\mathbf{x}), \tag{4.7}$$

with a corresponding masking function

$$\mathbf{B}_i(\mathbf{x}) = \begin{cases} 1 & \forall \mathbf{x} \in \Omega_i \\ 0 & \forall \mathbf{x} \notin \Omega_i \end{cases},$$

indicating whether a pixel $\mathbf{x}$ lies within a local region or not. These individual terms are combined in a common localized energy function

$$E_{rb} = -\frac{1}{n} \sum_{i=1}^{n} E_i, \tag{4.8}$$

used for pose estimation that measures the average cost over all local regions. Here, the local region membership probabilities $P_{f_i}(\mathbf{x})$ and $P_{b_i}(\mathbf{x})$ are accordingly computed individually from the respective local histograms as

$$\begin{aligned} P_{f_i}(\mathbf{x}) &= \frac{P(\mathbf{y}|M_{f_i})}{\eta_{f_i} P(\mathbf{y}|M_{f_i}) + \eta_{b_i} P(\mathbf{y}|M_{b_i})}, \\ P_{b_i}(\mathbf{x}) &= \frac{P(\mathbf{y}|M_{b_i})}{\eta_{f_i} P(\mathbf{y}|M_{f_i}) + \eta_{b_i} P(\mathbf{y}|M_{b_i})}, \end{aligned} \tag{4.9}$$

with

$$\eta_{f_i} = \sum_{\mathbf{x} \in \Omega_i} H_e(\Phi(\mathbf{x})), \quad \eta_{b_i} = \sum_{\mathbf{x} \in \Omega_i} 1 - H_e(\Phi(\mathbf{x})),$$

in analogy to the global model. In [HH16], however, temporal consistency of the local appearance models was not addressed. The local region centers $\mathbf{x}_i$ were calculated as arbitrary sets of pixel locations along $\mathbf{C}$ for each image. Thus, this approach in general does not allow to establish correspondences of the centers across multiple frames (i.e. $\mathbf{x}_i(t_l) \leftrightarrow \mathbf{x}_i(t_{l-1})$) which is required in order to update the respective histograms during tracking.

4.4.3 Temporally Consistent Local Color Histograms

The appearance model used within this work has first been proposed in [TSS17]. It is based on temporally consistent local color histograms (tclc-histograms) and

(a) (b) (c)

Figure 4.8: Object segmentation using tclc-histograms in the scene of Figure 4.6. Here, a tclc-histogram attached to a 3D vertex $\mathbf{X}_i$ of the squirrel model is shown in a schematic 3D visualization (a). Due to this relationship each center $\mathbf{x}_i$ of a local region Ω_i can be identified in every frame. In this example this is illustrated by depicting the local regions as colored circles where the RGB value matches the relative object coordinates of the corresponding $\mathbf{X}_i$ (b). Based on such tclc-histograms, the average pixel-wise posteriors $\bar{P}_f(\mathbf{x})$ and $\bar{P}_b(\mathbf{x})$ can be computed using (4.9), which provide an even higher quality object segmentation (see e.g. the bottom part of the squirrel) than the individual local posteriors in Figure 4.7 (c).

thereby addresses the mentioned issues related to the localized model of [HH16]. Here, each 3D model vertex $\mathbf{X}_i$ is associated with a local foreground and background histograms (see Figure 4.8). In contrast to [HH16] this allows to compute the 2D histogram centers from projecting all model vertices into the image plane as $\mathbf{x}_i = \pi\big(K(T_{cm}\tilde{\mathbf{X}}_i)_{3\times 1}\big)$ and selecting a subset of all $\mathbf{x}_i \in \mathbf{C}$. In practice also those with $d(\mathbf{x}_i, \mathbf{C}) \leq \lambda r$ are considered. For this, in the following $\lambda = 0.1$ is used, in order to ensure that the contour is evenly sampled. For runtime reasons, since this can lead to a large number of histograms that have to be updated, a maximum of 100 centers per frame are randomly picked. This Monte Carlo approach requires the mesh vertices $\mathbf{X}_i$ to be uniformly distributed across the 3D model in order to evenly cover all regions. They should also be limited in number to ensure that all histograms will get updated regularly. Therefore potentially two different 3D mesh representations of the 3D model are required. The original mesh is used to render exact silhouette views regardless of the mesh structure while a reduced (e.g. a maximum of 5000 vertices) and evenly sampled version of the model is used for computing the 2D centers of the histograms.

Since each histogram is anchored to the object's surface, center correspondences $\mathbf{x}_i(t_l) \leftrightarrow \mathbf{x}_i(t_{l-1})$ between frames are simply given by the projection of corresponding surface points, i.e. $\pi\big(K(T_{cm}(t_l)\tilde{\mathbf{X}}_i)_{3\times1}\big) \leftrightarrow \pi\big(K(T_{cm}(t_{l-1})\tilde{\mathbf{X}}_i)_{3\times1}\big)$. This enables to keep the individual histograms temporally consistent. Whenever a model vertex projects onto the contour for the first time, its histograms are initialized from the local region around its center in the current frame. Otherwise, if its histograms already contain information from any previous frame, they are updated as

$$P(\mathbf{y}|M_{f_i}) = (1-\alpha_f)P^{t_{l-1}}(\mathbf{y}|M_{f_i}) + \alpha_f P^{t_l}(\mathbf{y}|M_{f_i}),$$

$$P(\mathbf{y}|M_{b_i}) = (1-\alpha_b)P^{t_{l-1}}(\mathbf{y}|M_{b_i}) + \alpha_b P^{t_l}(\mathbf{y}|M_{b_i}),$$

in analogy to (4.6). In practice, the key idea to efficiently build and update the localized appearance model is to process each histogram region Ω_i^j in parallel on the CPU using the Bresenham algorithm [Bre77] to scan the pixels of the corresponding circles. Since the localized model captures spatial variations a lot more precisely than the global model, experiments have shown that higher learning rates of $\alpha_f = 0.1$ and $\alpha_b = 0.2$ can be used, enabling much faster adaptation to dynamic changes. Based on the results presented in [HH16], in the following a histogram region radius of $r = 40\,\text{px}$ is used regardless of the object's distance to the camera.

A fixed radius is used because for continuous tracking one can only compute the segmentation within the histogram regions belonging to the silhouette in the previous frame. Thus, in cases of fast translational movement of the object or rotation of the camera it is possible that the object in the current frame projects entirely outside the previous histogram regions. This becomes more likely for smaller histogram radii. Therefore, if the radius shrinks with distance to the camera, the object is more likely to get lost at greater distances. Non-overlapping histogram regions in case of close distances and sparse surface sampling hardly influence the reliability of the proposed approach.

The other extreme case is when the object is so far away that all histograms overlap. Here the discriminability of the appearance model is reduced since all pixels then lie within all histograms. The segmentation model then acts like the global approach constrained to a local region around the silhouette extended by the histogram radius. However, this is still better than the original global model and to be preferred over

the increased risk of losing the object easily with smaller radii.

In [TSS17] it has furthermore been shown that computing the average energy (4.8) over all local regions Ω_i potentially suffers from the same segmentation problems locally as the previous approach based on the global appearance model. This is because each individual local energy E_i (4.7) again is only influenced by the posteriors P_{f_i} and P_{b_i} computed from a single local foreground and background histogram. More robust result can be obtained by computing the average posteriors from all local histograms instead as

$$\bar{P}_f(\mathbf{x}) = \frac{1}{\sum_{i=1}^{n} \mathbf{B}_i(\mathbf{x})} \sum_{i=1}^{n} P_{f_i}(\mathbf{x}) \mathbf{B}_i(\mathbf{x}),$$

$$\bar{P}_b(\mathbf{x}) = \frac{1}{\sum_{i=1}^{n} \mathbf{B}_i(\mathbf{x})} \sum_{i=1}^{n} P_{b_i}(\mathbf{x}) \mathbf{B}_i(\mathbf{x}),$$

and use these within (4.2). This thereby leads to a slightly different energy formulation, changing (4.2) into

$$E_{rb}(\xi) = -\sum_{\mathbf{x}\in\Omega} \log\big(H_e(\Phi(\mathbf{x}(\xi)))\bar{P}_f(\mathbf{x}) + (1 - H_e(\Phi(\mathbf{x}(\xi))))\bar{P}_b(\mathbf{x})\big), \tag{4.10}$$

which will be used in the following for the proposed region-based pose tracking and detection approach. Although this may seem like a minor change, experiments have shown that it leads to a significant increase in robustness and accuracy for both pose tracking and detection in cluttered scenes.

In [TSS17] the performance of the different appearance models was compared experimentally. In the first experiment the approaches were evaluated for pose tracking without loss detection in a semi-synthetic image sequence that allows to compare with ground truth pose information (see Figure 4.9). More details on how such sequences are created are given in Section 4.8.1. In this case a textured driller model provided in the public single instance pose detection data set of [HLI+12] was used. Here, the proposed approach using tclc-histograms within (4.10) was compared to that of [TSS16] using global histograms, localized histogram segmentation without temporal consistency (i.e. [HH16] with the pose optimization strategy of [TSS16])

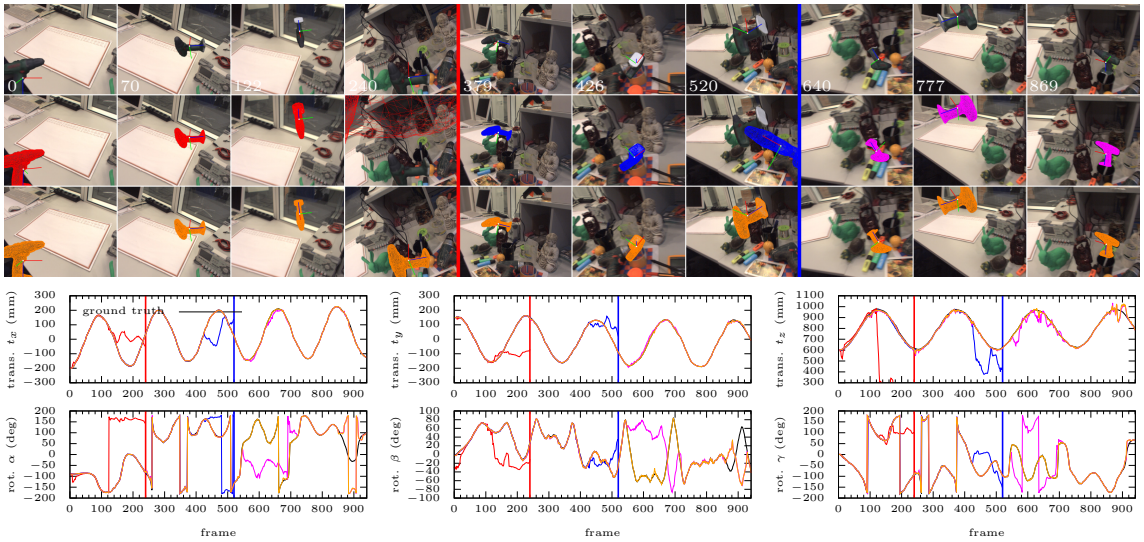


Figure 4.9: Top: Results of a semi-synthetic ground truth pose tracking experiment (1st row: virtually augmented input images with coordinates axes. 2nd row: Result of [TSS16] (red), tracking without temporal consistency (blue) and tclc-histograms within (4.8) (magenta). 3rd row: The proposed tclc-histograms within (4.10) (orange).). Bottom: Determined pose parameters. Plot colors correspond to mesh colors drawn in the example frames with respect to the algorithm used. The red and blue vertical lines between the frames and in the plots mark the tracking losses of the respective approaches.

as well as tclc-histograms used within (4.8) instead of (4.10).

The results show that the global segmentation leads to a tracking loss as soon as the object moves close to a background region that is similarly dark as the tip of the driller (e.g. frame 122), even though the tip itself is not near it. Next, localized segmentation without temporal consistency gets stuck in another driller of similar color present in the background while moving across it (e.g. frame 426) eventually also leading to a tracking loss. The proposed method (4.10) as well as the one using (4.8) are able to successfully track the object in the whole sequence. Here (4.8) suffers twice from silhouette pose ambiguities for rotation (e.g. frame 640 and 869) while this only happens once (e.g. frame 869) for the proposed method. Before this ambiguity occurred the difference to ground truth of the proposed approach for translation (t_x, t_y, t_z) is $(1.2 \pm 0.9\,\text{mm},\ 1.3 \pm 1.1\,\text{mm},\ 7.5 \pm 5.7\,\text{mm})$ and $(2.3 \pm 2.3^\circ, 1.3 \pm 1.2^\circ, 1.1 \pm 2.0^\circ)$ measured for the Euler angles (α, β, γ) describing the

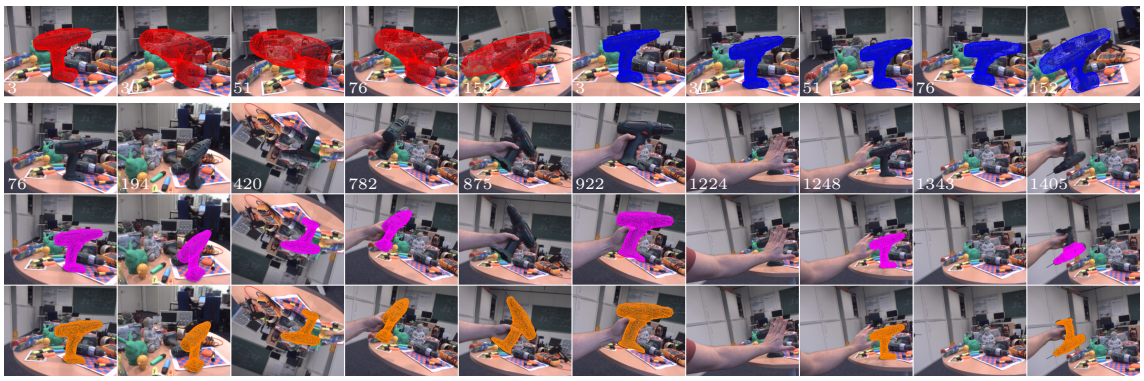


Figure 4.10: Results in the real image sequence (colors match Figure 4.9). 1st row: Tracking failure of [TSS16] (left) and local segmentation without temporal consistency (right) in the beginning. 2nd row: RGB input frames. 3rd row: The proposed tclc-histograms within (4.8). 4th row: The proposed tclc-histograms within (4.10).

rotation around the (X_c, Y_c, Z_c) axes of the camera.

The second experiment demonstrates the performance for both tracking and detection in a real image sequence (see Figure 4.10). Here, the same 3D model as in the first experiment was used to estimate the pose of an identically constructed real driller held in hand. The sequence contains a cluttered background including another driller similar in shape and color in the background, complex motion of both the camera and the object, partial and total occlusion. Furthermore, the driller was equipped with a drill bit that is not part of the 3D model.

For evaluation tracking loss detection and pose recovery (see Section 4.6.2 for details) were enabled for the method using tclc-histograms within both energy functions. Note that pose detection is not possible without temporal consistency and was not included by [TSS16]. In this scene the global model does not provide a sufficient segmentation to initially estimate the pose due to the large amount of background clutter and gets lost immediately. The localized segmentation model without temporal consistency again struggles with the background clutter and also gets lost at the beginning of the sequence. While rotating the driller 360° in hand the approach using tclc-histograms within (4.8) gets lost at one point (e.g. frame 782) and is not able to recover until the rotation is complete and the side of the driller that was previously tracked becomes visible again (e.g. frame 922). At the end of the sequence

the driller is totally occluded leading to a tracking loss from which the proposed method successfully recovers regardless of the energy used. A second loss occurs caused by moving the driller outside the field of view of the camera and moving it back in in an upside down pose. This time, pose detection using (4.10) successfully recovers while (4.8) leads to a misalignment which is characteristic for the results of the comprehensive pose detection experiment in Section 4.8. These two exemplary experiments overall demonstrate the effectiveness of the proposed appearance model using tclc-histograms. It is significantly more robust towards background clutter, partial occlusions and illumination changes than comparable previous solutions.

4.5 Efficient Pose Optimization

Traditionally cost functions of the form (4.10) have been optimized using gradient descent – see for example [PR12] or [HH16]. In comparison to second-order (Newton) methods, this has several drawbacks: Firstly, one has to determine suitable time step sizes associated with translation and rotation. Too small step sizes lead to often very slow convergence, too large step sizes easily induce oscillations and instabilities (as explained in Section 2.3.1). Secondly, convergence is typically not as robust and rather slow making the technique less suitable for accurate and robust tracking in real-time. In particular, employing line-search strategies to determine the optimal step size are inefficient in this context. This is because evaluating (4.10) and computing its gradient both requires rendering I_s and computing Φ which are quite costly operations. Therefore, a pure function evaluation is computationally demanding similar to calculating its gradient and should thus be avoided.

Applying a second-order optimization scheme such as Gauß-Newton, on the other hand, is not straightforward because the cost function (4.10) is not in the classical form of a nonlinear least-squares problem. Calculating the Hessian matrix in order to determine the step size within a quasi-Newton approach can be done with an acceptable increase of runtime. However, experiments proved it to be numerically unstable, as commonly reported in literature. In [TSSC18], a novel strategy was developed to circumvent these issues which is based on rewriting the original problem in form of a re-weighted nonlinear least-squares estimation as shown in the following.

4.5.1 Derivation of a Gauß-Newton Strategy

The cost function in (4.10) can be written compactly as

$$E_{rb}(\xi) = \sum_{\mathbf{x}\in\Omega} r_{rb}(\mathbf{x}, \xi),$$

where

$$r_{rb}(\mathbf{x}, \xi) = -\log\big(H_e(\Phi(\mathbf{x}(\xi)))\bar{P}_f(\mathbf{x}) + (1 - H_e(\Phi(\mathbf{x}(\xi))))\bar{P}_b(\mathbf{x})\big).$$

Unfortunately, this is not in the traditional form of a nonlinear least-squares estimation problem for which the Gauß-Newton algorithms is applicable. However, this expression can simply be rewritten as a *nonlinear weighted least-squares problem* of the form

$$E_{rb}(\xi) = \frac{1}{2}\sum_{\mathbf{x}\in\Omega} \psi(\mathbf{x}) r_{rb}(\mathbf{x}, \xi)^2, \text{ with } \psi(\mathbf{x}) = \frac{1}{r_{rb}(\mathbf{x}, \xi)}. \tag{4.11}$$

To optimize this cost function, one can apply the technique of iteratively re-weighted least-squares estimation which amounts to solving the above problem for fixed weights $\psi(\mathbf{x})$ by means of Gauß-Newton optimization and alternatingly updating the weights $\psi(\mathbf{x})$. Over the iterations, these weights will adaptively re-weight respective terms.

In the fixed-weight assumption, the energy gradient of (4.11) is given by

$$\begin{aligned}\nabla E_{rb}(\xi) =& \frac{1}{2}\sum_{\mathbf{x}\in\Omega}\left(\psi(\mathbf{x})\frac{\partial r_{rb}(\mathbf{x}, \xi)^2}{\partial \xi}\right)^\top = \sum_{\mathbf{x}\in\Omega}\left(\psi(\mathbf{x}) r_{rb}(\mathbf{x}, \xi)\frac{\partial r_{rb}(\mathbf{x}, \xi)}{\partial \xi}\right)^\top \\ =& \sum_{\mathbf{x}\in\Omega}\left(\frac{\partial r_{rb}(\mathbf{x}, \xi)}{\partial \xi}\right)^\top,\end{aligned} \tag{4.12}$$

since $\psi(\mathbf{x}) r_{rb}(\mathbf{x}, \xi) = 1$. Accordingly the Hessian is given by

$$\nabla^2 E_{rb}(\xi) = \sum_{\mathbf{x}\in\Omega} \psi(\mathbf{x})\left(\left(\frac{\partial r_{rb}(\mathbf{x}, \xi)}{\partial \xi}\right)^\top \frac{\partial r_{rb}(\mathbf{x}, \xi)}{\partial \xi} + r_{rb}(\mathbf{x}, \xi)\frac{\partial^2 r_{rb}(\mathbf{x}, \xi)}{\partial \xi^2}\right).$$

As described in Section 2.3.3, the Gauß-Newton algorithm then emerges based on

the assumption that r_{rb} is linear in ξ (i.e. ${}^{\partial^2 r_{rb}}/_{\partial \xi^2} \approx 0$), giving

$$\nabla^2 E_{rb}(\xi) \approx \sum_{\mathbf{x}\in\Omega} \psi(\mathbf{x}) \left(\frac{\partial r_{rb}(\mathbf{x},\xi)}{\partial \xi} \right)^\top \frac{\partial r_{rb}(\mathbf{x},\xi)}{\partial \xi}, \tag{4.13}$$

by dropping the second-order derivative of the residual r_{rb}. Denote the 1×6 Jacobian of r_{rb} at the current pose i.e. $\xi_0 = \mathbf{0}$ by

$$J_{rb}(\mathbf{x},\xi_0) = \left. \frac{\partial r_{rb}(\mathbf{x},\xi)}{\partial \xi} \right|_{\xi_0} \in \mathbb{R}^{1\times 6}. \tag{4.14}$$

Under the above assumptions the second-order Taylor expansion of the cost function E_{rb} is given by

$$E_{rb}(\xi_0 + \Delta\xi) \approx E_{rb}(\xi_0) + \sum_{\mathbf{x}\in\Omega} J_{rb}(\mathbf{x},\xi_0)\Delta\xi + \frac{1}{2}\sum_{\mathbf{x}\in\Omega} \psi(\mathbf{x})\Delta\xi^\top J_{rb}(\mathbf{x},\xi_0)^\top J_{rb}(\mathbf{x},\xi_0)\Delta\xi,$$

which leads to the optimal Gauß-Newton update step of

$$\Delta\xi_t = -\left(\sum_{\mathbf{x}\in\Omega} \psi(\mathbf{x}) J_{rb}(\mathbf{x},\xi_0)^\top J_{rb}(\mathbf{x},\xi_0) \right)^{-1} \sum_{\mathbf{x}\in\Omega} J_{rb}(\mathbf{x},\xi_0)^\top. \tag{4.15}$$

Note that the only difference to the Gauß-Newton-like step of [TSS16] are the weighting terms $\psi(\mathbf{x})$ in the approximated Hessian matrix. Including them, however, leads to an additional improvement of the convergence properties i.e. the tracking robustness (see Section 4.8).

