Technische Universität München
TUM School of Computation, Information and Technology

Closing the Loop: 3D Object Tracking for Advanced Robotic Manipulation

Manuel Stoiber

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz:
Prof. Dr. rer. nat. Achim J. Lilienthal

Prüfer der Dissertation:
1. Priv.-Doz. Dr. rer. nat. Rudolph Triebel
2. Prof. Andrew J. Davison, Ph.D.
   (Imperial College London, UK)
3. Prof. Dr.-Ing. Alin Albu-Schäffer

Die Dissertation wurde am 03.02.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 13.11.2023 angenommen.

Acknowledgments

The following work was created during my time at the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR) in Oberpfaffenhofen, Germany. I started with my research as a master's student in 2018 and was able to continue my work as a research scientist. It was a great learning experience where I not only got insights into research and academia but also got a glimpse into the future of robotics and computer vision. I want to thank all the forward-looking friends and colleagues at the institute who were always helpful no matter the question and every day contribute to a cooperative and friendly working environment.

Special thanks go to Priv.-Doz. Dr. Rudolph Triebel for introducing me to the world of academia and for supervising this thesis. His unwavering support, as well as continued feedback, greatly contributed to my research. Also, I deeply appreciate his trust and the flexibility he granted me in finding my own way. I would also like to sincerely thank Prof. Dr. Alin Albu-Schäffer for welcoming me to DLR, for supervising my master's thesis, and for assuming the third examiner role for this thesis. His vision for robotic manipulation inspired the foundations of this work, and his continued interest was highly motivating. My deepest gratitude also goes to Dr. Klaus H. Strobl and Dr. Martin Pfanne. They both acted as my mentors and supported me in numerous ways by answering questions, providing feedback, and discussing fruitful ideas. Also, I would like to thank Martin for supervising my master's thesis.

During my work at DLR, the humanoid robot *David* was my primary research platform. I would like to thank my friends and colleagues from the *David* team who continuously extend and maintain the system. In particular, my gratitude goes to Dr. Martin Pfanne and Ana Elvira Huezo Martin, who helped me with numerous experiments on the system. Also, I would like to thank the head of the *David* team Sebastian Wolf for his continued support. In addition, for the integration of the developed tracker on the *MiroSurge* system for minimal-invasive robotic surgery, special thanks go to Florian Steidle and Anne Elisabeth Reichert.

Moreover, I thank everyone who contributed to various publications. In addition to previously mentioned colleagues, I would like to highlight the help of Martin Sundermeyer and Wout Boerdijk, with whom I created the *RTB* dataset. Many thanks also go to the students Anne Elisabeth Reichert, Mariam Elsayed, and Jakob Rothe for their work. My deepest gratitude also goes to Jakob Bachler, Dr. Klaus H. Strobl, Anne Elisabeth Reichert, and Maximilian Ulmer for proofreading this dissertation. Furthermore, I want to sincerely thank Prof. Andrew J. Davison from Imperial College London for taking the second examiner role for this thesis.

Finally, I would like to thank my family and friends for their continued love and support. My deepest gratitude goes to my girlfriend Daniela for her love and patience and for keeping me company during numerous home-office days. Without you, the last few years would not have been the same.

Abstract

In robotics, it is often assumed that the world is static, forward kinematics are perfectly known, and interactions are fully deterministic. However, while those assumptions are true for some applications, in general, they significantly limit the complexity of manipulation tasks and the design of robotic hardware. To overcome such limitations, this thesis focuses on 3D object tracking techniques for advanced robotic manipulation. The goal is to develop a flexible and efficient algorithm that uses camera data to provide continuous pose estimates for the robot's end effector and relevant objects. Given such feedback, it is possible to adopt more human-like visual servoing approaches, react to changes in the environment, and facilitate safe and efficient robot designs.

The first main contribution of this work is a highly-efficient approach to region-based object tracking. It features a well-founded probabilistic model that considers image information sparsely along so-called correspondence lines. Experiments demonstrate that the algorithm performs significantly better than existing region-based methods while being approximately one order of magnitude faster. Subsequently, based on this approach, an extension to multi-modality tracking is discussed. The method allows to fuse depth, texture, and multi-region information from various cameras in a highly-modular probabilistic formulation. Again, experiments on different datasets show that the resulting algorithm is highly efficient and outperforms both conventional and deep learning-based methods by a considerable margin. Finally, the third part of this thesis considers the tracking of multi-body objects, which are composed of rigid bodies that are connected by joints. For this, a flexible framework is proposed that allows the extension of existing rigid object tracking methods to multi-body tracking. It considers both tree-like and closed kinematic structures and facilitates the flexible configuration of joints and constraints. For a detailed evaluation, a highly-realistic synthetic dataset is introduced that features a large number of sequences and various robots. Experiments demonstrate the excellent performance of the developed framework and tracker.