In [TSS16] the advantages of the Gauss-Newton-like second-order strategy in comparison to the first-order gradient descent method of the original PWP3D [PR12] were demonstrated. Essentially, the true Gauss-Newton optimization proposed here has similar convergence properties as the previous Gauss-Newton-like one. Therefore, here two exemplary experiments from [TSS16] are included in order to illustrate the general difference between first-order and second-order optimization in this context.

The two corresponding real image sequences were pre-recorded at 50 Hz. Here, both methods were using the global appearance model for image segmentation. The

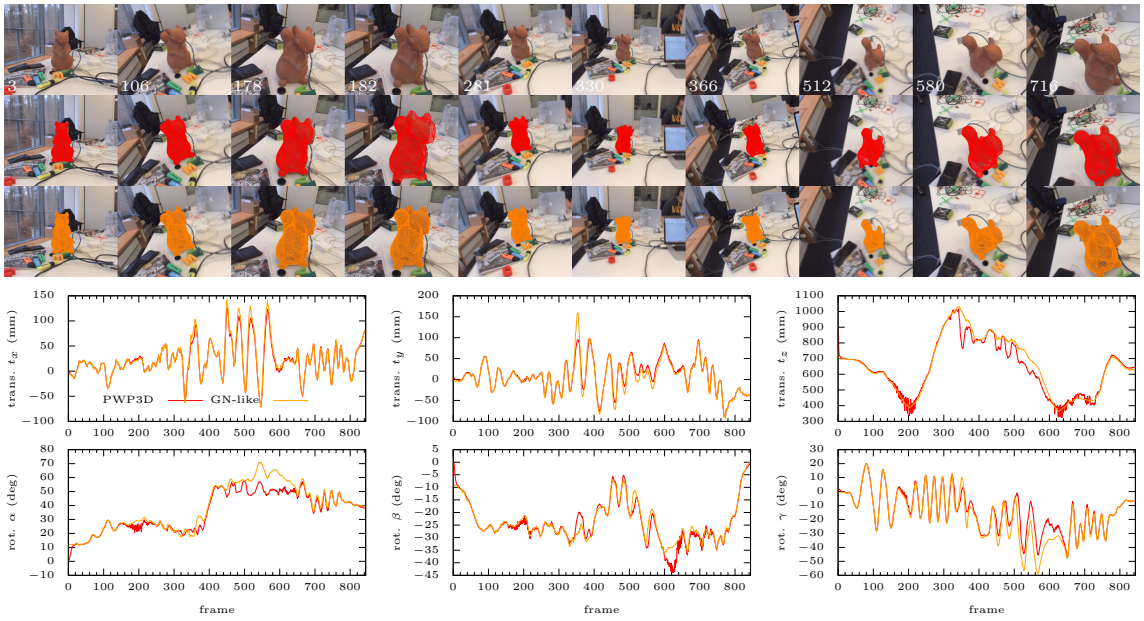


Figure 4.11: Top: Result of the squirrel pose tracking experiment (1st row: input images, 2nd row: Result of PWP3D [PR12] (red), 3rd row: Result of [TSS16] (orange)). Bottom: Determined pose parameters. Plot colors correspond to mesh colors drawn in the example frames with respect to the algorithm used.

implementation of [TSS16] was able to track the single object in real-time in both sequences while the original implementation of PWP3D (with GPU support) needed about 60 – 80 ms per frame on the same laptop. In order to neglect the influence of the runtime on the frame rate and the resulting pose difference between consecutive frames, all sequences were post-processed at full frame rate by both algorithms. For PWP3D eight optimization iterations were used since this would enable at least 20 – 25 Hz when using more powerful hardware according to [PR12]. While the implementation of [TSS16] was used with the same settings and number of iterations throughout, all four step sizes of PWP3D had to be adjusted individually for both sequences in order to achieve the best results.

In the first experiment the camera was moved around in a scene for tracking a stationary squirrel figurine (Figure 4.11). Here, it was endeavored to generate fast rotational motion at different distances between camera and object. The results show how much the step sizes in PWP3D depend on the distance to the camera. On

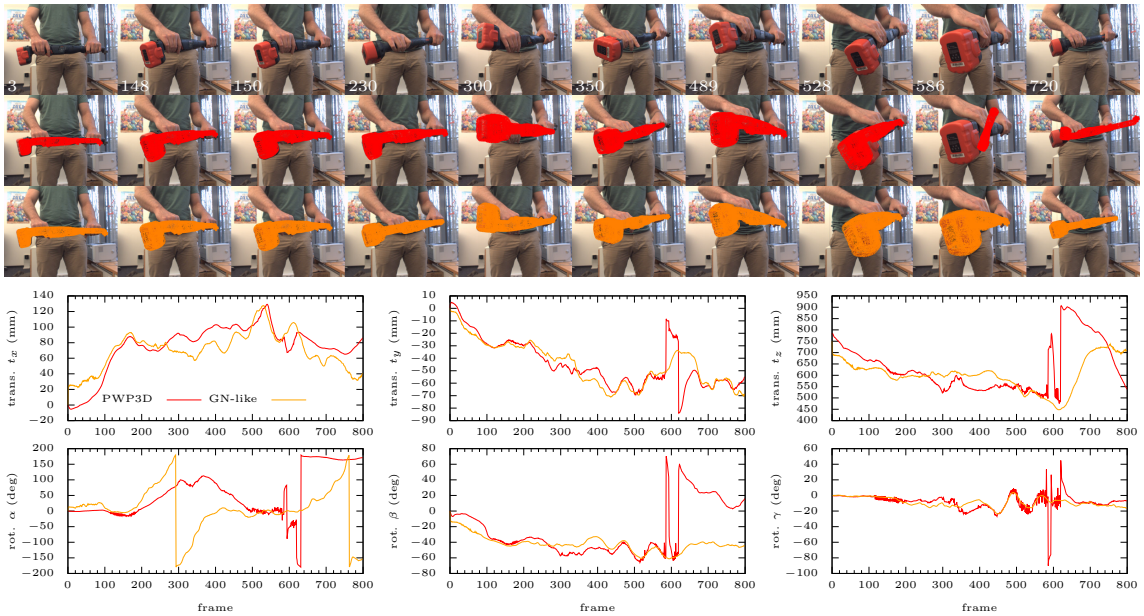


Figure 4.12: Results of the screwdriver pose tracking experiment. The visualization of the tracking results is analogous to Figure 4.11.

one hand, if the distance becomes too small, the step sizes are too large and the pose starts to oscillate (e.g. frames 150–230). On the other hand, if the distance between the object and the camera increases the overall optimization quality degrades, resulting in the step sizes to be too small to reach the minimum (e.g. starting around frame 280).

In contrast, the method presented in [TSS16] performed visually pleasing throughout the whole sequence. If the step sizes of PWP3D suit the distance, both methods perform equally well, which shows that the Gauß-Newton-like approach does not degrade the tracking quality (e.g. frames 20–150). At initialization the Gauß-Newton-like method always converged within the first three frames (18 iterations) while PWP3D needed at least about 12 frames (96 iterations) until alignment.

In the second experiment a hand held screwdriver was tracked while it moved in front of the stationary camera (Figure 4.12). The sequence contains a challenging full 360° turn around the X_m-axis of the screwdriver that shows the advantage of using twist coordinates for pose parametrization instead of a translation vector and a decoupled unit quaternion as used in PWP3D (frames 180–400). Here, the rotation

step size for PWP3D was set to a large value such that it was close to oscillating (e.g. frames 105–150) since this produced the best overall results.

While the method of [TSS16] was able to correctly track this motion, PWP3D fails to determine the rotation of the object despite the large step size for rotation. Starting at around frame 450, the screwdriver was moved closer towards the camera, leading to a tracking loss of PWP3D at frame 586. Here the method of [TSS16] remained stable even though both methods suffer from the global segmentation model, due to the heterogenous appearance of the screwdriver and when it gets occluded by the hands.

4.5.2 Computation of the Derivatives

The per pixel Jacobian term of (4.14) is computed by applying the chain-rule as

$$J_{rb}(\mathbf{x}, \xi_0) = -\frac{\bar{P}_f(\mathbf{x}) + \bar{P}_b(\mathbf{x})}{H_e(\Phi(\mathbf{x}))(\bar{P}_f(\mathbf{x}) - \bar{P}_b(\mathbf{x})) + \bar{P}_b(\mathbf{x})} \delta_e\left(\Phi(\mathbf{x})\right) \frac{\partial \Phi(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial \mathbf{x}(\xi)}{\partial \xi}\bigg|_{\xi_0}, \quad (4.16)$$

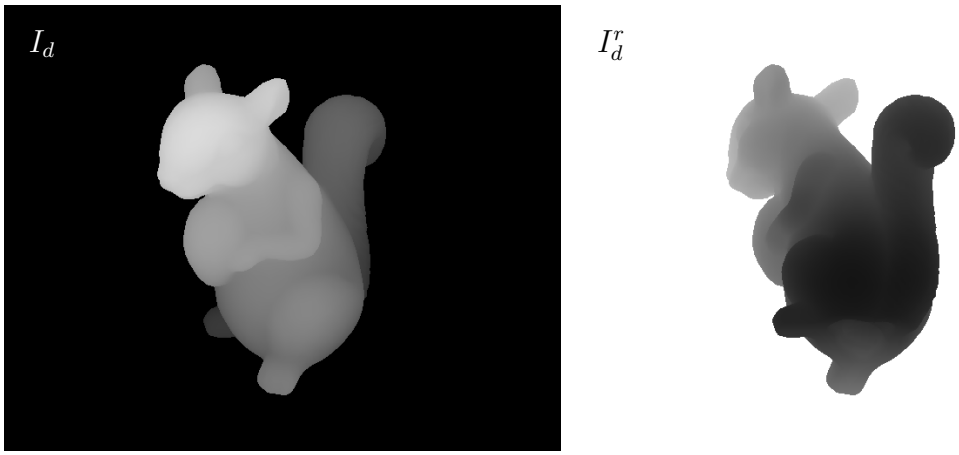
where δ_e is the smoothed Dirac delta function corresponding to H_e (see Section 4.3.2). The derivatives of $\Phi(\mathbf{x})$ with respect to a pixel $\mathbf{x}$ are computed as the transposed 2D image gradient

$$\frac{\partial \Phi(\mathbf{x})}{\partial \mathbf{x}} = \left[\tfrac{\partial \Phi(\mathbf{x})}{\partial x}, \tfrac{\partial \Phi(\mathbf{x})}{\partial y}\right] = \begin{bmatrix} \nabla_x \Phi \\ \nabla_y \Phi \end{bmatrix}^\top = \begin{bmatrix} \frac{\Phi(x+1,y) - \Phi(x-1,y)}{2} \\ \frac{\Phi(x,y+1) - \Phi(x,y-1)}{2} \end{bmatrix}^\top \in \mathbb{R}^{1\times 2}, \quad (4.17)$$

simply calculated from central differences. By again performing piecewise linearization of the matrix exponential $\exp(\hat{\xi})$ in each iteration, according to (2.40) the derivatives of the pixel coordinates $\mathbf{x}$ at ξ_0 are computed as

$$\frac{\partial \mathbf{x}(\xi)}{\partial \xi}\bigg|_{\xi_0} = \begin{bmatrix} \frac{f_x}{Z_c} & 0 & -\frac{X_c f_x}{Z_c^2} \\ 0 & \frac{f_y}{Z_c} & -\frac{Y_c f_y}{Z_c^2} \end{bmatrix} \begin{bmatrix} 0 & Z_c & -Y_c & 1 & 0 & 0 \\ -Z_c & 0 & X_c & 0 & 1 & 0 \\ Y_c & -X_c & 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 6}, \quad (4.18)$$

with $\mathbf{X}_c = [X_c, Y_c, Z_c]^\top = (T_{cm}^t \tilde{\mathbf{X}}_m)_{3\times 1}$. In case of this region-based approach the respective 3D surface points per pixel can be efficiently recovered with help of a depth



(a) (b)

Figure 4.13: The two depth map types used within the proposed approach, where brighter pixels are closer to the camera. The regular depth map I_d corresponding to the closest surface points (a). The reverse depth map I_d^r corresponding to the most distant surface points (b).

map I_d corresponding to I_s that are both rendered in each iteration. Given I_d, for all pixels $\mathbf{x} \in \Omega_f$ the coordinates are computed via back-projection as $\mathbf{X}_c(I_d, \mathbf{x})$ according to (2.33). For all pixels $\mathbf{x} \in \Omega_b$, $\mathbf{X}_c$ is computed as $\mathbf{X}_c(I_d, \mathbf{c}) = D(I_d, \mathbf{c})K^{-1}\tilde{\mathbf{c}}$ where $\mathbf{c} = \arg\min_{\mathbf{c}\in\mathbf{C}} \|\mathbf{c} - \mathbf{x}\|_2$ is the closest contour pixel to $\mathbf{x}$ (see Section 4.3.1).

In [PR12] it has been shown that it is beneficial to not only consider the points on the surface closest to the camera but also the most distant ones (at the rear of the object) for pose optimization. In order to obtain the respective coordinates for each pixel, an additional so-called *reverse* depth map denoted by I_d^r is computed (see Figure 4.13). It is obtained by inverting the OpenGL depth check used to compute the corresponding Z-buffer. Given I_d^r, the most distant surface point $\mathbf{X}_c(I_d^r, \mathbf{x})$ corresponding to a pixel $\mathbf{x}$ is then also recovered via back-projection as $\mathbf{X}_c(I_d^r, \mathbf{x}) = D(\mathbf{x}, I_d^r)K^{-1}\tilde{\mathbf{x}}, \forall \mathbf{x} \in \Omega_f$ and $\mathbf{X}_c(I_d^r, \mathbf{x}) = D(\mathbf{c}, I_d^r)K^{-1}\tilde{\mathbf{c}}, \forall \mathbf{x} \in \Omega_b$.

In the proposed implementation the approximated Hessian (4.13) and the gradient (4.12) of the energy needed for the parameter step (4.15) are calculated in parallel for each row of pixels on the CPU. Here, each thread calculates its own sums over $\psi(\mathbf{x})(J_{rb}(\mathbf{x}, \xi_0))^\top J_{rb}(\mathbf{x}, \xi_0)$ and $J_{rb}(\mathbf{x}, \xi_0)$ which are finally added up in the main thread. Following PWP3D, for each pixel both Jacobian terms with respect to the

coordinates of $\mathbf{X}_c(I_d, \mathbf{x})$ as well $\mathbf{X}_c(I_d^r, \mathbf{x})$ are added.

In order to increase the convergence speed, the iterative pose optimization is computed in a hierarchical coarse to fine approach. This also makes the tracking more robust towards fast movement or motion blur. Details on this multi-level strategy are given in Section 4.6.1.

4.5.3 Initialization

During successful tracking, for every new frame $I(t_l)$ the optimization starts at the previously estimated pose, i.e. $T_{cm}^0(t_l) = T_{cm}(t_{l-1})$. In the current system the whole tracking process for a previously unseen object is initialized manually. For this, the 3D model is rendered as an overlay to the live camera feed using a pre-defined initial pose $T_{cm}^0(t_0)$. The user is then required to roughly align this mask with the object in the scene. On initialization, the silhouette mask along with the current camera image are used to calculate the initial tclc-histograms that provide the pixel-wise posterior segmentation for the first pose optimization.

However, during successful tracking, from the more perspectives the object is being seen the more tclc-histograms get filled with information due to the temporal consistency update strategy. As shown in [TSS17], once initialized, tclc-histograms can act as an object descriptor for pose detection based on template matching (see Section 4.6.2). This strategy works particularly well for pose recovery from temporary tracking loss e.g. caused by massive occlusion or if the object leaves the camera's field of view. This detection strategy can also be used to start tracking in the first place. For this, when available, previously filled tclc-histograms (with help of manual initialization) are loaded in the beginning for bootstrapping the system.

4.6 Pose Tracking and Detection

Current region-based pose estimation systems (including the one presented in this work) are too computationally demanding to perform tracking by detection at real-time frame rates. Therefore, the proposed solution based on tclc-histograms is a

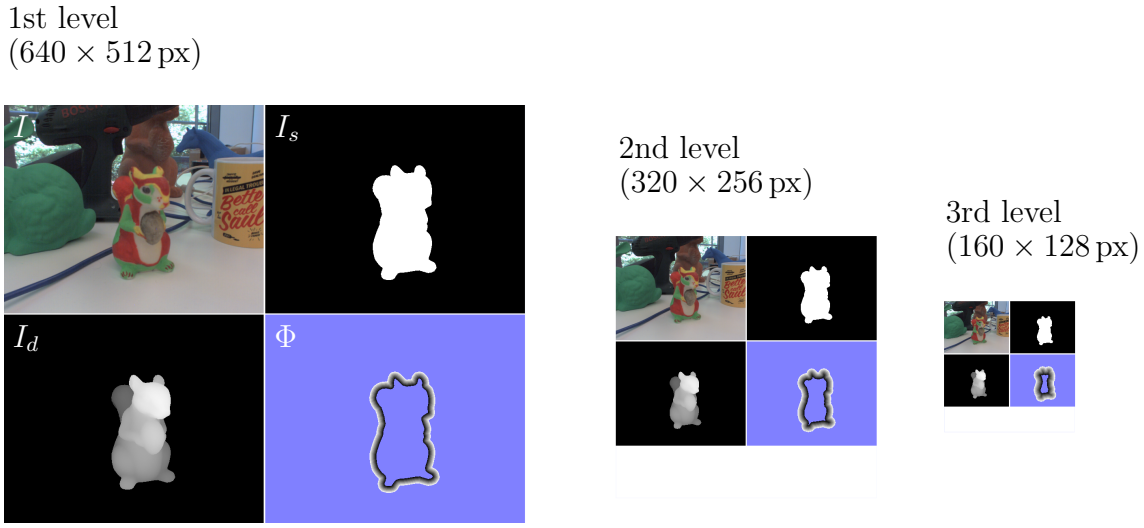


Figure 4.14: A visualization of the three-level image pyramid used for coarse-to-fine pose optimization. Here shown at all three levels are the input RGB camera image I, the rendered silhouette mask I_s and depth map I_d as well as the level-set Φ visualized in the ± 8 px band around the contour (grey pixels).

hybrid between so-called *coarse-to-fine* pose tracking (Section 4.6.1) and template matching strategy for pose detection (Section 4.6.2) to recover from tracking losses. It is thus the first level-set-based pose estimation approach to address the problem of pose detection. For this, a unique similarity measure (4.10) is used for both template matching and pose optimization.

4.6.1 Hierarchical Pose Tracking

For frame-to-frame tracking the iterative pose optimization of Section 4.5 is performed hierarchically within a three level image pyramid generated with a downscale factor of 2 (see Figure 4.14). For this, each new camera frame is scaled down to all pyramid levels in the beginning before pose optimization. Accordingly, in each iteration the silhouette mask I_s and the depth maps I_d and I_d^r are rendered with the respective image resolution using the current updated pose T_{cm}^t. Here, hierarchical rendering is achieved by simply adjusting the width and height of the OpenGL viewport according to the current pyramid level. The 3rd level thereby corresponds

to the camera matrix $^1/_4K$, the 2nd to $^1/_2K$ and the 1st to K, with

$$^1/_nK \doteq \begin{bmatrix} \frac{f_x}{n} & 0 & \frac{c_x}{n} \\ 0 & \frac{f_y}{n} & \frac{c_y}{n} \\ 0 & 0 & 1 \end{bmatrix}$$

used to compute the respective derivatives.

In the proposed real-time implementation, first four iterations are performed on the third, followed by two iterations on the second and finally a single iteration on the first level i.e. the original full image resolution. The number of iterations per level can of course be increased for offline applications in order to improve the pose accuracy for fast motion even further. Regardless of the pyramid level the Heaviside function $H_e(\Phi(\mathbf{x}))$ (4.3) is used with $s = 1.2$. Therefore, the pose optimization only needs to be performed within a band of ± 8 px around each contour $\mathbf{C}$ i.e. $\forall \mathbf{x} \in \Omega$ with $\Phi(\mathbf{x}) \in [-8, 8]$. For other distances the corresponding Dirac delta value $\delta_e(\Phi(\mathbf{x}))$ (4.4) vanishes (see Figure 4.4). Since δ_e scales all other derivates per pixel (see (4.16)), those outside this narrow band have a neglectable influence on the overall optimization.

4.6.2 Template Matching-based Pose Detection

For each frame, E_{rb} (4.10) is evaluated after pose optimization. If $^{E_{rb}}/_{|\Omega|} > e_t$, where e_t is a prescribed threshold, the tracking is considered to be lost. Especially for a handheld object, no assumptions can me made about the pose when it becomes visible again, for example, when it is moved outside the field of view and back in from a different side or held upside down. Once the tracking has been lost, a full search is therefore performed for the object and its current pose in each subsequent frame until pose recovery.

The proposed strategy for re-localization is generally related to current state of the art template matching-based pose detection methods [HLI+12, HCI+12, RT13, KHKH16]. A template view consists of a Heaviside representation $H_e(\Phi)$ of a contour $\mathbf{C}$ obtained from a specific pose and an associated set of tclc-histograms along

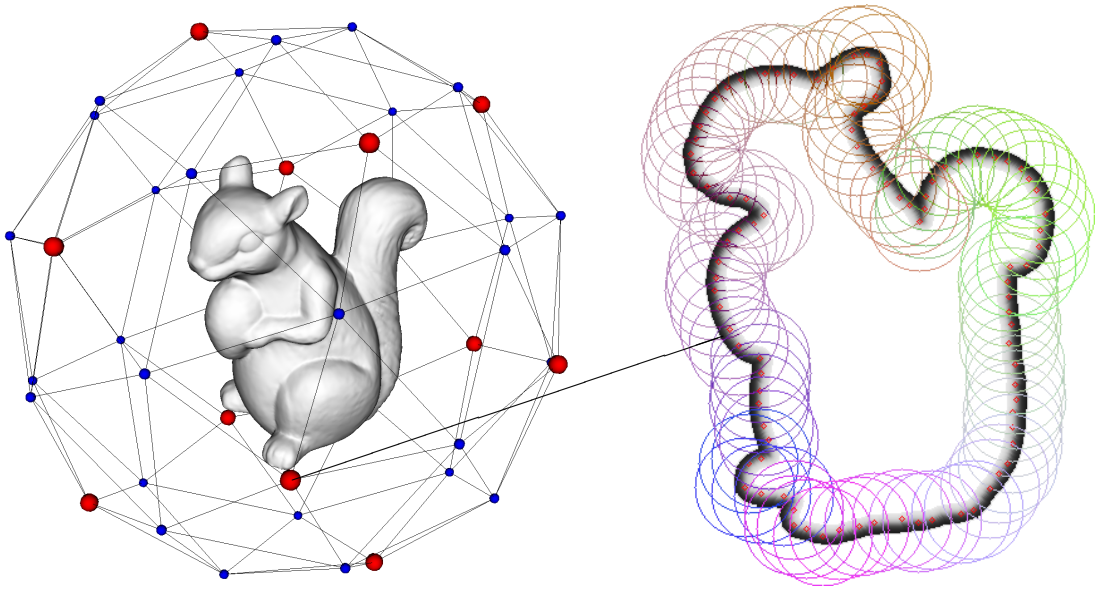


Figure 4.15: The template views used for pose detection. Left: A subdivided icosahedron generates the outer plane rotation of the template views. Red dots indicate vertices corresponding to the base templates, blue dots indicate those provided by subdivision used for the neighboring templates. Right: An example base template visualized by the corresponding $H_e(\Phi)$ (grey pixels) with the local histogram regions depicted (colored circles) along the contour.

it. Here, the orientation of the so-called *base templates* is given by one of the twelve corner vertices of an icosahedron, defining the outer image plane rotation α and β using Euler angles (see Figure 4.15). The base templates are augmented with four different rotations within the image plane, using $0°, 90°, 180°$ and $270°$ for γ. In order to cover different scales, each of these 48 base orientations is used to generate a template at a close, an intermediate and a far distance to the camera, resulting in overall 144 base templates. As soon as all histograms corresponding to a template have been filled, either during a dedicated training stage or regular tracking, the template can be used for re-localization.

The 2D template matching starts at an additional 4th level of an image pyramid (e.g. 80×64 px resolution) that is exclusively used for pose detection. Here an exhaustive search is performed for all base templates by evaluating (4.10) in a sliding-window approach across the image. This first matching step is split into two stages in order to reduce the number of function evaluations and thereby improve the run-

time. The relatively wide basin of convergence for in-plane translation of (4.8) and (4.10) (see [HH16] for a detailed analysis) allows to use a stride of four pixels in the first stage to get a rough estimate of the best matching 2D location of each base template.

In the second stage, this location is refined considering all pixels in a 5×5 neighborhood around the coarse estimate. Here, both stages of sliding-window searches are performed in parallel per base template on the CPU. Inspired by the gradient response maps of LINE-2D [HCI+12], in [TSS17] *posterior response maps* have been introduced. They are a novel image representation based on tclc-histograms, that allow to skip image regions within template matching by a rough statistical pixel-wise foreground or background membership decision. For this, a posterior response map I_p is a binary representation of the current camera image I (see Figure 4.16) defined as

$$I_p(\mathbf{x}) \doteq \begin{cases} 1 & \text{if} \quad \bar{P}_{f_*}(\mathbf{x}) > \bar{P}_{b_*}(\mathbf{x}) \\ 0 & \text{else} \end{cases}, \quad \forall \mathbf{x} \in \Omega,$$

with

$$\bar{P}_{f_*}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} P_{f_i}(\mathbf{x}) \quad \text{and} \quad \bar{P}_{b_*}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} P_{b_i}(\mathbf{x}),$$

being the average posterior probabilities over all histograms regardless of the pixels' locations, computed from (4.9). Note, that in a practical implementation the computation of I_p can be vastly speeded up by using a look-up table. Given this representation, the binary overlap between the silhouette mask I_s of a template at each potential 2D matching location and I_p can be computed. If this intersection is less than 50% of the area of Ω_f, this location is skipped directly without evaluating the energy function, which reduces the number of necessary computations.

To further refine the orientation, the search continues in the next higher resolution of the image pyramid (160×128 px). For this, two out of three distances per base templates are discarded by keeping only that with the best matching score. For the remaining base templates the matching score of the so-called *neighboring orientation templates* is computed at the previously estimated 2D location of its corresponding base template. This is again performed in parallel per base template on the CPU. These neighboring templates were generated at the corner vertices

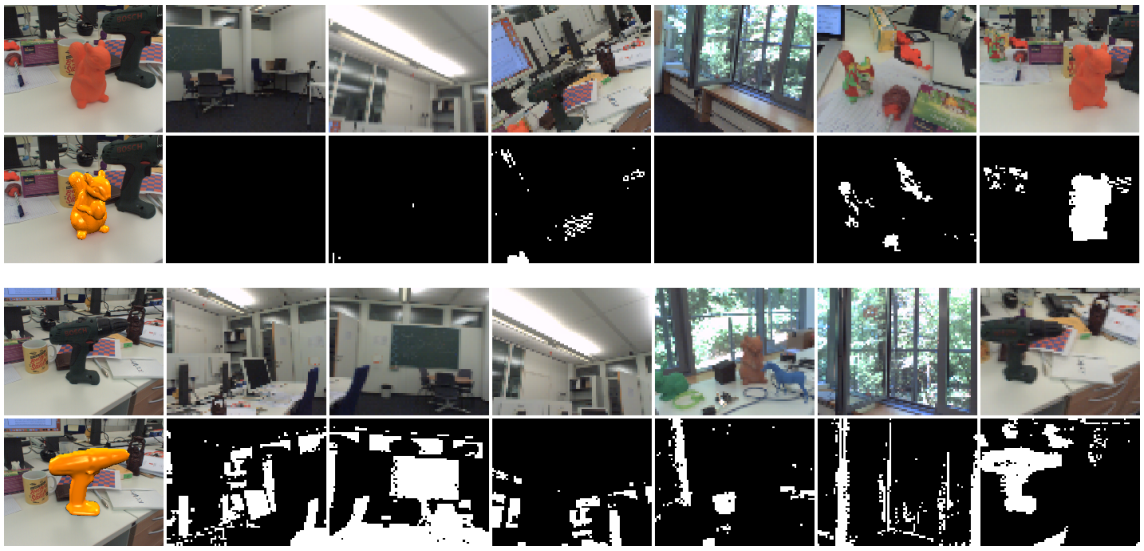


Figure 4.16: Examples of posterior response maps computed during pose detection for a bright homogeneous (top row of image pairs) and a dark heterogenous object (bottom row of image pairs). On the left is an input image of the respective object at full resolution (top) with a visualization of the tracked pose (bottom). On the right of that are some input images I at the 4th pyramid level used for pose detection (top) and the corresponding posterior response maps I_p (bottom), shown upscaled to match the first image.

resulting from sub-dividing the icosahedron (see Figure 4.15) in order to finer sample the outer plane rotation. They are augmented with twelve in-plane rotations of $0°, 30°, \ldots, 330°$. Each base template is associated with eighteen neighboring templates in the next pyramid level. These include itself and those corresponding to the five closest vertices of the subdivided icosahedron, each with the same in-plane rotation as the base template as well as those with $\pm 30°$. The poses of the four best matches of all neighboring templates are finally optimized as described in section 4.6.1 with three times as many iterations as used for tracking.

If ${}^{E_{rb}}/_{|\Omega|}$ corresponding to one of these optimized template poses is smaller than the threshold e_t, the re-localization is considered successful and it is switched back to frame-to-frame tracking. Increasing the value of e_t thus results in both the tracking and re-localization to be more tolerant to slight misalignment and false positive detections. This can help to improve the overall performance in case of heterogenous objects and background clutter if continuity is more important than precision. For

the experiments in Section 4.8 this threshold was chosen $e_t \in [0.5, 0.6]$.

4.7 Extension to Multiple Objects

So far this chapter has only dealt with region-based pose estimation of a single object. In the following it will be explained how the proposed approach can easily be extended to tracking and detecting multiple objects simultaneously.

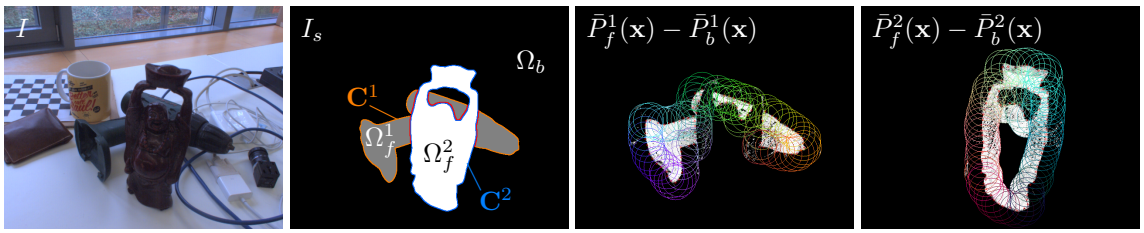
4.7.1 Extended Notation

In case of multiple objects, each is represented by its own 3D model, with corner vertices $\mathbf{X}_i^j$, $j = 1, \ldots, m$, where m is the total number of objects. Accordingly, the individual poses are denoted by T_{cm}^j. Projecting all models into the image plane yields a common segmentation mask $I_s : \Omega \to \{0, \ldots, m\}$. In OpenGL this can be obtained by rendering each model using a unique intensity corresponding to the model index j. Thus the common I_s contains m contours $\mathbf{C}^j$ that split the image into multiple foreground regions $\Omega_f^j \subset \Omega$ and background regions $\Omega_b^j = \Omega \setminus \Omega_f^j$ (see Figure 4.17).

For pose tracking and detection a separate energy function for each object is optimized and evaluated, denoted by

$$E_{rb}^j(\xi^j) = -\sum_{\mathbf{x}\in\Omega} \log\big(H_e(\Phi^j(\mathbf{x})(\xi^j))\bar{P}_f^j(\mathbf{x}) + (1 - H_e(\Phi^j(\mathbf{x})))\bar{P}_b^j(\mathbf{x})\big),$$

with its own level-set Φ^j and region membership probabilities $\bar{P}_f^j(\mathbf{x})$ and $\bar{P}_b^j(\mathbf{x})$ computed from its individual set of n^j tclc-histograms. Here n^j gives the number of vertices per model. Each such optimization is performed regardless of the other objects as long as they do not occlude each other. However, in cases of mutual occlusions, the foreground regions overlap, which results in contour segments that do not belong to the actual silhouette of the objects (see again Figure 4.17). These



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

Figure 4.17: An illustration of the multi-object tracking scenario. An input color image I shows a driller lying behind a Buddha figurine (a). The estimated poses allow to render a common silhouette mask I_s. Here the segments of $\mathbf{C}^1$ that appear due to the occlusion are marked red (b). Based on I_s the corresponding per pixel object segmentations (visualized as $\bar{P}_f^j(\mathbf{x}) - \bar{P}_b^j(\mathbf{x}) > 0$) are computed from the respective tclc-histograms of both the driller (c) as well as the statue (d).

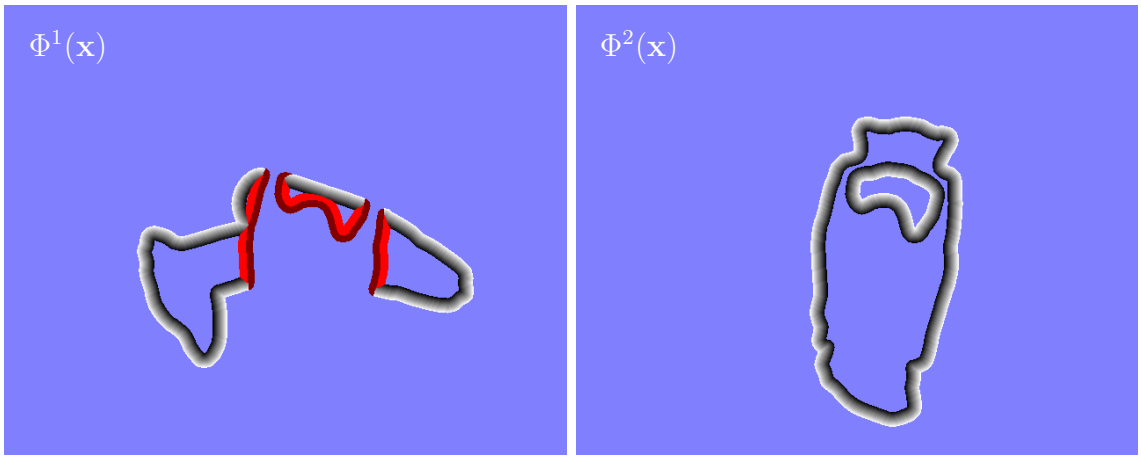
must be detected and handled appropriately during pose optimization as explained in detail in the following Section 4.7.2.

This shows that for all formulas extending the proposed approach to multiple objects is essentially done by adding the object index j. For the sake of clarity, the index j will therefore be dropped again in the remainder of this chapter, unless absolutely required.

4.7.2 Occlusion Handling

As previously shown in Figure 4.17, when tracking multiple objects simultaneously mutual occlusions are very likely to emerge. These must be handled appropriately on a per pixel level for pose optimization. In the proposed approach occlusions can be detected with help of the common silhouette mask I_s due to the Z-buffer OpenGL. Thus, the respective contours C^j, computed directly from I_s, can contain segments resulting from occlusions that are considered in the respective signed distance transform. To handle this, for each object all pixels with a distance value that was influenced by occlusion have to be discarded for pose optimization (see Figure 4.18).

A straightforward approach as realized in [PR12] is to render each model's silhou-



(a) (b)

Figure 4.18: The two level-sets corresponding to $\mathbf{C}^1$ (a) and $\mathbf{C}^2$ (b) of Figure 4.17 visualized in the ± 8 px band around the contours (grey pixels). Here the distance values of $\Phi^1(\mathbf{x})$ that are influenced by the occlusion of $\mathbf{C}^1$ are marked red (bright inside and dark outside of Ω_f^1).

ette I_s^j separately as well as the common silhouette mask I_s. The signed distance transforms Φ^j are then computed from the non-occluded I_s^j, where I_s is only used to identify whether a pixel belongs to a foreign object region and thus has to be discarded.

Although this strategy is easy to implement, it does not scale well with the number of objects since all $I_s^j, I_d^j, (I_d^r)^j$, $j = 1, \ldots, m$ and I_s have to be rendered and transferred to host memory in each iteration. In order to minimize rendering and memory transfer, a different strategy can be used, that was originally presented in [TSS16]. There, the entire scene is rendered once per iteration and the common silhouette mask I_s and the respective depth-buffer I_d are downloaded. The individual level-sets Φ^j are then directly computed from I_s. In addition to this, each model has to get rendered once separately, only in order to obtain and download the individual reverse depth buffers $(I_d^r)^j$. This is not possible in a common scene rendering because the reverse depths of the occluded object would overwrite those of the object in front.

By only using I_s and I_d the detection of pixels with a distance value that was influenced by occlusion is split into two cases. For a pixel $\mathbf{x} \in \Omega_b^j$ outside of the silhouette region i.e. $\Phi^j(\mathbf{x}) > 0$, it is first checked whether $I_s(\mathbf{x})$ equals another

object index using the common mask rendering. If so, $\mathbf{x}$ is discarded if also the depth at $I_d(\mathbf{x})$ of the other object is smaller than that of the closest contour pixel to $\mathbf{x}$, meaning that the other surface is actually in front of the current object (indicated with dark red in Figure 4.18). For $\mathbf{x} \in \Omega_f^j$ inside of the silhouette region i.e. $\Phi^j(\mathbf{x}) \leq 0$, the same checks are performed for all neighboring pixels outside of Ω_f^j to the closest contour pixel to $\mathbf{x}$. If any of these pixels next to the contour passes the mask and depth checks, $\mathbf{x}$ is discarded (indicated with bright red in Figure 4.18).

4.8 Evaluation

In the following, a comprehensive experimental evaluation of the proposed region-based approach will be given. Here, pose tracking and detection are evaluated separately. Pose tracking is evaluated in the newly constructed RBOT dataset (see Section 4.8.1), originally presented in [TSSC18]. The proposed pose detection strategy was evaluated in the popular publicly available LINE-MOD dataset of [HLI+12]. For the proposed region-based method, the runtime not only depends on the number of objects, but also on their distance to the camera (that impacts the number of processed pixels) and their 3D models' complexity (polygon count). These factors hamper general performance measurements. Therefore, in the following the average runtime will be given along with each of the numerous experiments. The proposed implementation was run on a commodity laptop with an Intel Core i7 quad core CPU @ 2.8 GHz and an AMD Radeon R9 M370X GPU.

4.8.1 The RBOT Dataset

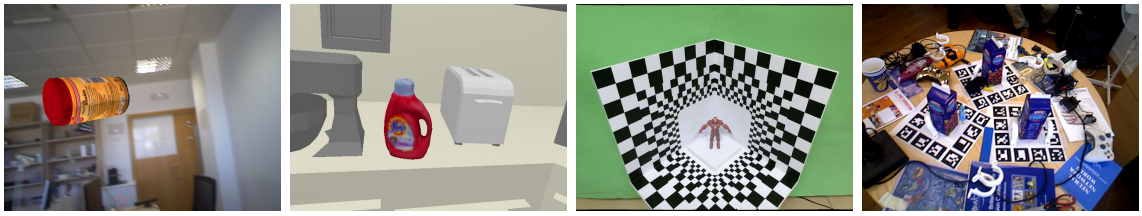
The *RBOT* (*Region-based Object Tracking*) dataset was developed within the course of this thesis. It is the first to address the common and important scenario in which both the camera as well as the objects are moving simultaneously in cluttered scenes. It thereby closed a gap in literature regarding previous publicly available datasets for monocular 6DOF pose tracking of 3D objects as explained in the following.

There are several different aspects that can potentially be covered by an object

pose tracking dataset. One important feature is the type and complexity of motion included in the respective image sequences. Here, the *type* of motion refers to whether they include either movement of only the camera, only the object or both simultaneously. This is particularly important because it has a direct impact on e.g. the intensity gradients (i.e. shading) inside the object region that depend on the lighting in the scene and how quickly the background changes. Another aspect is whether the light sources in the scene are moving. It is closely related to the case of object motion but typically has an even stronger impact on the objects appearance (e.g. self-shadowing). With regard to the previously discussed gradient-based image features, it should also be considered to include both well-textured as well as weakly textured objects. Datasets can furthermore contain a single or multiple objects, occlusions, different amounts of background clutter and motion blur to simulate common problems in real scenarios.

Another essential question when generating a dataset for passive object tracking is how the ground truth pose information is obtained for each frame, since this is basically a chicken-egg problem, as explained in Section 3.2.8. One popular and straightforward approach is to render synthetic image sequences from artificial and thus fully controllable 3D scenes. However, the resulting images often lack photo-realism and require sophisticated rendering techniques and expert knowledge in 3D animation. On the other hand, in case of real-data image sequences typically some sort of fiducial markers are placed in the scene to provide a reference pose from all perspectives independent of the rest of the scene. This requires the relative pose between markers and object to remain static at all times. Datasets of this kind thus often only contain motion of the camera unless markers are also attached to the object which however changes its appearance in an unnatural and undesired way.

In the context of 6DOF object pose tracking a third, sort of intermediate strategy can be applied, where semi-synthetic image sequences are created, combining advantages of both previous solutions. Here, animated renderings of a realistically textured 3D model are composed with a real image sequence providing for a background in each frame. This technique has been used for the *Rigid Pose* dataset presented in [PRDR13], which is considered most closely related to the proposed one. Here, semi-synthetic stereo image pair sequences of six different objects are provided, each available in a noise-free, a noisy and an occluded version. However, five out of the six



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

Figure 4.19: Example images extracted from different related pose tracking datasets: *Rigid Pose* [PRDR13] (a), *RGB-D Object Pose Tracking* [CC13] (b), *OPT* [WLT+17] (c) and [TTKK14] (d).

objects used within this dataset are particularly well textured. Also the objects are rendered using a Lambertian illumination model without including any directional light source, meaning that the intensity of corresponding pixels between frames does not change. These renderings are furthermore simply pasted onto real images without e.g. blurring their contours in order to smooth the transition between the object and the background (e.g. Figure 4.19 (a)).

Then there is the *RGB-D Object Pose Tracking* dataset of [CC13] that contains two real-data and four fully synthetic sequences including four different objects that are static in the scene. However, ground truth information is only provided for the four synthetic sequences which were primarily designed for depth-based tracking. Here, apart from the respective object itself the rest of the scene is very simple and completely texture-less, which is why the resulting RGB color images look very artificial overall (e.g. Figure 4.19 (b)).