All theoretical concepts are implemented into the multi-body, multi-modality, and multi-camera tracking library *M3T*, which we released as open-source software. It provides a highly-modular architecture that supports a wide range of kinematic structures, object characteristics, and camera setups. In addition, the algorithm allows the incorporation of robot joint measurements of varying accuracy and reliability. In real-world experiments on the humanoid robot *David* and the *MiroSurge* system, it is used to continuously provide the pose and configuration of manipulated objects and robot end effectors. The developed approach thereby allows to close the perception-action loop and facilitates new capabilities for advanced robotic manipulation.

Contents

List of Abbreviations

1D	One-dimensional	23
2D	Two-dimensional	11
3D	Three-dimensional	2
6DoF	Six degrees of freedom	3
AAE	Augmented Autoencoder	71
ADD	Average-distance	62
ADD-S	Average-shortest-distance	62
AR	Augmented reality	3
AUC	Area-under-curve	45
CAD	Computer-aided design	12
CNN	Convolutional neural network	6
CPU	Central processing unit	42
EKF	Extended Kalman Filter	17
FPS	Frames per second	65
GPU	Graphics processing unit	12
GT	Ground-truth	41
ICP	Iterative Closest Point	14
ID	Identifier	87
IM	Iterative matching	73
IMU	Inertial measurement unit	17
LN	Links and Nodes	113
MH	Multi-hypothesis	73
MOT	Multiple object tracking	5
PDF	Probability density function	53
PnP	Perspective-n-Point	107
RGB	Red-green-blue	17
RGB-D	Red-green-blue-depth	17
RMS	Root-mean-square	67
RTB	Robot Tracking Benchmark	7
SDF	Signed distance function	65
SLAM	Simultaneous localization and mapping	5

UKF	Unscented Kalman Filter	17
YCB	Yale-CMU-Berkeley	62

List of Figures

List of Tables

List of Algorithms

1 Introduction

1.1. Motivation

Imagine you have to remove a mug from a dishwasher with closed eyes and wearing a thick glove. With your most essential senses impeded, even such a simple task might turn into a challenge. While you probably remember the mug's rough location from the moment you closed your eyes, you will most likely not move your hand to the perfect location. Even if, by chance, the object is within reach when you try to grasp it, the mug might move in ways that you cannot anticipate. Once the object is in your hand, you have to turn it into an upright position and place it at the desired location. If you get lucky, it lands close to where you intended, wobbling only a little. It might, however, be completely off, tip over, or fall down. Given the high probability of failure, in general, humans do not solve manipulation tasks in this way. Instead, they use continuous visual and tactile feedback from their eyes, muscles, and skin. Based on this information, they are able to close the perception-action loop, allowing them to continuously react and adapt to changes in the environment and the object's location.

In contrast to humans, most robotic systems have to solve manipulation tasks without that kind of feedback. For many applications, objects are placed at a defined initial location, or their pose is estimated from a single image. Manipulation is then executed blindly without continuous feedback. To increase robustness and enable at least some tasks, simplifications have to be made. In most cases, the environment is considered to be static, without objects moving in unpredictable ways before or during manipulation. Also, to know the end effector's location relative to the world or a camera coordinate frame, robots are supposed to be stiff and provide high-quality joint measurements. While robots are successfully deployed in many applications where those constraints are fulfilled, the described simplifications significantly limit both the complexity of manipulation tasks and the design of robotic hardware.

The static world assumption does, for example, not allow for humans to interact with objects after poses were estimated. Also, very precise robot motion is required to ensure nothing moves in unpredictable ways. Handling a diverse set of objects with flexible but imprecise end effectors, such as robotic hands, is often very difficult. Instead, in many cases, a task- and workpiece-specific setup with dedicated grippers and fixtures has to be employed. Moreover, complex yet natural interactions like the manipulation of an object within a robotic hand remain impossible. In addition, the requirement for high-quality forward kinematics also imposes constraints on the design of robotic hardware. In order