Very recently the *OPT* dataset was presented in [WLT+17], being the most complex 6DOF object pose tracking dataset yet. It contains multiple real-data RBG-D sequences of six 2D patterns and six 3D objects that vary in texture and complexity. The images were captured with a camera mounted on a robot arm that moves around each single object at different speeds and varying lighting conditions. The dataset thereby even covers scenarios with a moving light source and contains motion blur. Despite its complexity the dataset does not include object motion, background clutter or occlusions, since in all sequences the object is placed statically in front of an entirely white background surrounded by a passive black and white marker pattern

(e.g. Figure 4.19 (c)).

Lastly, there is the dataset of [TTKK14], that contains six real-data RBG-D sequences involving six different objects and partial occlusions. In each sequence multiple static instances of the same object are placed on a cluttered table each surrounded by a marker pattern (e.g. Figure 4.19 (d)). Therefore, the dataset also only contains movement of the camera. Although this dataset does provide test sequences of consecutive video frames, it was primarily designed for the task of 6DOF object pose detection. In that case the object pose is supposed to be recovered from only a single image as opposed to an image sequence in case of pose tracking. Other pose detection datasets (e.g. the LINE-MOD dataset of [HLI+12]) typically do not contain any consecutive frames at all and can therefore not be used for pose tracking, although the two tasks are strongly related.

To summarize, there are currently only a few datasets that have been created explicitly for the task of monocular 6DOF pose tracking. Of those available, [PRDR13, CC13, TTKK14] are relatively small and do not cover many of the initially mentioned aspects. The most complex dataset currently available [WLT+17] unfortunately neither simulates scenarios in which both the object and the camera are moving (e.g. a hand-held object in a cluttered environment), which is targeted in this work.

Thus in [TSSC18] the novel large semi-synthetic RBOT dataset was proposed, that covers most of the previously mentioned important aspects. It was made publicly available for download under: http://cvmr.mi.hs-rm.de/research/RBOT. It comprises a total number of eighteen different objects, all available as textured 3D triangle meshes. In addition to a model of a squirrel clay figurine, a selection of twelve models from the LINE-MOD dataset [HLI+12] and five from the Rigid Pose dataset [PRDR13] were included, as shown in Figure 4.20.

For each model, four variants of a semi-synthetic image sequence were generated with increasing complexity (see Figure 4.21). The first *regular* variant contains a single object rendered with respect to a static point light source located above the origin of the virtual camera. This simulates a moving object in front of a static camera. The second variant is the same as the first but with a dynamic light source in order to simulate simultaneous motion of both the object and the camera. The

Figure 4.20: An overview of all eighteen models included in the RBOT dataset. The well-textured models from the Rigid Pose tracking dataset [PRDR13] are marked with■ and the weakly-textured models from the LINE-MOD detection dataset [HLI+12] are marked with♦. Here all models are displayed with the same pose for scale comparison.

images in the third variant were also rendered with respect to the moving light source and further distorted by adding artificial Gaussian noise. Finally, the fourth variant contains an additional second object (the squirrel) which orbits around and thus frequently occludes the first object. These multi-object sequences also include the dynamic light source.

Regardless of the object and the variant, the model renderings were animated using the same pre-defined trajectory of continuous 6DOF motion in all sequences. Also, always the same background video was used for compositing. This video was recorded while moving a hand-held camera arbitrarily in a cluttered desktop scene. In order to increase the realism of the semi-synthetic images, the models were rendered with anti-aliasing and the object regions were blurred in the composition using a 3×3 Gaussian kernel. The latter in particular smooths the transition between the object and the background making it blend more realistically with the rest of the scene. Each sequence contains 1001 RGB frames of 640×512 px resolution, where the first is typically used for initialization. This makes a total number of $72000 = 18 \cdot 4 \cdot 1000$ color images that can be used to evaluate monocular pose tracking algorithms. For each frame the ground truth poses are provided for the two objects along with the intrinsic camera matrix used for rendering which was

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

Figure 4.21: An example frame from the RBOT dataset with the duck model in the four different variants: the regular (a), the dynamic light (b), the noisy (c) and multi-object version (d) with occlusions. Visually comparing the regular with the dynamic light version clarifies the impact of the lighting aspect on the appearance of the object region.

obtained from calibrating the camera that recorded the background video.

4.8.2 Tracking Results in the RBOT Dataset

Let the sequence of ground truth poses be denoted by $T^j_{gt}(t_k)$, composed of $R^j_{gt}(t_k)$ and $\mathbf{t}^j_{gt}(t_k)$ with $k = 0, \ldots, 1000$. Starting at $T^j_{cm}(t_0) = T^j_{gt}(t_0)$, for each subsequent frame the tracking error is computed separately for translation $e^j_k(\mathbf{t}) = \|\mathbf{t}^j_{cm}(t_k) - \mathbf{t}^j_{gt}(t_k)\|_2$ and rotation

$$e^j_k(R) = \cos^{-1}\left(\frac{\operatorname{trace}(R^j_{cm}(t_k)^\top R^j_{gt}(t_k)) - 1}{2}\right),$$

for each object to evaluate the tracking success rate. If $e^j_k(\mathbf{t})$ is below 5 cm and $e^j_k(R)$ below 5°, the pose is considered to be successfully tracked, in relation to [SGZ+13]. Otherwise if one of the errors is not within its boundaries, the tracking is considered to be lost and reset to the ground truth pose by $T^j_{cm}(t_k) = T^j_{gt}(t_k)$. In the corresponding experiments the multi-object sequences were evaluated in two different ways. Here, either only the pose of the first (varying) object or both of them (the varying object and the occluding squirrel) are tracked. When tracking only one of the objects, the occurring occlusions are here referred to as *unmodelled*. These are much harder to handle than *modelled* occlusions, where the pose and geometry of

Variant	Method	Ape	Baking Soda	Bench Vise	Broccoli Soup	Camera	Can	Cat	Clown	Cube	Driller	Duck	Egg Box	Glue	Iron	Koala Candy	Lamp	Phone	Squirrel
Regular	[TSS17]	62.1	30.5	95.8	66.2	61.6	81.7	**96.7**	89.1	44.1	87.7	74.9	50.9	20.2	68.4	20.0	92.3	64.9	98.5
	[TSSC18]	**85.0**	**39.0**	**98.9**	**82.4**	**79.7**	**87.6**	95.9	**93.3**	**78.1**	**93.0**	**86.8**	**74.6**	**38.9**	**81.0**	**46.8**	**97.5**	**80.7**	**99.4**
Dynamic Light	[TSS17]	61.7	32.0	94.2	66.3	68.0	84.1	**96.6**	85.8	45.7	88.7	74.1	56.9	29.9	49.1	20.7	91.5	63.0	98.5
	[TSSC18]	**84.9**	**42.0**	**99.0**	**81.3**	**84.3**	**88.9**	95.6	**92.5**	**77.5**	**94.6**	**86.4**	**77.3**	**52.9**	**77.9**	**47.9**	**96.9**	**81.7**	**99.3**
Noisy +	[TSS17]	55.9	35.3	75.4	67.4	27.8	10.2	94.3	33.4	8.6	50.9	76.3	2.3	2.2	18.2	11.4	36.6	31.3	93.5
Dynamic Light	[TSSC18]	**77.5**	**44.5**	**91.5**	**82.9**	**51.7**	**38.4**	**95.1**	**69.2**	**24.4**	**64.3**	**88.5**	**11.2**	**2.9**	**46.7**	**32.7**	**57.3**	**44.1**	**96.6**
Unmodelled	[TSS17]	55.2	29.9	82.4	56.9	55.7	72.2	87.9	75.7	39.6	78.7	68.1	47.1	26.2	35.6	16.6	78.6	50.3	77.6
Occlusion	[TSSC18]	**80.0**	**42.7**	**91.8**	**73.5**	**76.1**	**81.7**	**89.8**	**82.6**	**68.7**	**86.7**	**80.5**	**67.0**	**46.6**	**64.0**	**43.6**	**88.8**	**68.6**	**86.2**
Modelled	[TSS17]	60.3	31.0	94.3	64.5	67.0	81.6	92.5	81.4	43.2	89.3	72.7	51.6	28.8	53.5	19.1	89.3	62.2	96.7
Occlusion	[TSSC18]	**82.0**	**42.0**	**95.7**	**81.1**	**78.7**	**83.4**	**92.8**	**87.9**	**74.3**	**91.7**	**84.8**	**71.0**	**49.1**	**73.0**	**46.3**	**90.9**	**76.2**	**96.9**
Avg. Runtime	[TSSC18]	15.5	15.7	20.3	16.6	17.4	18.9	16.9	15.9	16.8	19.4	15.7	16.8	16.1	19.1	16.5	21.8	17.6	19.1

Table 4.1: Tracking success rates (in %) of the proposed method of [TSSC18] in comparison to that of [TSS17] in the RBOT dataset. In the last row furthermore the average runtimes of the proposed approach are denoted (in ms) measured across the first three variants. These experiments confirm that the systematic derivation of a Gauß-Newton optimization suggested in Section 4.5.1 leads to a significant performance improvement over [TSS17] for almost all objects.

the occluding object is known, in case of tracking both objects. Here, such modelled occlusions were considered using the approach described in Section 4.7.2.

Table 4.1 presents the tracking success rates of the proposed method using tclc-histograms and the true Gauß-Newton optimization [TSSC18] compared to the method in [TSS17] using the Gauß-Newton like strategy for all sequences in all variants. Other methods such as PWP3D [PR12] or [HH16] were not included, because previous experiments indicated that they would not perform competitively in such a complex dataset (see again e.g. Figure 4.9 and Figure 4.11). The results show that the re-weighted Gauß-Newton optimization outperforms the previous Gauß-Newton-like strategy in most cases by a large margin.

The experiments also demonstrate the robustness of the appearance model based on tclc-histograms towards a moving light source. For both methods, compared to the regular scenario, the performance often even improves in the dynamic light variant and hardly deteriorated otherwise. However, it can also be seen that both approaches perform significantly worse for objects with ambiguous silhouettes (e.g. Baking Soda, Glue and Koala Candy) and struggle more with image noise in case of objects of a less distinct color (e.g. Camera, Can, Cube, Egg Box etc.).

4.8.3 Detection Results in the LINE-MOD Dataset

The performance of the re-localization strategy of Section 4.6.2 was evaluated experimentally in [TSS17]. Here it was compared to using the localized energy (4.10) of [HH16] within the proposed strategy as well as LINE-2D [HCI+12] and the method of [BMK+16]. For this, the popular so-called *LINE-MOD* single instance RGB-D pose detection dataset of [HLI+12] was used, including 13 of the 15 provided models (for two of the models no proper triangle mesh was given) and all corresponding RGB color images. Here, the training stage of the tclc-histograms was simulated for each model by projecting it with the given ground truth pose information in a randomly picked subset of the test images. After this initialization, it was tried to detect the pose of the object in the entire set of test images (around 1000 - 1200 per object). Figure 4.22 shows the results for each individual model trained with 25, 50, 100, 200 images using the proposed method.

Here, in relation to [BMK+16], four different error metrics were used to compare the detected pose $T_{cm}(t_k)$ to the given ground truth $T_{gt}(t_k)$ in each image for computing detection success rates. For the first metric the two silhouette projections were computed using both $T_{cm}(t_k)$ and $T_{gt}(t_k)$ in order to determine the exact 2D bounding boxes containing them. Here, the detection is considered successful if the intersection over union (IoU) of these 2D bounding boxes is at least 50%. For the second metric the average 2D projection error is computed as

$$e_k^{proj} = \frac{1}{n} \sum_{i=1}^{n} \| \mathbf{x}_i \left(T_{cm}(t_k) \right) - \| \mathbf{x}_i \left(T_{gt}(t_k) \right) \|_2,$$

where a successful pose detection is counted, if e_k^{proj} is below 5 px. The third metric was adopted from [HLI+12]. It is computed as

$$e_k^{diam} = \frac{1}{n} \sum_{i=1}^{n} \min_{\mathbf{X}'_i, i=1,\ldots,n} \| (T_{cm}(t_k)\tilde{\mathbf{X}}_i)_{3\times 1} - (T_{gt}(t_k)\tilde{\mathbf{X}}'_i)_{3\times 1} \|_2,$$

which is thereby invariant to rotation symmetries. Here, pose detection is considered successful if $e_k^{diam} \leq 0.1 diam$, with $diam$ being the the model diameter (i.e. the largest distance between two points). Finally, the fourth metric is the same as

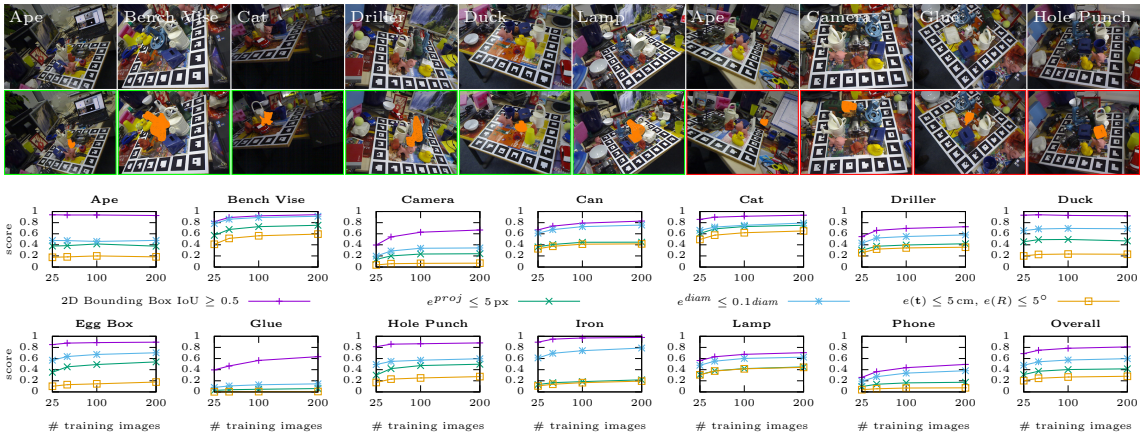


Figure 4.22: Top: Examples of the pose detection experiment (1st row: Original input images from the LINE-MOD dataset. 2nd row: Visual results of successful pose detection (green outline) and failure cases (red outline) of the proposed method.). Bottom: Detection scores of the proposed method separately visualized for each model as well as the overall average of all models of the LINE-MOD dataset with respect to the number of randomly picked training images averaged over 10 runs.

previously used for tracking in Section 4.8.2, meaning that pose detection is counted successful if $e_k(\mathbf{t}) \leq 5\,\text{cm}$ and $e_k(R) \leq 5°$. Here, the latter is most important in the context of pose recovery for subsequent tracking.

The results in Figure 4.22 show that even for very small amounts of training images the proposed method performs well for most of the models and the improvement quickly converges when their number is increased. As expected, here the exact pose can be recovered especially well for models with a distinct color and silhouette (e.g. Bench Vise or Cat) while it gets confused with regions of similar shapes and color.

Table 4.2 gives the overall results of the proposed method (with (4.10) and (4.8)) using 100 training images to those presented in [BMK+16] including their LINE-2D implementation. Firstly, this confirms the advantage of using (4.10) as cost function over (4.8). Secondly, it can also be seen that the proposed approach overall performs significantly better than LINE-2D, which runs at comparable speed ($\sim$10 Hz). However, the proposed method cannot be directly compared to it for two reasons. The first is, that the proposed method was trained with images from the dataset

Metric	Proposed with (4.10)	Proposed with (4.8)	[HCI+12] LINE-2D	[BMK+16] w. L_1 reg.
2D IoU ≥ 0.5	78.5	68.5	86.5	**97.5**
$e^{proj} \leq 5\,\text{px.}$	40.2	28.8	20.9	**73.7**
$e^{diam} \leq 0.1 diam$	**57.3**	49.8	24.2	50.2
$e(\mathbf{t}) \leq 5\,\text{cm}$, $e(R) \leq 5°$	26.7	18.6	8.1	**40.6**
Avg. Runtime (s)	~ 0.15	~ 0.17	$\sim$**0.1**	$\sim 1-2$

Table 4.2: Overall detection score comparison in the LINE-MOD dataset. Detection success rates (in %) of the proposed method of [TSS17] are presented in comparison to those of using the localized energy (4.8) of [HH16] instead of (4.10) as well as LINE-2D [HCI+12] and the method of [BMK+16]. In the last row, furthermore the average runtimes of all four considered approaches are denoted (in seconds) measured across the entire dataset.

including the backgrounds, which is required in order to fill our tclc-histograms, while LINE-2D can be trained without scene knowledge. However, this constraint is considered acceptable for the proposed approach, since it is mainly designed to recover from temporary tracking losses in known scenes other than detecting objects in arbitrary scenes. On the other hand, within LINE-2D only templates of the upper hemisphere for outer plane rotation and ±80° in-plane rotation are used, while the proposed approach is potentially able to detect the object at all possible rotations.

The proposed pose detection implementation typically runs at 4–10 Hz when including the full sphere for outer plane and 360° for in-plane rotation. However, in cases when the object is not present in the current image, it can go up to 100 Hz depending on how many regions can directly be skipped with help of the posterior response map. Note, that the runtime of this approach decreases almost linearly with respect to the number of templates. This becomes relevant in case of scenarios that are more constrained for viewpoints (e.g. only require the upper hemisphere).

Finally, although the proposed method performs worse in most cases compared to [BMK+16], it should be pointed out that this method also uses a subset of the original images (15% i.e. ∼170 images) per model for training their random forests, while detection takes 1 – 2 seconds per image.

4.9 Summary

In this chapter a novel region-based 6DOF pose estimation approach was presented that was developed within this course of this dissertation. Along with this, a new large semi-synthetic object pose tracking dataset was created and made available to the community. It covers many practically relevant and challenging scenarios beyond any previously released dataset for passive, monocular 6DOF object pose tracking.

The proposed pose estimation method uses novel tclc-histograms and is thereby the only region-based hybrid solution for both pose tracking and detection of weakly textured and textureless objects. As part of this approach a novel fully analytic derivation of a Gauß-Newton optimization has been presented by formulating the region-based cost function as a weighted nonlinear least-squares problem. It has been shown that this leads to a major improvement of the convergence properties over the previously used gradient descent in the context of region-based pose estimation. Numerous experiments demonstrated that the proposed system thereby currently achieves state of the art results for homogeneous and heterogeneous objects in challenging conditions such as cluttered backgrounds, changing lighting conditions and partial occlusions. Another advantage of this solution is, that it is very light-weight on the hardware side. It only requires a single camera and can be implemented on the CPU using the GPU only for OpenGL rendering. The proposed implementation is currently also state of the art in terms of runtime performance and is thereby the first to be able to handle multiple (two to three) arbitrarily textured objects simultaneously in real-time.

However, the approach also has two main drawbacks. Firstly, the main downside of the proposed pose detection approach is that training the templates (i.e. the tclc-histograms) requires scene or background knowledge. In other words, the descriptor generally struggles in previously unseen environments if the foreground and background color distribution is too different from the scene it was originally trained in. This prevents its application to the classical pose detection task, where a known object is to be detected in arbitrary scenes. However, it does work well for recovering from temporary tracking loss and comes with the significant advantage that

new objects or scenes can be trained within a couple of seconds.

Secondly, both tracking and detection suffer from pose ambiguities that are inherent to methods only relying on silhouette information. For example in case of bodies of revolution (e.g. Baking Soda and Koala Candy from the RBOT dataset) the rotation around one axis cannot be determined. Therefore, in the following Chapter 5 it will be shown how the tracking accuracy can be further improved by incorporating a photometric term in the cost function with regard to the objects' texture. Assuming an object is sufficiently textured this resolves the silhouette ambiguity in many cases.

5 Direct Photometric Pose Estimation

This chapter deals with the general concept of estimating the pose of an object based on direct image alignment. It starts by introducing the basic idea (Section 5.1) and then moves on by showing how it can be applied to 6DOF object pose tracking (Section 5.2). Based on this, a novel cost function is introduced that combines the region-based term of Chapter 4 with the photometric term presented in this chapter (Section 5.3) in order to further increase the tracking robustness. Here it is also shown how this cost function can be optimized with respect to the object's pose using a combined Gauß-Newton strategy. Next, the performance of combined region-based and photometric pose tracking is evaluated and compared to the mere region-based and the mere photometric solution (Section 5.4). The chapter concludes with a summary discussing advantages and drawbacks of the proposed combined pose tracking solution (Section 5.5).

5.1 Direct Image Alignment

Direct photometric pose estimation is based on the so-called *photo-consistency* assumption. This means that the color of any projected surface point should appear to be similar in every image regardless of perspective. Given a pair of images $I(t_r)$

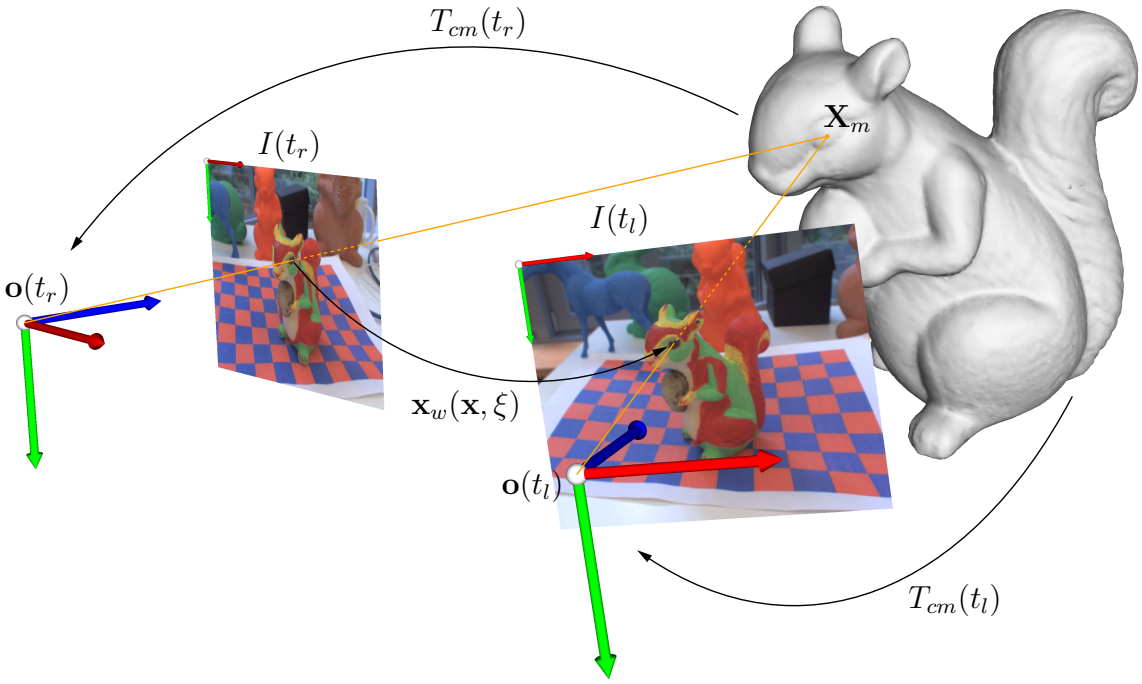


Figure 5.1: An overview of the direct photometric pose estimation setting. Knowing the poses $T_{cm}(t_r)$ and $T_{cm}(t_l)$ any model point $\mathbf{X}_m$ (here within the eye) is assumed to project to a pixel of similar color (here black) in both the reference frame $I(t_r)$ and the current live image $I(t_l)$. The warp function $\mathbf{x}_w(\mathbf{x}, \xi)$ describes the transition from each pixel in the reference frame to the corresponding location in the live image.

and $I(t_l)$, the exact corresponding poses $T_{cm}(t_r)$, $T_{cm}(t_l)$ and a 3D surface point $\mathbf{X}_m$ visible in both images, it should hold

$$I\left(\mathbf{x}(T_{cm}(t_r), \mathbf{X}_m), t_r\right) \approx I\left(\mathbf{x}(T_{cm}(t_l), \mathbf{X}_m), t_l\right), \tag{5.1}$$

with

$$\mathbf{x}(T_{cm}(t_k), \mathbf{X}_m) = \pi\left(K(T_{cm}(t_k)\tilde{\mathbf{X}}_m)_{3\times 1}\right), \text{ for } k = 0, \ldots, l,$$

assuming Lambertian surface reflection properties (see Figure 5.1). In the context of pose estimation, typically the pose $T_{cm}(t_l)$ corresponding to the current live frame $I(t_l)$ is unknown. Here $I(t_r)$ is a reference frame of which the pose $T_{cm}(t_r)$ is known, that serves to estimate the missing pose difference $T_{cc}(t_l, t_r) = T_{cm}(t_l)T_{cm}(t_r)^{-1}$ by performing so-called *direct image alignment* as explained in the following.

Assuming a dense spatial surface model of the entire scene as well as $T_{cm}(t_r)$ are known, a dense depth map $I_d(t_r)$ can be generated that contains the depth values for every pixel in $I(t_r)$ (see e.g. Section 2.2.4). By representing the sought pose with an exponential map as $T_{cm}(t_l) = \exp(\hat{\xi})T^0_{cm}$, one can then define a 6DOF so-called *warp function*

$$\mathbf{x}_w(\mathbf{x}, \xi) \doteq \pi \left(K(\exp(\hat{\xi})T^0_{cm}T_{mc}(t_r)\tilde{\mathbf{X}}_c(I_d(t_r), \mathbf{x}))_{3\times 1} \right) \in \mathbb{R}^2, \tag{5.2}$$

with

$$\mathbf{X}_c(I_d(t_r), \mathbf{x}) = D(I_d(t_r), \mathbf{x})K^{-1}\tilde{\mathbf{x}} \in \mathbb{R}^3,$$

that relates every pixel $\mathbf{x}$ in the reference frame $I(t_r)$ to warped coordinates $\mathbf{x}_w(\mathbf{x}, \xi)$ in the current live frame $I(t_l)$. Thus, given $I_d(t_r)$, this warp is achieved by explicitly projecting the pixel back onto the model surface using (2.33) and then projecting it into the live image (see again Figure 5.1). Accordingly, the photo-consistency constraint (5.1) may be rewritten as

$$I(\mathbf{x}_w(\mathbf{x}, \xi), t_l) - I(\mathbf{x}, t_r) = 0,$$

which eventually leads to a general pixel-wise photometric cost function

$$E_{ph}(\xi) = \frac{1}{2} \sum_{\mathbf{x}\in\Omega} \| I\left(\mathbf{x}_w(\mathbf{x}, \xi), t_l\right) - I\left(\mathbf{x}, t_r\right) \|_2^2, \tag{5.3}$$

in the form of a nonlinear least-squares problem. When minimizing (5.3) with respect to twist coordinates, the two images are aligned by estimating the corresponding warp function. Direct photometric pose estimation can thus be seen as a 6DOF extension of the well-known Lucas-Kanade framework for direct image alignment [LK81, BM04a], which was originally used to compute 2D image transformations.

Since here the pose is estimated passively and only by minimizing the intensity differences between two images, such direct photometric approaches are closely related to those based on image intensity features, as previously introduced in Section 3.3. However, instead of being limited to high contrast keypoint or edge features and matching descriptors computed from small intensity patches, direct methods poten-

tially integrate all pixels in both images by directly comparing their pixel intensities (hence the name *direct*). Due to this large amount of redundant information, it has been shown that these approaches are much more robust towards motion blur and defocus (e.g. [NLD11, New12]). They therefore generally do not require the scenes or objects to yield as strong and descriptive image gradients as needed for computing and matching intensity based features. They do however struggle with changing lighting conditions and non-Lambertian surfaces since these violate the implied photo-consistency assumption. Therefore, in recent years, different strategies have been investigated to make direct methods robust towards varying illumination [SMR12, SL12, CDM14, CL14].

Early direct pose estimation approaches were initially demonstrated for tracking 2D templates [BM04b, SM07, BLLN07, BM07]. A few years later, they became popular in the context of SLAM (simultaneous localization and mapping). Here, the 6DOF camera pose is estimated relative to a dense [NLD11] or semi-dense [ESC14] 3D reconstruction of the scene that gets computed in parallel. More recently, the first direct approach to 3D object tracking was presented in [SW16]. However, this method essentially only works well with textureless, angular objects (e.g. metallic parts), as it relies on the reflectance of the surface instead of its texture. The application of direct photometric methods to pose estimation of arbitrary 3D objects has so far not been investigated. Thus, in the following a simple and straightforward approach to direct 6DOF object pose tracking will be introduced and it will be shown how it can be combined with the region-based method of the previous chapter.

5.2 Application to Object Pose Tracking

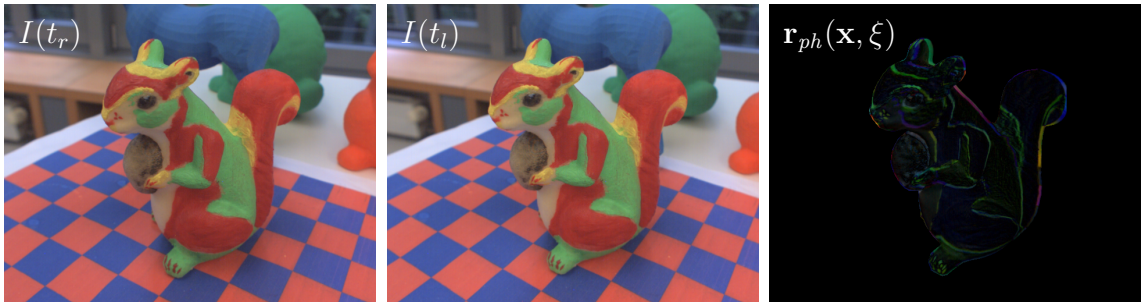
In case of direct photometric object pose tracking each object is again represented by a dense 3D triangle mesh, as previously within the region-based approach. Since here this 3D model is the only part of the scene of which the geometry is known, the warp function can only be computed within Ω_f i.e. for all pixels where $I_d(\mathbf{x}) > 0$. As before, tracking starts with a known pose of the object $T_{cm}(t_0)$ corresponding to the first frame of a sequence (e.g. computed by a detection step). This pose and the corresponding image hence define the initial reference frame as $I(t_r) = I(t_0)$ with

$T_{cm}(t_r) = T_{cm}(t_0)$. This further implicitly yields a pixel-wise relationship between the image intensities and the surface geometry of the 3D model from that perspective. This reference frame can then be used in order to estimate the pose in the next frame via direct photometric optimization. For continuous pose tracking the obvious strategy would therefore be to always update the reference frame with the current live frame as $I(_r) = I(t_l)$ and $T_{cm}(t_r) = T_{cm}(t_l)$ after successful pose optimization. In practice however, the tracked pose will never be perfectly estimated due to e.g. image discretization or an already slightly inaccurate initial pose detection. Since here no explicit correspondences between the image data and the object's surface are given, over time irreversible pose drift will accumulate. This causes the alignment of $I(t_r)$ and $T_{cm}(t_r)$ to deteriorate more and more with every new frame. Unfortunately this drift is inherent to this straightforward approach. However, in scenarios where the object pose is only to be estimated from one side (i.e. a small range of outer plane rotation), the reference frame would not have to get updated at all after a suitable initialization, assuming that the lighting conditions remain stable.

Experiments have shown that the drift can be significantly reduced by updating the reference frame only when the difference between the current live pose $T_{cm}(t_l)$ and the reference pose $T_{cm}(t_r)$ is above a certain boundary. It is thus sort of an intermediate solution between updating the reference with every frame and not updating it at all. Therefore, in the proposed implementation a reference frame is only updated if the difference between $T_{cm}(t_l)$ and $T_{cm}(t_r)$ is greater than 5° for rotation or 5 cm for translation after optimization, i.e. the same criteria as used for tracking loss detection in Section 4.8.2.

5.2.1 Photometric Pose Optimization

Since the photometric cost function (5.3) is in the classical form of a nonlinear least-squares problem, it can be optimized with a straightforward Gauß-Newton method. For this, all camera frames are assumed to be given as RGB color images with normalized intensities $I(t_k) : \Omega \to [0, 1]^3 \subset \mathbb{R}^3$. This normalization was chosen with respect to the combined cost that will be introduced in Section 5.3.1, such that the two individual errors are of similar numerical scale.



(a) (b) (c)

Figure 5.2: A visualization of the RGB color residuum minimized during direct photometric pose optimization. Here shown are the reference frame $I(t_r)$ (a), the live frame $I(t_l)$ (b) and the corresponding per pixel RGB color residuum $\mathbf{r}_{ph}(\mathbf{x}, \xi)$ within the object region (c).

For deriving the Gauß-Newton step, the per pixel RGB color residuum is denoted by

$$\mathbf{r}_{ph}(\mathbf{x}, \xi) = I(\mathbf{x}_w(\mathbf{x}, \xi), t_l) - I(\mathbf{x}, t_r) \in \mathbb{R}^3,$$

as visualized in Figure 5.2. This allows to rewrite (5.3) compactly as

$$E_{ph}(\xi) = \frac{1}{2} \sum_{\mathbf{x} \in \Omega_f} \|\mathbf{r}_{ph}(\mathbf{x}, \xi)\|_2^2.$$

Accordingly, the gradient of $E_{ph}(\xi)$ is given by

$$\nabla E_{ph}(\xi) = \sum_{\mathbf{x} \in \Omega} \left(\mathbf{r}_{ph}(\mathbf{x}, \xi)^\top \frac{\partial \mathbf{r}_{ph}(\mathbf{x}, \xi)}{\partial \xi} \right)^\top, \tag{5.4}$$

and the approximated Hessian by

$$\nabla^2 E_{ph}(\xi) \approx \sum_{\mathbf{x} \in \Omega} \left(\frac{\partial \mathbf{r}_{ph}(\mathbf{x}, \xi)}{\partial \xi} \right)^\top \frac{\partial \mathbf{r}_{ph}(\mathbf{x}, \xi)}{\partial \xi}, \tag{5.5}$$

when again assuming that $\mathbf{r}_{ph}(\xi, \mathbf{x})$ is linear in ξ. By denoting the 3×6 Jacobian of the residuum $\mathbf{r}_{ph}(\mathbf{x}, \xi)$ at the current pose i.e. $\xi_0 = \mathbf{0}$ as

$$J_{ph}(\mathbf{x}, \xi_0) = \left. \frac{\partial \mathbf{r}_{ph}(\mathbf{x}, \xi)}{\partial \xi} \right|_{\xi_0} \in \mathbb{R}^{3 \times 6}, \tag{5.6}$$

the update step is computed by

$$\Delta\xi_t = -\left(\sum_{\mathbf{x}\in\Omega} J_{ph}(\mathbf{x},\xi_0)^\top J_{ph}(\mathbf{x},\xi_0)\right)^{-1} \sum_{\mathbf{x}\in\Omega} J_{ph}(\mathbf{x},\xi_0)^\top \mathbf{r}_{ph}(\mathbf{x},\xi_0). \tag{5.7}$$

As before, during successful tracking the optimization is initialized with $T_{cm}^0 = T_{cm}(t_{l-1})$. Furthermore, as previously explained for the region-based method in Section 4.6.1, it is performed using a hierarchical coarse-to-fine strategy. As before, this is done in order to improve the convergence rate of the iterative optimization and thereby increase the pose tracking's robustness towards image noise, fast movement, defocus and motion blur.

5.2.2 Hierarchical Target-to-Source Tracking

In this work, the iterative photometric optimization is computed within the same three-level coarse-to-fine image pyramid as explained before in Section 4.6.1. Here, also the same number of iterations per level is performed (four on the 3rd, two on the 2nd and one on the 1st level).

The per pixel Jacobian (5.6) is computed as

$$J_{ph}(\mathbf{x},\xi_0) = \left.\frac{\partial I(\mathbf{x}_w(\mathbf{x},\xi),t_l) - I(\mathbf{x},t_r)}{\partial\xi}\right|_{\xi_0} = \left.\frac{\partial I(\mathbf{x}_w,t_l)}{\partial\mathbf{x}_w}\frac{\partial\mathbf{x}_w(\mathbf{x},\xi)}{\partial\xi}\right|_{\xi_0} \in \mathbb{R}^{3\times 6},$$

where the derivatives of $I(\mathbf{x}_w,t_l)$ with respect to a pixel $\mathbf{x}_w$ are computed as the transposed 2D RGB image gradient

$$\begin{aligned}\frac{\partial I(\mathbf{x}_w,t_l)}{\partial\mathbf{x}_w} &= \left[\tfrac{\partial I(\mathbf{x}_w,t_l)}{\partial x_w}, \tfrac{\partial I(\mathbf{x}_w,t_l)}{\partial y_w}\right] = \begin{bmatrix}\nabla_x I(t_l)\\ \nabla_y I(t_l)\end{bmatrix}^\top \\ &= \left[\tfrac{I(x+1,y,t_l)-I(x-1,y,t_l)}{2}, \tfrac{I(x,y+1,t_l)-I(x,y-1,t_l)}{2}\right] \in \mathbb{R}^{3\times 2},\end{aligned}$$

calculated from central differences similar to (4.17). As previously, with regard to (2.40) the derivatives of the warped pixel coordinates $\mathbf{x}_w$ with respect to ξ_0 are

calculated as

$$\left.\frac{\partial \mathbf{x}_w(\mathbf{x},\xi)}{\partial \xi}\right|_{\xi_0} = \begin{bmatrix} \frac{f_x}{Z_c} & 0 & -\frac{X_c f_x}{Z_c^2} \\ 0 & \frac{f_y}{Z_c} & -\frac{Y_c f_y}{Z_c^2} \end{bmatrix} \begin{bmatrix} 0 & Z_c & -Y_c & 1 & 0 & 0 \\ -Z_c & 0 & X_c & 0 & 1 & 0 \\ Y_c & -X_c & 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 6}, \tag{5.8}$$

where $\mathbf{X}_c = [X_c, Y_c, Z_c]^\top = (T_{cm}^t T_{mc}(t_r)\tilde{\mathbf{X}}_c(I_d(t_r), \mathbf{x}))_{3\times 1}$, according to (5.2). Projecting $\mathbf{X}_c$ into the current live image then gives the corresponding color value needed to compute the residuum for the respective $\mathbf{x}$, since $I(\pi(K\mathbf{X}_c), t_l) = I(\mathbf{x}_w(\mathbf{x}, \xi_0), t_l)$. Here, in practice bi-linear interpolation should be used to compute the color corresponding to the warped sub-pixel coordinates, in order to increase the accuracy of the pose optimization.

The strategy as described above for obtaining the color in the live frame and the corresponding coordinates of X_c, can be considered *source-to-target* mapping. Here, based on the reference depth map $I_d(t_r)$ the pixels are mapped from the reference frame I_{t_r} (the source) to the live frame (the target). This mapping can however also be done in the opposite direction, i.e. *target-to-source*. This approach is used within the proposed implementation with regard to the combined optimization in Section 5.3.2. It can be achieved by rendering a new (warped) depth map I_d corresponding to the current pose estimate T_{cm}^t, in each iteration (as in Section 4.5.2). Given this, for all pixels considered as warped where $I_d(\mathbf{x}_w(\mathbf{x}, \xi_0)) > 0$ the corresponding $\mathbf{X}_c$ can be computed via back-projection as $\mathbf{X}_c(I_d, \mathbf{x}_w(\mathbf{x}, \xi))$ according to (2.33). This surface point is then used in order to obtain the corresponding pixel $\mathbf{x}$ in the reference frame by essentially applying the inverse warp function such that $\mathbf{x} = \pi\left(K(T_{cm}(t_r)T_{mc}^t\tilde{\mathbf{X}}_c)_{3\times 1}\right)$. Eventually, the color in the reference frame is given by $I(\mathbf{x}, t_r)$, which is also computed using bi-linear interpolation.

5.3 Combining Region-based and Direct Photometric Pose Tracking

Direct photometric and region-based pose estimation approaches are to some degree complementary. Firstly, the statistical image segmentation models used with

region-based pose estimation are most suitable for homogeneous objects and in general more likely to fail for very heterogenous or textured objects. As opposed to this, direct photometric approaches work best for strongly textured objects and are least suitable for homogeneous, textureless objects. Secondly, region-based methods are essentially only using the image information in a narrow band around the contour of the object without regarding most of the pixels within the object region. In contrast, direct photometric methods only use the pixels within the object region, without explicitly enforcing any constraint on the object's boundary (i.e. the contours of silhouettes to match). Thirdly, due to this difference in utilized image information, region-based methods suffer from pose ambiguities which are unlikely to occur for direct photometric approaches (assuming the objects are sufficiently textured). Correspondingly, direct photometric methods suffer from pose drift, which is not likely to occur for region-based methods (assuming the object can be segmented from the background).

It is therefore reasonable to fuse these two approaches, such that they naturally compensate each others weaknesses and combine their advantages. In the following, a common cost function for pose estimation will thus be introduced that comprises both a region-based term as well as a photometric term.

5.3.1 An extended common cost function

In this work, a straightforward combination of region-based tracking with direct photometric pose tracking is proposed. For this, a common cost function is introduced, that simply extends the region-based cost (4.10) (in weighted least-squares form) by the photometric cost (5.3) in form of

$$E_{rbph}(\xi) \doteq E_{rb}(\xi) + \lambda E_{ph}(\xi) = \frac{1}{2}\sum_{\mathbf{x}\in\Omega} \psi(\mathbf{x}) r_{rb}(\mathbf{x},\xi)^2 + \lambda \frac{1}{2}\sum_{\mathbf{x}\in\Omega_f} \|\mathbf{r}_{ph}(\mathbf{x},\xi)\|_2^2, \quad (5.9)$$

where $\lambda \in [0,1]$ serves as a weighting parameter. It thereby determines the influence of the photometric term on the pose optimization. In the following experiments it is set to $\lambda = 0.8$, which gives the best results on average according to an extensive empirical investigation.

5.3.2 Joint Gauß-Newton Optimization

Both individual cost functions can be optimized efficiently using a Gauß-Newton strategy. Thus, also for the combined cost (5.9) a joint iterative Gauß-Newton scheme is proposed. Since the terms are combined in form of a sum, the derivation is straightforward. For the overall gradient one obtains

$$\nabla E_{rbph}(\xi) = \nabla E_{rb}(\xi) + \lambda \nabla E_{ph}(\xi),$$

which is computed according to (2.43) and (5.4). In analogy to this, the overall approximated Hessian is given by

$$\nabla^2 E_{rbph}(\xi) = \nabla^2 E_{rb}(\xi) + \lambda \nabla^2 E_{ph}(\xi),$$

which is computed according to (2.45) and (5.5). The common update step is then finally computed as

$$\Delta\xi_t = -\left(\nabla^2 E_{rb}(\xi_0) + \lambda \nabla^2 E_{ph}(\xi_0)\right)^{-1} \left(\nabla E_{rb}(\xi_0) + \lambda \nabla E_{ph}(\xi_0)\right),$$

in each iteration at $\xi_0 = \mathbf{0}$.

As for the individual cost functions, in the proposed implementation, this joint optimization is again performed coarse-to-fine within the familiar three-level image pyramid (see Section 4.6.1). By default, here also the same number of iterations is used, meaning four on the 3rd, two on the 2nd and one on the first level. As explained previously, the photometric term is optimized using a target-to-source strategy. This comes with the advantage that the coordinates of $\mathbf{X}_c$, which are needed within the derivatives of the projected image pixels (4.18) and (5.8), can be computed from the same depth map I_d (updated in each iteration) for both terms.

5.4 Evaluation

Here, the proposed combined approach (5.9) is evaluated in comparison to pure direct photometric (5.3) and pure region-based (4.10) pose tracking. For this, the

Variant	Cost Func.	Ape	Baking Soda	Bench Vise	Broccoli Soup	Camera	Can	Cat	Clown	Cube	Driller	Duck	Egg Box	Glue	Iron	Koala Candy	Lamp	Phone	Squirrel
Regular	E_{ph} (5.3)	44.8	33.5	48.5	35.5	59.3	55.4	46.1	38.9	18.3	58.3	54.2	54.5	44.5	57.4	28.2	53.5	57.1	60.6
	E_{rb} (4.10)	**85.0**	39.0	**98.9**	82.4	79.7	87.6	**95.9**	93.3	**78.1**	93.0	86.8	74.6	38.9	81.0	46.8	97.5	80.7	99.4
	E_{rbph} (5.9)	84.7	**51.7**	98.0	**85.8**	**85.5**	**88.8**	95.4	**93.6**	43.9	**94.3**	**87.8**	**85.4**	**50.3**	**86.1**	**58.3**	**97.5**	**86.9**	**99.5**
Dynamic Light	E_{ph} (5.3)	36.7	27.1	46.4	31.4	56.0	47.8	44.6	37.6	19.2	54.8	41.6	46.0	33.1	51.2	27.2	47.1	53.8	58.0
	E_{rb} (4.10)	84.9	42.0	**99.0**	81.3	84.3	88.9	**95.6**	**92.5**	**77.5**	94.6	86.4	77.3	52.9	77.9	47.9	**96.9**	81.7	99.3
	E_{rbph} (5.9)	**85.9**	**49.5**	98.5	**85.7**	**89.3**	**88.9**	94.8	91.7	49.2	**95.6**	**87.0**	**86.5**	**66.1**	**83.2**	**59.1**	96.4	**86.7**	**99.3**
Noisy + Dynamic Light	E_{ph} (5.3)	27.2	19.7	42.5	28.9	47.9	44.9	41.3	36.5	18.4	45.2	34.9	41.8	**27.7**	45.0	25.9	43.5	44.4	52.6
	E_{rb} (4.10)	77.5	44.5	91.5	82.9	51.7	38.4	**95.1**	69.2	**24.4**	64.3	**88.5**	11.2	2.9	46.7	32.7	57.3	44.1	**96.6**
	E_{rbph} (5.9)	**78.4**	**45.2**	**91.9**	**84.6**	**60.2**	**52.3**	93.8	**75.0**	22.2	**68.4**	87.9	**35.8**	10.4	**47.1**	**52.1**	**64.7**	**50.9**	96.5
Unmodelled Occlusion	E_{ph} (5.3)	32.4	24.3	43.7	28.4	49.8	42.3	41.0	35.0	18.1	48.1	37.1	40.5	29.8	45.9	25.5	44.2	47.1	52.5
	E_{rb} (4.10)	80.0	42.7	**91.8**	73.5	76.1	**81.7**	**89.8**	82.6	**68.7**	86.7	80.5	67.0	46.6	64.0	43.6	**88.8**	68.6	86.2
	E_{rbph} (5.9)	**80.6**	**46.5**	91.7	**77.2**	**80.0**	81.3	89.5	**82.8**	47.3	**88.4**	**80.7**	**76.1**	**58.3**	**71.6**	**52.7**	88.4	**74.0**	**88.3**
Avg. Runtime	E_{ph} (5.3)	7.9	7.7	9.5	7.8	8.6	8.7	8.0	7.7	7.9	8.6	7.8	8.3	7.8	9.3	7.8	9.4	8.4	8.6
	E_{rb} (4.10)	15.5	15.7	20.3	16.6	17.4	18.9	16.9	15.9	16.8	19.4	15.7	16.8	16.1	19.1	16.5	21.8	17.6	19.1
	E_{rbph} (5.9)	16.3	16.7	24.2	17.9	19.3	21.5	17.9	17.2	18.0	21.3	17.2	18.5	16.8	20.9	17.8	20.3	19.6	20.5

Table 5.1: Tracking success rates (in %) in the RBOT dataset for using the combined cost function (5.9) in comparison to the pure photometric (5.3) and the pure region-based cost (4.10). Furthermore, in the last three rows the average runtime of the different approaches are denoted (in ms) measured across the first three variants.

same experiment using the same hardware as in Section 4.8.2 was conducted within the RBOT dataset. Here also the same 5° and 5 cm criterion was used for detecting a tracking loss. All three cost functions were optimized with an equal number of iterations per pyramid level in order to provide a fair comparison. Note, that the *modelled occlusion* variant was not included in the experiments, since the implementation of the direct photometric approach was not extended to multiple objects at this point.

The results of this experiment are presented in Table 5.1. They show that in most cases the combined method outperforms the region-based approach although the pure photometric method takes the last place in almost all scenarios. For the well-textured bodies of revolution (e.g. Baking Soda and Koala Candy), which the region-based methods generally struggle with (see Section 4.9) this improvement is even above 10% in the regular variant. It can also be seen that for many objects (e.g. Camera, Clown, Can, Cube, Egg Box, Iron, Lamp and Phone) the photometric tracking cost function is significantly more robust to image noise than the region-based term. Lastly, these results show that optimizing the combined cost only increases the average runtime by about 1–2 ms in almost all sequences compared

Variant	Cost Func.	Ape	Baking Soda	Bench Vise	Broccoli Soup	Camera	Can	Cat	Clown	Cube	Driller	Duck	Egg Box	Glue	Iron	Koala Candy	Lamp	Phone	Squirrel
Regular	E_{ph} (5.3)	59.6	56.4	68.4	53.1	77.2	68.3	63.0	57.4	28.6	72.9	67.6	69.4	**57.8**	74.1	42.7	66.1	71.3	77.3
	E_{rb} (4.10)	91.9	37.8	99.7	83.8	80.8	91.9	99.6	97.7	**82.0**	94.2	90.2	78.8	40.2	85.4	48.6	98.1	86.0	99.9
	E_{rbph} (5.9)	**92.9**	**69.7**	**99.9**	**93.7**	**90.1**	**95.2**	**99.6**	**99.3**	76.4	**95.6**	**93.3**	**91.5**	56.1	**92.0**	**80.7**	**98.7**	**93.0**	**100.0**
Dynamic Light	E_{ph} (5.3)	47.8	45.6	62.7	47.6	71.1	57.7	56.8	53.2	27.6	67.9	48.5	58.0	42.6	63.8	44.0	57.0	65.8	71.4
	E_{rb} (4.10)	91.5	41.5	**99.8**	84.3	86.6	92.6	99.5	96.0	80.9	95.2	89.2	82.3	57.7	83.5	50.8	97.5	86.4	99.8
	E_{rbph} (5.9)	**94.0**	**60.8**	99.6	**92.1**	**92.2**	**94.6**	**99.6**	**98.0**	**82.3**	**96.7**	**91.8**	**91.9**	**73.0**	**86.4**	**83.2**	**98.7**	**91.8**	**100.0**
Noisy + Dynamic Light	E_{ph} (5.3)	40.2	33.4	59.1	43.9	**66.2**	**56.3**	53.8	52.2	27.0	61.0	45.4	**55.1**	**39.1**	**59.8**	42.4	54.4	**60.5**	68.8
	E_{rb} (4.10)	84.1	41.0	91.4	86.4	52.0	27.8	98.7	73.3	22.8	63.0	92.9	6.3	7.9	37.3	34.8	56.1	47.7	97.9
	E_{rbph} (5.9)	**85.1**	**56.4**	**95.2**	**90.8**	62.5	54.8	**99.1**	**84.3**	**40.5**	**68.5**	**93.5**	35.9	4.8	47.3	**70.5**	**69.2**	52.7	**98.1**
Unmodelled Occlusion	E_{ph} (5.3)	43.1	41.5	60.0	43.2	64.5	52.4	52.5	49.7	27.5	60.0	43.8	51.3	38.9	56.8	40.7	54.5	59.8	66.5
	E_{rb} (4.10)	86.6	40.0	93.4	76.5	76.6	84.9	94.2	84.7	72.7	88.5	84.5	73.4	49.9	67.8	45.7	89.5	71.8	88.2
	E_{rbph} (5.9)	**88.5**	**57.6**	**94.1**	**83.0**	**83.1**	**88.7**	**95.0**	**88.9**	**76.0**	**90.2**	**86.0**	**81.7**	**65.8**	**76.5**	**77.3**	**90.9**	**75.4**	**90.8**
Avg. Runtime	E_{ph} (5.3)	12.8	13.1	15.2	13.3	13.9	14.2	13.0	13.2	13.7	14.2	12.9	13.7	12.9	15.0	13.2	15.1	13.9	14.4
	E_{rb} (4.10)	26.9	27.3	34.0	28.8	29.6	31.8	28.9	27.7	28.4	31.6	26.8	28.4	27.4	31.5	27.7	33.6	30.2	31.0
	E_{rbph} (5.9)	27.9	29.1	37.5	30.6	32.2	34.8	30.4	29.2	30.4	34.8	28.5	31.2	29.1	35.5	30.5	37.9	32.9	35.2

Table 5.2: The same tracking success rate (in %) and runtime (in ms) result comparison as in Table 5.1. For this experiment however, all three methods have been used with twice the number of optimization iterations per pyramid level.

to the region-based method. Given the improved robustness, this slight temporal increment is neglectable and demonstrates the effectiveness of the proposed target-to-source implementation.

In a second experiment the influence of the number of optimization iterations on the tracking performance was investigated. For this, all three methods were evaluated in the RBOT dataset again but this time with twice as many iterations per pyramid level as before. This means that now eight iterations on the 3rd level, four on the second level and two on the first level (the full image resolution) were computed.

The corresponding results of this second experiment are shown in Table 5.2. It can be seen that the increased number of iterations generally leads to an improvement for all three cost functions. Here, the success rates of the photometric method increased the most while the region-based approach benefited least from it. Especially in case of image noise, the pure direct photometric method has gained by far the most and now wins for six out of the eighteen objects (Camera, Can, Egg Box, Glue, Iron and Phone). This significant boost of the pure photometric method also has a positive impact on the combined solution. Compared to the first experiment, its improvement over the region-based approach has increased even more. For some

objects the success rates are now up to 20% – 30% higher (e.g. Baking Soda, Glue, Koala Candy). In case of occlusions the combined approach surpassed the pure region-based method for all objects and only loses once to it in the regular, dynamic light and noisy variant. The combined solution is further the only to successfully track the pose in 100% of the images in two sequences (the regular and dynamic light variant of the Squirrel). In analogy to the first experiment, the runtime only increases by about 2–4 ms. Thus, the average overall time per frame of the combined approach for a single object remains below 40 ms, meaning that is it still real-time capable (i.e. runs with more than 25 frames per second). This demonstrates the advantages of the proposed combined region-based and direct photometric solution for 6DOF object pose tracking over the previous state of the art.

5.5 Summary

In this chapter, the general concept of direct image alignment and a corresponding cost function were introduced. It was then shown how it can be applied to the problem of 6DOF rigid object pose estimation and how the corresponding photometric cost function can be optimized iteratively using a Gauß-Newton strategy.

Based on this, a novel cost function was presented that was developed within the course of this dissertation. It combines the region-based with the photometric term, in order to improve the pose tracking robustness in scenarios where at least one of the two separate approaches is prone to fail. It was further shown, how this combined cost can be optimized using a joint Gauß-Newton optimization. This was enabled by the novel weighted least-squares formulation of the region-based term, derived in Chapter 4.

The quantitative experimental comparison at the end of this chapter demonstrated that this combination does in fact lead to a significant improvement over the previously introduced state of the art region-based approach. However, the extension of the combined solution to multiple objects and pose detection were not addressed and remain future work.

6 Conclusion

This final chapter starts by summarizing the overall achievements developed in this dissertation (Section 6.1). It then goes on by showing a selection of actual and potential practical application scenarios of the presented systems (Section 6.2). The chapter concludes with an overview of open topics and remaining future work (Section 6.3).

6.1 Summary of Thesis Achievements

Within the course of this dissertation, essentially two main real-time monocular pose estimation systems have been developed. Firstly, the infrared LED marker-based approach of Section 3.2, and secondly, the markerless combined region-based and direct photometric method derived in Chapter 4 and Chapter 5. Both these proposed solutions are currently state of the art in many aspects.

The key innovations and advantages of the active marker approach can be summarized as follows:

- Accurate monocular 6DOF pose estimation of a single marker within $\sim$1 ms even at 2 megapixels image resolution on a single laptop CPU core.

- No frame-to-frame information is required or used (pure tracking by detection).
- A novel and minimal strategy to robustly avoid pose ambiguities even at large distances to the camera.
- Two novel nearly co-planar dot patterns (cross-shaped and circular) that allow to identify each LED from a single image.
- Simultaneous tracking by detection of multiple markers.
- Single image marker identification based on their cross-ratio ID.
- A novel custom 3D reconstruction algorithm to semi-automatically calibrate the markers with a single camera.

Accordingly, the proposed passive markerless approach comprises the following contributions and enhancements:

- The first combined region-based and direct photometric cost function for monocular 6DOF object pose estimation.
- A novel nonlinear weighted least-squares formulation of the region-based term enabling an iteratively re-weighted Gauß-Newton scheme and thereby a joint optimization of the combined cost.
- A novel statistical object segmentation model based on the proposed temporally consistent local color histograms.
- The first region-based pose detection approach for a level-set method enabled by the tclc-histograms.
- State of the art robustness to cluttered backgrounds, partial occlusions, dynamic lighting changes, direct sunlight and complex motion for textureless and weakly textured objects.
- The pose of a single object can be tracked within $\sim$16 ms and detected within $\sim$150 ms on a commodity laptop.

- It is thus the first monocular pose estimation system capable of region-based multi-object tracking in real-time.

In addition to the proposed algorithms, a novel comprehensive dataset for monocular object pose tracking has been created and made available to the community (see Section 4.8.1). It thereby closes a gap in previous literature on datasets dedicated to the task of monocular 6DOF object pose estimation due to its increased coverage, complexity and diversity.

6.2 Applications

While the potential fields of application for object pose estimation have been outlined in the introduction of this work, the systems developed here have so far been applied in a variety of specific practical scenarios. An exemplary selection of these will be presented in the following.

Industrial The proposed active marker approach (with the cross-shaped marker) has a patent pending and has been licensed by the soft2tec GmbH[1]. They have built several commercial products around it and have integrated it in existing solutions where in many cases it replaced a previously utilized ultra sonic pose estimation system. HSRM-Tracking is for example the backbone of the so-called *nexonar assembly scout*[2]. Here it is used to estimate the pose of markers attached to tools, wristbands and gloves (see Figure 6.1) to supervise and support manual industrial assembly processes in real-time.

Their products including HSRM-Tracking have ever since been sold to numerous different companies all across Europe, Asia and the USA. These companies include, but are not limited to, major automotive corporations and research institutions. So far more than a hundred systems licensing HSRM-Tracking have been sold.

[1] Website of the company soft2tec GmbH: http://www.soft2tec.com/
[2] Website of the nexonar product series: http://www.nexonar.com/

(a) (b)

Figure 6.1: Example images from an official nexonar assembly scout advertising brochure[3]. These images show different cross-shaped HSRM-Tracking markers attached to a screwdriver (a) as well as a wristband (b). Here the markers are integrated in custom cases that include a battery and a USB power charging interface, designed by soft2tec GmbH.

In addition to this, HSRM-Tracking was used within an internal research project that aimed at measuring the movement range of a car's engine block. For automotive companies it is of great interest to know how much the individual parts within the engine compartment move while the car drives. Being able to precisely acquire this information allows to optimize the arrangement of the components and thereby use all available space or decrease the overall size.

The leading existing solution for measuring the movement of an engine block is the *EngineWatch* system[4] by AICON 3D Systems GmbH. It is based on an optical monocular method but requires the hood to be removed and a rather large construction for the camera to be mounted onto the front of the car (see Figure 6.2). This however influences the aerodynamics and thus the behavior of the car and also restricts it from driving at full speed during experiments.

Using HSRM-Tracking however, comparable measurements could be obtained without externally changing the shape of the car. In a prototypical experimental setup a consumer *GoPro* camera[5] equipped with an infrared filter was duct-taped to the inside of the engine compartment and a small marker was strapped to the engine

[3]Thanks to soft2tec for kindly providing these images which remain entirely their property.
[4]The EngineWatch system: https://www.aicon3d.com/products/vehicle-testing/enginewatch
[5]GoPro consumer cameras: http://www.gopro.com/

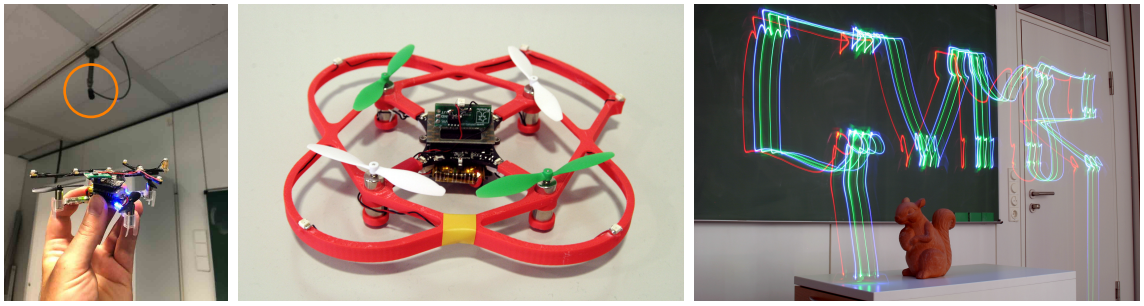
(a) (b)

Figure 6.2: A photo of the experimental setup inside an engine compartment, using HSRM-Tracking for measuring the movement of a car's engine block (a) in comparison to the EngineWatch system[6] of the AICON 3D Systems GmbH (b).

block (see again Figure 6.2). Due to these minimal space requirements of the hardware setup, the hood could remain closed as usual and the car could drive at any desired speed without any noticeable change in its behavior. The video recorded by the camera was then evaluated using HSRM-Tracking in a post-processing step.

Robotics Within the Computer Vision and Mixed Reality (*CVMR*) group[7] at the RheinMain University of Applied Sciences, HSRM-Tracking was used for numerous robotics related research projects. It is for example one of the main components of *ICARUS*, a low-cost infrastructure for supervising and controlling autonomous *micro aerial vehicles* (MAVs) [LTB+17]. Here, HSRM-Tracking has mainly been used to determine the exact location and orientation of the MAVs (e.g. quadcopters) in order to enable conducting research on autonomous control algorithms. The robustness and precision of such control approaches directly depends on the accuracy, frequency and temporal delay of the pose measurements, which therefore makes HSRM-Tracking a perfect fit. In a minimal setup one only needs to attach a single marker to the MAV and place a single camera connected to a laptop computer somewhere in the room for the pose estimation part (see Figure 6.3). This, among the other parts of ICARUS, makes the infrastructure very low cost compared to those used in other related laboratories [MMLK10, LHM+14] that typically include

[6]Thanks to AICON 3D for kindly providing this image which remains entirely their property.
[7]Website of the Computer Vision and Mixed Reality Group: http://cvmr.mi.hs-rm.de/



(a) (b) (c)

Figure 6.3: With HSRM-Tracking the pose of a miniature, light-weight MAV can be estimated robustly, requiring only a single camera (e.g. mounted to the ceiling – here tagged by an orange circle) and a marker attached to it (a). While for the first prototypes the cross-shaped marker was mounted onto the MAV, later versions used the circular pattern which could be integrated more natively in form of a rotor protecting frame (b). Using the ICARUS infrastructure, such a modified MAV can then follow pre-defined trajectories autonomously within the camera's field of view. The results can for example be visualized with help of long-exposure photography in form of a so-called *light-painting* (c).

expensive multi-camera motion capturing systems. The light-weight markers of HSRM-Tracking further allow to utilize very small hobby copters in contrast to the otherwise commonly used larger, heavier and more expensive devices. Another advantage of this is, that these small MAVs do not cause any harm or injuries to a human even when a body part touches a rotor. This enables ICARUS for example to be demonstrated directly in front of uninstructed people (e.g. in a class room or at an exhibition) without requiring safety nets or cages.

For most of the experiments within ICARUS the camera was mounted to the ceiling, such that one or multiple MAVs can perform autonomous maneuvers in its field of view. In addition to this, an experimental *search and rescue* (SAR) scenario was explored in which the MAV collaborates with a mobile ground robot that serves as a landing platform. Here, a light-weight quadcopter, equipped with an onboard RGB camera, acts as a so-called *flying periscope*. In this scenario, the live video stream of the copter's camera can for example be used to reconstruct a 3D map of the environment by using a monocular SLAM algorithm, such as *ORB-SLAM* [MMT15]. This in return enables the otherwise *blind* so-called *unmanned ground vehicle* (UGV)

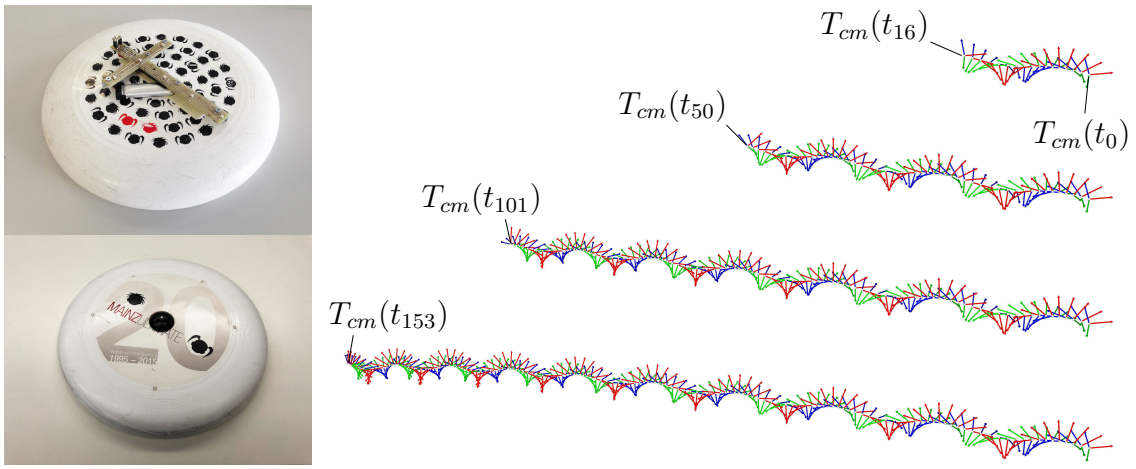
(a) (b) (c)

Figure 6.4: For the *flying periscope* experiments a miniature MAV with an onboard RGB camera (tagged by an orange circle), was equipped with a marker on its bottom (a). Correspondingly, the UGV carries an upward-facing, monochrome camera equipped with an infrared filter (tagged by another orange circle) connected to a Raspberry Pi 3 computer as well as a transparent landing platform for the MAV (b). This enables onboard-controlled maneuvers of the MAV (tagged by a green circle) above the UGV (c).

to navigate autonomously and see beyond obstacles.

In this scenario, the camera is placed on top of the UGV looking upwards (see Figure 6.4). Accordingly, a marker was attached to the bottom of the MAV such that it can be seen by the moving ground vehicle's camera. The UGV has further been equipped with a so-called *Raspberry Pi 3* single-board computer[8], that runs HSRM-Tracking while simultaneously computing control commands based on the pose measurements and sending them to the MAV. Due to its low runtime enabled by the overall simplicity of the approach and its optimized implementation, HSRM-Tracking still runs in real-time on a single core of the relatively low-powered Raspberry Pi. Even when using the full 1080p image resolution of the natively supported Raspberry Pi camera, the pose of the MAV can be estimated at sufficient speed (∼30 fps) and accuracy. This allows to run the entire closed control loop onboard on the UGV to perform autonomous landing and takeoff maneuvers and

[8]Raspberry Pi single-board computers: https://www.raspberrypi.org/



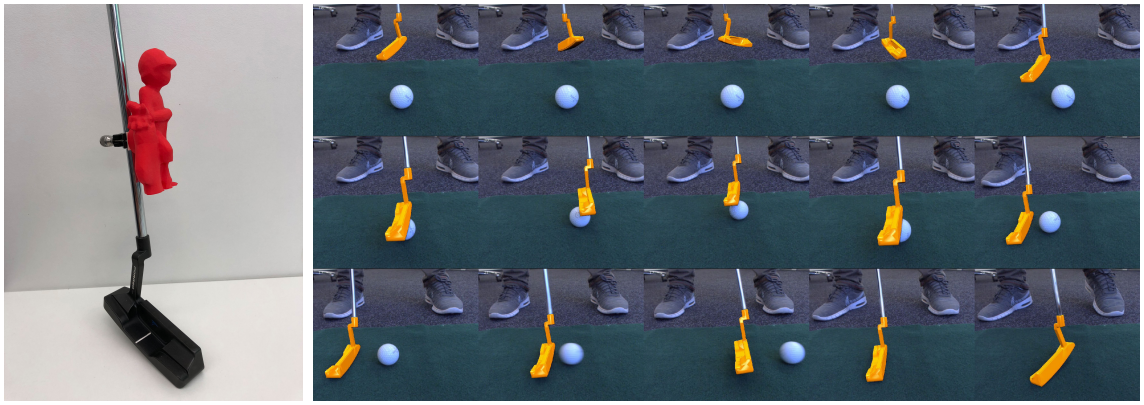
(a) (b)

Figure 6.5: Live 6DOF reconstruction of frisbee throws with HSRM-Tracking. For a first prototype a cross-shaped marker was simply glued onto the disc (top), which was later improved by almost seamlessly integrating the circular LED pattern (bottom) into it (a). Here shown is a visualization of the progressing reconstruction of an example throw (b). This corresponding sequence was captured within 0.955 seconds and contains 154 frames. The reconstructed trajectory started at $\mathbf{t}_{cm}(t_0) = [2577.4, 60.3, 4564.8]^\top$ and ended at $\mathbf{t}_{cm}(t_{153}) = [2896.8, 660.8, 7093.4]^\top$ (in millimeters) including nine full turns. Accordingly, the frisbee traveled a total distance of 6.073 meters at an average speed of approximately 23 km/h.

ensure that the MAV stays within its field of view in the meantime, even while the platform itself is steadily moving. For further details and more related application examples please refer our corresponding publication [LTB+17].

Sports In order to create a hard experimental scenario for evaluating HSRM-tracking, an LED marker was attached to a frisbee. While initially a cross-shaped prototype marker was simply glued on top of it, later the invention of the circular pattern allowed to better integrate the marker without significantly changing its aerodynamics (see Figure 6.5). This allows to compute a 6DOF reconstruction of the frisbee's trajectory in real-time in a large volume with high spatial accuracy and temporal resolution beyond any other currently available monocular solution.

In a practical application, this kind of feedback could potentially be used to sup-



(a) (b)

Figure 6.6: Using the proposed passive method, the trajectory of a golf club can either be estimated markerlessly, based on a 3D CAD model of it, or with help of a light-weight, passive 3D marker (here in form of a printed, red golf player figure) attached to its shaft (a). Here also shown are example frames from an image sequence used to estimate the golf club's pose markerlessly, while interacting with a ball (b). The results are visualized by overlaying the input images with virtual renderings of the corresponding 3D CAD model based on the estimated poses.

port athletes in order improve on their throwing technique and monitor training progression. At several exhibitions and similar events the system was used as an interactive live demonstration. Here, visitors can throw a correspondingly modified frisbee across the field of view of the camera and get their results evaluated live in form of average speed and spin (degrees per second). During these occasions, numerous throws with velocities up to 80 km/h were successfully captured.

The proposed markerless pose estimation approach was in parts developed within the course of a research project, that aimed to develop a novel system for reconstructing, analyzing and evaluating the 6DOF trajectories of golf clubs during the swing. The resulting system should also potentially be used to support the training of professional athletes and amateurs. Here, the main requirement was that the golf club should have to get modified as little as possible or ideally not at all, such that the athlete does not notice any difference when swinging. The solution should also be portable (i.e. require a minimum amount of hardware) and work robustly both indoors and outdoors (e.g. on a golf course) at all weather conditions. Hence, the

Figure 6.7: A few examples of the proposed passive method estimating the pose of a single or multiple complex objects under different challenging conditions. These include cluttered scenes/backgrounds, strong occlusions as well as direct sunlight. Top: The raw RGB input frames. Bottom: A mixed reality visualization of the tracking result where the input images are virtually augmented with renderings of the corresponding 3D models using the estimated poses. All results were obtained within ∼16 ms per object.

trajectory of the golf club was ideally supposed to be estimated passively with a single camera without requiring any marker at all.

Given the proposed passive pose estimation algorithm, two solutions are now possible. Firstly, completely markerless pose estimation of the golf club, in case a 3D CAD model of it is available (see Figure 6.6). Secondly, since in practice there are countless different golf club designs, a fall-back solution has been developed. Here, a passive, light-weight, 3D-printed marker is attached to the golf club, of which a 3D model is known regardless of the club's design. This unfortunately comes with the drawback of having to modify the golf club slightly. However, compared to an active marker solution, the advantages are that it does not require any battery pack and works in direct sunlight.

Mixed Reality The prime potential field of application for the proposed passive, markerless approach are mixed reality systems. The features, that make the method especially suitable for such applications are its low runtime, its high accuracy and its robustness under many challenging conditions (see Figure 6.7). The ability to track multiple objects in real-time enables immersive visualizations in highly dynamic scenarios. Here, the virtual augmentations can realistically be occluded by the

Figure 6.8: Mixed reality visualizations in two different complex scenes with two tracked objects each. Based on the pose estimates the real objects are augmented with virtual attachments (hats, a carrot and a patch of grass), in real-time. Here, despite large perspective changes and both modelled and unmodelled occlusions, the augmentations remain precisely in place.

real objects since their geometry and poses are accurately known (see Figure 6.8). What makes it even more attractive for mixed reality systems is, that the proposed approach can handle fair amounts of ummodelled occlusion e.g. by hands. Therefore, object-specific augmentations will remain in place while a user inspects objects by manipulating them manually. This further allows to turn arbitrary objects into 6DOF motion input devices.

A prototypical mixed reality application similar to that in Figure 6.8 was shown during the official live demonstration sessions of both the *European Conference on Computer Vision* 2016 (see: https://youtu.be/YNhiFPlAYyg) as well as the *International Conference on Computer Vision* 2017 (see: https://youtu.be/BW1jum0G88s), where also the two corresponding papers were presented as part of the main conferences.

6.3 Future Work

The following discusses potential further development possibilities of the presented methods as well as open, related topics that could not be addressed within the scope of this work.

The proposed active marker-based approach could easily be extended to be used within a multi-camera stereo system in order to further increase the depth accuracy.

Current multi-camera systems would thereby directly benefit from the single image LED identification since the markers could be used in order to dynamically calibrate the multi-camera system without having to solve stereo correspondence problems. Being able to estimate the marker pose from a single camera would also vastly increase the tracking volume of such setups and could be used in conjunction with stereo methods, whenever the marker is visible in more than one camera. Here, it would of course also be possible to construct the markers using the familiar passive retro-reflective tags, such that they can be used natively within active-camera systems. This would furthermore save the otherwise required battery pack.

Another way to potentially further improve the pose measurement accuracy of HSRM-Tracking is to combine the markers with an IMU and perform so-called *visual-inertial sensor fusion* [CLD07, KS11, LS13]. This should help to decrease the measurement noise of the rotation regardless of the distance to the camera and thereby enable the use of even smaller markers at larger distances. This combination could then also be used to bridge temporary visual losses of the marker by at least roughly estimating its motion based on the IMU data in the meantime.

In case of the the proposed passive, markerless approach, several further developments are possible. For example, despite its relatively low runtime along with a comparably high accuracy, the main drawback of the current pose detection method is that it requires the templates to be re-trained for every new scene, as discussed in Section 4.9. Therefore, the approach is not suitable for the classical pose detection task, were a familiar object has to be detected in an arbitrary scene. When more computational power is available, this issue could be resolved by combining recent deep learning-based approaches for pose detection (e.g. [RL17, KMT$^+$17]) with the proposed tracking. These are currently achieving state of the art results for this task and can be trained only from semi-synthetic images [KMT$^+$17, HLWK17]. Similar to those of the RBOT dataset, these training images can be generated completely automatically by pasting renderings of the object under different local lighting conditions onto random background images. This makes these techniques robust to different environments without ever having seen the actual real image of the objects before. They do however currently require a powerful GPU in order to achieve similar frame rates as the proposed method based on tclc-histograms does on the CPU.

In analogy to this, a complementing deep learning-based approach for pose tracking i.e. continuous frame-to-frame pose optimization could be investigated, which currently still remains an open problem. Here, based on the insights presented in [HLWK17], the semi-synthetic images of the proposed RBOT dataset could for example be used for training the respective deep convolutional neural networks (CNNs).

If however, it should turn out that accurate 6DOF pose tracking cannot directly be resolved using CNNs only or that this does not lead to an improvement, at least parts of the proposed so-called *classical* pipeline could potentially be enhanced with deep learning strategies. For example the proposed *classical* image segmentation model based on tclc-histograms could be replaced with a CNN in order to compute $P_f(\mathbf{x})$ and $P_b(\mathbf{x})$ within the region-based cost (4.2) with increased robustness to lighting conditions and occlusions. For this, existing network architectures such as the recently presented *Mask R-CNN* [HGDG17] could be build upon. While it was originally designed and trained for the more general task of category-based image segmentation, in [HLWK17] it was demonstrated that Mask R-CNN can also be trained from synthetic images only for accurate object instance specific silhouette segmentation as required for region-based object pose estimation.

Also, the robustness of the direct photometric cost especially towards lighting changes can be further improved. This could either be achieved by using a *classical* approach, e.g. transforming the original image intensities to so-called *dense descriptor fields* [CL14], or as well with help of a deep neural network. For the latter, for example the general strategy of using so-called *semantic textures* [CLD17] could be utilized. Here, the idea is to replace the regular raw RGB image pyramid with learned feature layers coming from a standard CNN pre-trained for image regocnition, such as the famous *VGG* network [SZ14]. Regardless of the employed image representation, the direct photometric pose estimation would significantly benefit from developing a technique for so-called *live texture mapping* of the object's 3D model. Being able to fuse the current live images into a globally optimized texture map that is correctly aligned with the 3D triangle mesh would thereby establish an absolute relationship between the camera images and the object's surface. This would then further allow to perform drift-free photometric frame-to-model tracking, as well as direct photometric pose detection (e.g. template-based).

Apart from these algorithmic enhancements, the proposed markerless method could also be implemented on hand-held mobile devices such as tablets and mobile phones in order to make it even more attractive and suitable for mixed-reality applications. Here, the object pose estimation could further be supported by and fused with the commonly available IMU sensor of such devices (similar to [PKMR15]). Correspondingly, it could also be combined with an onboard monocular SLAM algorithm, e.g. that of Apple's recently released *ARKit*[9] and Google's *ARCore*[10] equivalent. Such a combination would thereby enable advanced mixed reality experiences beyond the current capabilities of ARKit and ARCore, which are essentially restricted to displaying virtual augmentations in static scenes.

Finally, the proposed passive, markerless approach could also form the basis for developing novel monocular methods for estimating the pose of deformable objects while simultaneously reconstructing occurring spatial object surface displacements. This is a far more complex problem than the pure 6DOF pose estimation of rigid bodies, as dealt with in this work. Although this has recently become possible for strongly textured objects [CB15], real-time, monocular pose estimation and 3D reconstruction of weakly textured and textureless soft-bodies still remains an important unresolved problem in computer vision.

[9]The ARKit of Apple: https://developer.apple.com/arkit/
[10]The ARCore platform of Google: https://developers.google.com/ar/discover/

Bibliography

[ABD12] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J. Davison. KAZE features. In *Proceedings of the European Conference on Computer Vision (ECCV), Part VI*, pages 214–227. Springer, 2012.

[AD02] Adnan Ansar and Konstantinos Daniilidis. Linear pose estimation from points or lines. In *Proceedings of the European Conference on Computer Vision (ECCV), Part IV*, pages 282–296. Springer, 2002.

[AGHWK16] C. Anthes, R. J. Garca-Hernndez, M. Wiedemann, and D. Kranzlmller. State of the art of virtual reality technology. In *Proceedings of the IEEE Aerospace Conference*, pages 1–19, 2016.

[AHH10] Bradley Atcheson, Felix Heide, and Wolfgang Heidrich. Caltag: High precision fiducial markers for camera calibration. In *Proceedings of the Vision, Modeling, and Visualization Workshop (VMV)*, pages 41–48. Eurographics Association, 2010.

[Ang06] Edward Angel. *Interactive computer graphics - a top-down approach using OpenGL (4. ed.)*. Addison-Wesley, 2006.

[AOH+18] M Allan, S Ourselin, DJ Hawkes, JD Kelly, and D Stoyanov. 3d pose estimation of articulated instruments in robotic minimally invasive surgery. *IEEE Transactions on Medical Imaging*, 37(5):1204–1213, 2018.

[AOV12] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. FREAK: fast retina keypoint. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 510–517. IEEE Computer Society, 2012.

[Bal98] Robert Stawell Ball. *A Treatise on the Theory of Screws*. Cambridge Mathematical Library. Cambridge University Press, 1998.

[BART11] Filippo Bergamasco, Andrea Albarelli, Emanuele Rodolà, and Andrea Torsello. Rune-tag: A high accuracy fiducial marker with strong occlusion resilience. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 113–120. IEEE Computer Society, 2011.

[BASJ17] David Bouget, Max Allan, Danail Stoyanov, and Pierre Jannin. Vision-based and marker-less surgical tool detection and tracking: a review of the literature. *Medical Image Analysis*, 35:633–654, 2017.

[BKM+14] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *Proceedings of the European Conference on Computer Vision (ECCV), Part II*, pages 536–551. Springer, 2014.

[BLLN07] Selim Benhimane, Alexander Ladikos, Vincent Lepetit, and Nassir Navab. Linear and quadratic subsets for template-based tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–6. IEEE Computer Society, 2007.

[BM04a] Simon Baker and Iain A. Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision (IJCV)*, 56(3):221–255, 2004.

[BM04b] Selim Benhimane and Ezio Malis. Real-time image-based tracking of planes using efficient second-order minimization. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 943–948. IEEE Computer Society, 2004.

[BM07] Selim Benhimane and Ezio Malis. Homography-based 2d visual tracking and servoing. *International Journal of Robotics Research (IJRS)*, 26(7):661–676, 2007.

[BMAK14] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis - A survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2014.

[BMK+16] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Uncertainty-driven 6d pose estimation of objects and scenes from a single RGB image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3364–3372. IEEE Computer Society, 2016.

[BR08] Charles Bibby and Ian D. Reid. Robust real-time visual tracking using pixel-wise posteriors. In *Proceedings of the European Conference on Computer Vision (ECCV), Part II*, pages 831–844. Springer, 2008.

[Bre77] Jack Bresenham. A linear algorithm for incremental digital display of circular arcs. *Communications of the ACM*, 20(2):100–106, 1977.

[BRGC10] Thomas Brox, Bodo Rosenhahn, Juergen Gall, and Daniel Cremers. Combined region and motion-based 3d tracking of rigid and articulated objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(3):402–415, 2010.

[BTG06] Herbert Bay, Tinne Tuytelaars, and Luc J. Van Gool. SURF: speeded up robust features. In *Proceedings of the European Conference on Computer Vision (ECCV), Part I*, pages 404–417. Springer, 2006.

[Can86] John F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 8(6):679–698, 1986.

[CB15] Toby Collins and Adrien Bartoli. Realtime shape-from-template: System and applications. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 116–119. IEEE Computer Society, 2015.

[CC12] Changhyun Choi and Henrik I. Christensen. 3d textureless object detection and tracking: An edge-based approach. In *Proceedings of the*

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3877–3884. IEEE Computer Society, 2012.

[CC13] Changhyun Choi and Henrik I. Christensen. RGB-D object tracking: A particle filter approach on GPU. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1084–1091. IEEE Computer Society, 2013.

[CDM14] Guillaume Caron, Amaury Dame, and Éric Marchand. Direct model based visual tracking and pose estimation using mutual information. *Image and Vision Computing*, 32(1):54–63, 2014.

[CL14] Alberto Crivellaro and Vincent Lepetit. Robust 3d tracking with descriptor fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3414–3421. IEEE Computer Society, 2014.

[CLD07] Peter Corke, Jorge Lobo, and Jorge Dias. An introduction to inertial and visual sensing. *International Journal of Robotics Research (IJRS)*, 26(6):519–535, 2007.

[CLD17] Jan Czarnowski, Stefan Leutenegger, and Andrew J. Davison. Semantic texture for robust dense tracking. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, pages 851–859. IEEE Computer Society, 2017.

[CLSF10] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. BRIEF: binary robust independent elementary features. In *Proceedings of the European Conference on Computer Vision (ECCV), Part IV*, pages 778–792. Springer, 2010.

[CMC03] Andrew I. Comport, Éric Marchand, and François Chaumette. A real-time tracker for markerless augmented reality. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 36–45. IEEE Computer Society, 2003.

[CRD07] Daniel Cremers, Mikaël Rousson, and Rachid Deriche. A review of statistical approaches to level set segmentation: Integrating color, texture, motion and shape. *International Journal of Computer Vision (IJCV)*, 72(2):195–215, 2007.

[CRV+15] Alberto Crivellaro, Mahdi Rad, Yannick Verdie, Kwang Moo Yi, Pascal Fua, and Vincent Lepetit. A novel representation of parts for accurate 3d object detection and tracking in monocular images. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4391–4399. IEEE Computer Society, 2015.

[DC02] Tom Drummond and Roberto Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(7):932–946, 2002.

[DSYT08] Samuel Dambreville, Romeil Sandhu, Anthony J. Yezzi, and Allen Tannenbaum. Robust 3d pose estimation and efficient 2d region-based segmentation from a 3d shape prior. In *Proceedings of the European Conference on Computer Vision (ECCV), Part II*, pages 169–182. Springer, 2008.

[ESC14] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: large-scale direct monocular SLAM. In *Proceedings of the European Conference on Computer Vision (ECCV), Part II*, pages 834–849. Springer, 2014.

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[FH12] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Distance transforms of sampled functions. *Theory of Computing*, 8(1):415–428, 2012.

[Fia05] Mark Fiala. Artag, a fiducial marker system using digital techniques. In *Proceedings of the IEEE Conference on Computer Vision and Pat-*

tern Recognition (CVPR), pages 590–596. IEEE Computer Society, 2005.

[FMSS14] Matthias Faessler, Elias Mueggler, Karl Schwabe, and Davide Scaramuzza. A monocular pose estimation system based on infrared leds. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 907–913. IEEE Computer Society, 2014.

[FPF96] Andrew W. Fitzgibbon, Maurizio Pilu, and Robert B. Fisher. Direct least squares fitting of ellipses. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 253–257. IEEE Computer Society, 1996.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[GHTC03] Xiao-Shan Gao, Xiaorong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25(8):930–943, 2003.

[Gra98] F. Sebastian Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics, GPU, & Game Tools*, 3(3):29–48, 1998.

[Gru41] Johann August Grunert. Das pothenotische problem in erweiterter gestalt nebst über seine anwendungen in der geodäsie. *Grunerts Archiv für Mathematik und Physik*, pages 238–248, 1841.

[Har93] Chris Harris. Tracking with rigid models. In *Active Vision*, pages 59–73. MIT Press, Cambridge, MA, USA, 1993.

[HBN07] Stefan Hinterstoisser, Selim Benhimane, and Nassir Navab. N3M: natural 3d markers for real-time object detection and pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1–7. IEEE Computer Society, 2007.

[HCH10] Edward Hsiao, Alvaro Collet, and Martial Hebert. Making specific features less discriminative to improve point-based 3d object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2653–2660. IEEE Computer Society, 2010.

[HCI+12] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter F. Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(5):876–888, 2012.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988. IEEE Computer Society, 2017.

[HH16] Jonathan Hexner and Rami R. Hagege. 2d-3d pose estimation of heterogeneous objects using a region based approach. *International Journal of Computer Vision (IJCV)*, 118(1):95–112, 2016.

[HHC+11a] Seongkook Heo, Jaehyun Han, Sangwon Choi, Seunghwan Lee, Geehyuk Lee, Hyong-Euk Lee, SangHyun Kim, Won-Chul Bang, Do-Kyoon Kim, and Changyeong Kim. Ircube tracker: an optical 6-dof tracker based on LED directivity. In *Proceedings of the Annual ACM Symposium on User Interface Software and Technology (UIST)*, pages 577–586. ACM, 2011.

[HHC+11b] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 858–865. IEEE Computer Society, 2011.

[HLI+10] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Pascal Fua, and Nassir Navab. Dominant orientation templates for real-time detection

of texture-less objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2257–2264. IEEE Computer Society, 2010.

[HLI+12] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary R. Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Proceedings of the Asian Conference on Computer Vision (ACCV), Part I*, pages 548–562. Springer, 2012.

[HLON94] Robert M. Haralick, Chung-Nan Lee, Karsten Ottenberg, and Michael Nölle. Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision (IJCV)*, 13(3):331–356, 1994.

[HLWK17] Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. On pre-trained image features and synthetic images for deep learning. *CoRR*, abs/1710.10710, 2017.

[HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 1–6. Alvey Vision Club, 1988.

[HSZ+13] Adam Herout, Istvan Szentandrasi, Michal Zacharias, Markéta Dubská, and Rudolf Kajan. Five shades of grey for fast and reliable camera pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1384–1390. IEEE Computer Society, 2013.

[HZ06] Andrew Harltey and Andrew Zisserman. *Multiple view geometry in computer vision (2. ed.)*. Cambridge University Press, 2006.

[KB99] Hirokazu Kato and Mark Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR)*, pages 85–94. IEEE Computer Society, 1999.

[KHKH16] Yoshinori Konishi, Yuki Hanzawa, Masato Kawade, and Manabu Hashimoto. Fast 6d pose estimation from a monocular image using hierarchical pose trees. In *Proceedings of the European Conference on Computer Vision (ECCV), Part I*, pages 398–413. Springer, 2016.

[KLW10] Kiyoung Kim, Vincent Lepetit, and Woontack Woo. Keyframe-based modeling and tracking of multiple 3d objects. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 193–198. IEEE Computer Society, 2010.

[KMB+14] Alexander Krull, Frank Michel, Eric Brachmann, Stefan Gumhold, Stephan Ihrke, and Carsten Rother. 6-dof model based tracking via object coordinate regression. In *Proceedings of the Asian Conference on Computer Vision (ACCV), Part IV*, pages 384–399. Springer, 2014.

[KMT+16] Wadim Kehl, Fausto Milletari, Federico Tombari, Slobodan Ilic, and Nassir Navab. Deep learning of local RGB-D patches for 3d object detection and 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV), Part III*, pages 205–220. Springer, 2016.

[KMT+17] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. SSD-6D: making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1530–1538. IEEE Computer Society, 2017.

[KR17] Tong Ke and Stergios I. Roumeliotis. An efficient algebraic solution to the perspective-three-point problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4618–4626. IEEE Computer Society, 2017.

[Kru13] Erwin Kruppa. Zur Ermittlung eines Objektes aus zwei Perspektiven mit innerer Orientierung. *Sitzungsberichte der Mathematisch Naturwissenschaftlichen Kaiserlichen Akademie der Wissenschaften*, 122:1939–1948, 1913.

[KS11] Jonathan Kelly and Gaurav S. Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *International Journal of Robotics Research (IJRS)*, 30(1):56–79, 2011.

[KSS11] Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2969–2976. IEEE Computer Society, 2011.

[KTIN17] Wadim Kehl, Federico Tombari, Slobodan Ilic, and Nassir Navab. Real-time 3d model tracking in color and depth on a single CPU core. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 465–473. IEEE Computer Society, 2017.

[LA09] Manolis I. A. Lourakis and Antonis A. Argyros. SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software (TOMS)*, 36(1):2:1–2:30, 2009.

[LCS11] Stefan Leutenegger, Margarita Chli, and Roland Siegwart. BRISK: binary robust invariant scalable keypoints. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2548–2555. IEEE Computer Society, 2011.

[Lev44] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly Journal of Applied Mathmatics*, II(2):164–168, 1944.

[LF05] Vincent Lepetit and Pascal Fua. Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends in Computer Graphics and Vision*, 1(1):1–89, 2005.

[LFS13] Ruan Lakemond, Clinton Fookes, and Sridha Sridharan. Resection-intersection bundle adjustment revisited. *ISRN Machine Vision*, 2013, 2013.

[LHM+14] Sergei Lupashin, Markus Hehn, Mark W. Mueller, Angela P. Schoellig, Michael Sherback, and Raffaello D'Andrea. A platform for aerial robotics research and demonstration: The flying machine arena. *Mechatronics*, 24(1):41 – 54, 2014.

[LK81] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 674–679. William Kaufmann, 1981.

[Llo82] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–136, 1982.

[LMF09] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate $O(n)$ solution to the pnp problem. *International Journal of Computer Vision (IJCV)*, 81(2):155–166, 2009.

[Low99] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1150–1157. IEEE Computer Society, 1999.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[LS13] Gabriele Ligorio and Angelo Maria Sabatini. Extended kalman filter-based methods for pose estimation using visual, inertial and magnetic sensors: Comparative analysis and performance evaluation. *Sensors*, 13(2):1919–1941, 2013.

[LT08] Shawn Lankton and Allen Tannenbaum. Localizing region-based active contours. *IEEE Transactions on Image Processing*, 17(11):2029–2039, 2008.

[LTB+17] Marc Lieser, Henning Tjaden, Robert Brylka, Lasse Löffler, and Ulrich Schwanecke. A low-cost mobile infrastructure for compact aerial robots under supervision. In *European Conference on Mobile Robots (ECMR)*, pages 1–6. IEEE Computer Society, 2017.

[MAFK17] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5632–5640. IEEE Computer Society, 2017.

[Mar63] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

[MBC01] Éric Marchand, Patrick Bouthemy, and François Chaumette. A 2d-3d model-based approach to real-time visual tracking. *Image and Vision Computing*, 19(13):941–955, 2001.

[MIS17] Shohei Mori, Sei Ikeda, and Hideo Saito. A survey of diminished reality: Techniques for visually concealing, eliminating, and seeing through real objects. *IPSJ Transactions on Computer Vision and Applications*, 9:17–31, 2017.

[MKCK17] J. Krishna Murthy, G. V. Sai Krishna, Falak Chhaya, and K. Madhava Krishna. Reconstructing vehicles from a single image: Shape priors for road scene understanding. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 724–731. IEEE Computer Society, 2017.

[MLF07] Francesc Moreno-Noguer, Vincent Lepetit, and Pascal Fua. Accurate non-iterative o(n) solution to the pnp problem. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1–8. IEEE Computer Society, 2007.

[MLS94] Richard M. Murray, Zexiang Li, and Shankar Sastry. *A mathematical introduction to robotics manipulation*. CRC Press, 1994.

[MMLK10] Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. The GRASP multiple micro-uav testbed. *IEEE Robotics & Automation Magazine*, 17(3):56–65, 2010.

[MMT15] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[MSKS05] Yi Ma, Stefano Soatto, Jana Kosecka, and S. Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Interdisciplinary Applied Mathematics. Springer New York, 2005.

[MUS16] Éric Marchand, Hideaki Uchiyama, and Fabien Spindler. Pose estimation for augmented reality: A hands-on survey. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 22(12):2633–2651, 2016.

[MZ14] Andreas Masselli and Andreas Zell. A new geometric approach for faster solving the perspective-three-point problem. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 2119–2124. IEEE Computer Society, 2014.

[New12] Richard Newcombe. *Dense visual SLAM*. PhD thesis, Imperial College London, UK, 2012.

[NF05] Leonid Naimark and Eric Foxlin. Encoded LED system for optical trackers. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 150–153. IEEE Computer Society, 2005.

[Nis04] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(6):756–777, 2004.

[NLD11] Richard A. Newcombe, Steven Lovegrove, and Andrew J. Davison. DTAM: dense tracking and mapping in real-time. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2320–2327. IEEE Computer Society, 2011.

[NPM13] Pedro Neto, J. Norberto Pires, and A. Paulo Moreira. 3-d position estimation from inertial sensing: minimizing the error from the process of double integration of accelerations. *CoRR*, abs/1311.4572, 2013.

[NS07] David Nistér and Henrik Stewénius. A minimal solution to the generalised 3-point pose problem. *Journal of Mathematical Imaging and Vision*, 27(1):67–79, 2007.

[ÖCLF10] Mustafa Özuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua. Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(3):448–461, 2010.

[Ols11] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407. IEEE Computer Society, 2011.

[PAF+16] Patrick Poirson, Phil Ammirato, Cheng-Yang Fu, Wei Liu, Jana Kosecka, and Alexander C. Berg. Fast single shot detection and pose estimation. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 676–684. IEEE Computer Society, 2016.

[PDG17] Lerrel Pinto, James Davidson, and Abhinav Gupta. Supervision via competition: Robot adversaries for learning tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1601–1608. IEEE Computer Society, 2017.

[PKMR15] Victor Adrian Prisacariu, Olaf Kähler, David W. Murray, and Ian D. Reid. Real-time 3d tracking and reconstruction on mobile phones. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 21(5):557–570, 2015.

[PLW08] Youngmin Park, Vincent Lepetit, and Woontack Woo. Multiple 3d object tracking for augmented reality. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 117–120. IEEE Computer Society, 2008.

[PM06] Muriel Pressigout and Éric Marchand. Real-time 3d model-based tracking: Combining edge and texture information. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2726–2731. IEEE Computer Society, 2006.

[PR12] Victor Adrian Prisacariu and Ian D. Reid. PWP3D: real-time segmentation and tracking of 3d objects. *International Journal of Computer Vision (IJCV)*, 98(3):335–354, 2012.

[PRDR13] Karl Pauwels, Leonardo Rubio, Javier Díaz, and Eduardo Ros. Real-time model-based rigid object pose estimation and tracking combining dense and sparse visual cues. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2347–2354. IEEE Computer Society, 2013.

[PSK+11] Jin-Hyung Park, Dong-Hwan Seo, Min-Seok Ku, In-Yong Jung, and Chang-Sung Jeong. Multiple 3d object tracking using ROI and double filtering for augmented reality. In *Proceedings of the FTRA International Conference on Multimedia and Ubiquitous Engineering (MUE)*, pages 317–322. IEEE Computer Society, 2011.

[PT11] Nadia Payet and Sinisa Todorovic. From contours to 3d object detection and pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 983–990. IEEE Computer Society, 2011.

[PTVF07] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes: the art of scientific computing, 3rd Edition*. Cambridge University Press, 2007.

[RBW07] Bodo Rosenhahn, Thomas Brox, and Joachim Weickert. Three-dimensional shape knowledge for joint image segmentation and pose tracking. *International Journal of Computer Vision (IJCV)*, 73(3):243–262, 2007.

[RD05] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1508–1511. IEEE Computer Society, 2005.

[RD06] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Proceedings of the European Conference on Computer Vision (ECCV), Part I*, pages 430–443. Springer, 2006.

[RL17] Mahdi Rad and Vincent Lepetit. BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3848–3856. IEEE Computer Society, 2017.

[Ros06] Edward Rosten. *High performance rigid body tracking*. PhD thesis, University of Cambridge, 2006.

[RPK+14] Carl Yuheng Ren, Victor Adrian Prisacariu, Olaf Kähler, Ian D. Reid, and David W. Murray. 3d tracking of multiple objects with identical appearance using RGB-D input. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 47–54. IEEE Computer Society, 2014.

[RPK+17] Carl Yuheng Ren, Victor Adrian Prisacariu, Olaf Kähler, Ian D. Reid, and David W. Murray. Real-time tracking of single and multiple objects from depth-colour imagery using 3d signed distance functions. *International Journal of Computer Vision (IJCV)*, 124(1):80–95, 2017.

[RPMR13] Carl Yuheng Ren, Victor Adrian Prisacariu, David W. Murray, and Ian D. Reid. STAR3D: simultaneous tracking and reconstruction of 3d objects using RGB-D data. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1561–1568. IEEE Computer Society, 2013.

[RRKB11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski. ORB: an efficient alternative to SIFT or SURF. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571. IEEE Computer Society, 2011.

[RSI12] Alexander Reuter, Hans-Peter Seidel, and Ivo Ihrke. BlurTags: spatially varying PSF estimation with out-of-focus patterns. In *Proceed-*

ings of the International Conference on Computer Graphics, Visualization and Computer Vision (WSCG), pages 239–247. Eurographics Association, 2012.

[RT13] Reyes Rios-Cabrera and Tinne Tuytelaars. Discriminatively trained templates for 3d object detection: A real time scalable approach. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2048–2055. IEEE Computer Society, 2013.

[SGZ+13] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew W. Fitzgibbon. Scene coordinate regression forests for camera relocalization in RGB-D images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2930–2937. IEEE Computer Society, 2013.

[SH16] Julius Schöning and Gunther Heidemann. Taxonomy of 3d sensors - A survey of state-of-the-art consumer 3d-reconstruction sensors and their field of applications. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 194–199. SciTePress, 2016.

[SHN+15] Tanner Schmidt, Katharina Hertkorn, Richard A. Newcombe, Zoltan-Csaba Marton, Michael Suppa, and Dieter Fox. Depth-based tracking with physical constraints for robot manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 119–126. IEEE Computer Society, 2015.

[Skl82] Jack Sklansky. Finding the convex hull of a simple polygon. *Pattern Recognition Letters*, 1(2):79–83, 1982.

[SL12] Laura Sevilla-Lara and Erik G. Learned-Miller. Distribution fields for tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1910–1917. IEEE Computer Society, 2012.

[SM07] Geraldo F. Silveira and Ezio Malis. Real-time visual tracking under arbitrary illumination changes. In *Proceedings of the IEEE Conference*

on Computer Vision and Pattern Recognition (CVPR), pages 1–6. IEEE Computer Society, 2007.

[SMR12] Glauco Garcia Scandaroli, Maxime Meilland, and Rogério Richa. Improving ncc-based direct visual tracking. In *Proceedings of the European Conference on Computer Vision (ECCV), Part VI*, pages 442–455. Springer, 2012.

[SP06] Gerald Schweighofer and Axel Pinz. Robust pose estimation from a planar target. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(12):2024–2030, 2006.

[SP08] Gerald Schweighofer and Axel Pinz. Globally optimal o(n) solution to the pnp problem for general camera models. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1–10. British Machine Vision Association, 2008.

[SPP+14] Byung-Kuk Seo, Hanhoon Park, Jong-Il Park, Stefan Hinterstoisser, and Slobodan Ilic. Optimal local searching for fast and robust textureless 3d object tracking in highly cluttered backgrounds. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 20(1):99–110, 2014.

[SRBW12] Christian Schmaltz, Bodo Rosenhahn, Thomas Brox, and Joachim Weickert. Region-based pose tracking with occlusions using 3d models. *Machine Vision and Applications*, 23(3):557–577, 2012.

[SSBJ06] Pedro Santos, André Stork, Alexandre Buaes, and Joaquim A. Jorge. Ptrack: Introducing a novel iterative geometric pose estimation for a marker-based single camera tracking system. In *Proceedings of the IEEE Virtual Reality Conference (VR)*, pages 143–150. IEEE Computer Society, 2006.

[SSS+17] Artur Sagitov, Ksenia Shabalina, Leysan Sabirova, Hongbing Li, and Evgeni Magid. Artag, apriltag and caltag fiducial marker systems: Comparison in a presence of partial marker occlusion and rotation. In

Proceedings of the International Conference on Informatics in Control, Automation and Robotics (ICINCO), pages 182–191. SciTePress, 2017.

[SSV+17] Pavel A. Savkin, Shunsuke Saito, Jarich Vansteenberge, Tsukasa Fukusato, Lochlainn Wilson, and Shigeo Morishima. Outside-in monocular IR camera based HMD pose estimation via geometric optimization. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST)*, pages 1–9. ACM, 2017.

[ST94] Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600. IEEE Computer Society, 1994.

[SW16] Byung-Kuk Seo and Harald Wuest. A direct method for robust model-based 3d object tracking from a monocular RGB image. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops, Part III*, pages 551–562. Springer, 2016.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[TNT17] David Joseph Tan, Nassir Navab, and Federico Tombari. Looking beyond the simple scenarios: Combining learners and optimizers in 3d temporal tracking. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 23(11):2399–2409, 2017.

[TSS16] Henning Tjaden, Ulrich Schwanecke, and Elmar Schömer. Real-time monocular segmentation and pose tracking of multiple objects. In *Proceedings of the European Conference on Computer Vision (ECCV), Part IV*, pages 423–438. Springer, 2016.

[TSS17] Henning Tjaden, Ulrich Schwanecke, and Elmar Schömer. Real-time monocular pose estimation of 3d objects using temporally consistent local color histograms. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 124–132. IEEE Computer Society, 2017.

[TSSC18] Henning Tjaden, Ulrich Schwanecke, Elmar Schömer, and Daniel Cremers. A gauss-newton approach to real-time monocular multiple object tracking. Submitted to *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018.

[TSSS15] Henning Tjaden, Ulrich Schwanecke, Frédéric Stein, and Elmar Schömer. High-speed and robust monocular tracking. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 462–471. SciTePress, 2015.

[TTKK14] Alykhan Tejani, Danhang Tang, Rigas Kouskouridas, and Tae-Kyun Kim. Latent-class hough forests for 3d object detection and pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV), Part VI*, pages 462–477. Springer, 2014.

[VLF04] Luca Vacchetti, Vincent Lepetit, and Pascal Fua. Combining edge and texture information for real-time accurate 3d camera tracking. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 48–57. IEEE Computer Society, 2004.

[WF02] Greg Welch and Eric Foxlin. Motion tracking: No silver bullet, but a respectable arsenal. *IEEE Computer Graphics and Applications*, 22(6):24–38, 2002.

[WLT16] Li-Chen Wu, I-Chen Lin, and Ming-Han Tsai. Augmented reality instruction for object assembly based on markerless tracking. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*, pages 95–102. ACM, 2016.

[WLT+17] Po-Chen Wu, Yueh-Ying Lee, Hung-Yu Tseng, Hsuan-I Ho, Ming-Hsuan Yang, and Shao-Yi Chien. A benchmark dataset for 6dof object pose tracking. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR) Adjunct*, pages 186–191. IEEE Computer Society, 2017.

[WRM+10] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg. Real-time detection and tracking for augmented reality on mobile phones. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 16(3):355–368, 2010.

[WWZ+15] Guofeng Wang, Bin Wang, Fan Zhong, Xueying Qin, and Baoquan Chen. Global optimal searching for textureless 3d object tracking. *The Visual Computer*, 31(6-8):979–988, 2015.

[YTLF16] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. LIFT: learned invariant feature transform. In *Proceedings of the European Conference on Computer Vision (ECCV), Part VI*, pages 467–483. Springer, 2016.

[YWX+12] Haiwei Yang, Fei Wang, Jingmin Xin, Xuetao Zhang, and Yoshifumi Nishio. A robust pose estimation method for nearly coplanar points. In *Proceedings of the International Workshop on Nonlinear Circuits, Communications and Signal Processing (NCSP)*, pages 345–348. The Research Institute of Signal Processing (RISP), 2012.

[YZY10] Hao Yang, Feng Zhang, and Juntao Ye. A six-dof motion tracking system for markered environment. In *Proceedings of the International Congress on Image and Signal Processing (CISP)*, pages 294–298. IEEE Computer Society, 2010.

[Zha00] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(11):1330–1334, 2000.

[ZKS+13] Yinqiang Zheng, Yubin Kuang, Shigeki Sugimoto, Kalle Åström, and Masatoshi Okutomi. Revisiting the pnp problem: A fast, general and optimal solution. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2344–2351. IEEE Computer Society, 2013.

[ZWS+14] Song Zhao, LingFeng Wang, Wei Sui, Huai-Yu Wu, and Chunhong Pan. 3d object tracking via boundary constrained region-based model.

In *Proceesings of the IEEE International Conference on Image Processing (ICIP)*, pages 486–490. IEEE Computer Society, 2014.

[ZYCY17] Lin Zhang, Menglong Ye, Po-Ling Chan, and Guang-Zhong Yang. Real-time surgical tool tracking and pose estimation using a hybrid cylindrical marker. *International Journal of Computer Assisted Radiology and Surgery*, 12(6):921–930, 2017.