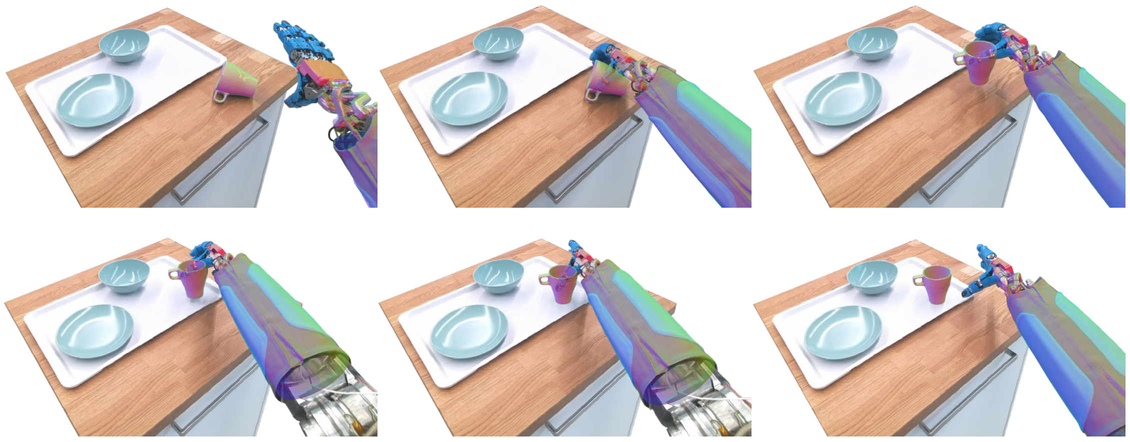


Figure 1.1.: Manipulation of a mug object by the humanoid robot *David*. Pose predictions from a 3D object tracking approach are visualized as an overlay. Without accurate predictions for the mug and robot, the task would be unfeasible.

to know the relative location between robot joints, a very stiff mechanical structure is required, adding weight and cost. Also, joint measurements have to be highly accurate. For example, tendon-driven mechanisms, where sensors are placed at the end of cables that are exposed to elastic elongation, friction, imprecise guides, and decalibration, are highly problematic. Furthermore, soft mechanisms with purely elastic elements, for which forward kinematics are difficult to estimate, typically remain impractical.

To overcome such limitations, this thesis focuses on the development of three-dimensional (3D) object tracking techniques for advanced robotic manipulation. Using camera data, our goal is to continuously provide the pose and configuration of known objects and robots. Together with tactile information, this visual feedback can be used to control interactions with objects and the environment. Consequently, more lightweight and elastic mechanical structures can be deployed, leading to more dynamic and cost-efficient robots. In addition, soft and elastic components can be realized, which increase the robustness of systems to collisions and improve the safety for human-robot collaboration. Finally, continuous feedback is also critical for robots to react to changes in the environment. Given those capabilities, it is possible for humans to interact with objects during operation. Also, it facilitates the effective use of complex end effectors such as robotic hands and the creation of more advanced manipulation skills.

An example sequence that illustrates the usage of 3D object tracking for robotic manipulation is given in Fig. 1.1. The sequence shows the humanoid robot *David*, which features a fully elastic design, grasping a mug and placing it on a tray. Given imprecise forward kinematics and unpredictable object movements, the task would be unfeasible without continuous pose predictions. In summary, with the development of new 3D object tracking techniques, our goal is to close the perception-action loop and allow robots to adopt more human-like strategies for object manipulation.

1.2. Requirements

In the following, we build on the motivation to continuously provide the pose and configuration of known objects and robots using camera data. While 3D object tracking is also highly important for augmented reality (AR), where digital information is imposed on real-world images, in this work, we focus on robotic manipulation. Note, however, that requirements are mostly the same and that developed solutions can often be applied in both fields. For our research, we also concentrate on tracking instead of detection and global pose estimation. While global estimation algorithms search for an object in a single image and over the entire pose domain, tracking follows an object over consecutive images and uses local optimization. For the initialization of the tracker, prior knowledge or global detection and pose estimation algorithms can be used. Because of the simplified task, tracking is typically more efficient and accurate. Also, it often produces more consistent results. Consequently, we believe that tracking is better suited to provide continuous feedback for robotic manipulation. Based on our focus on tracking and the general task of advanced robotic manipulation, the following requirements are derived:

Robust Since tracking uses local optimization, it is possible that objects get lost and predictions become erroneous. In such a case, it is often impossible to recover the correct pose without global pose estimation. For real-world applications, it is critical to ensure robustness and make such events as rare as possible. Typically, tracking loss can happen if the motion from one image frame to the next is too big. Also, tracking can fail if information is missing or ambiguous because of effects such as occlusions, motion blur, textureless surfaces, object symmetries, or background clutter. It is, therefore, crucial that the tracker has a large basin of convergence and robustly considers a diverse set of information.

Accurate In order for a robot to grasp an object and manipulate it, sufficient accuracy is essential. While there is no upper limit with respect to accuracy, since higher will always be better, our goal is to support manipulation tasks humans encounter in everyday life. As described in the motivation, a typical example could be the grasping and positioning of a mug. To successfully execute such a task, a translational error below one centimeter and a rotational error of a few degrees are desirable. In contrast to augmented reality, it is thereby not only important to have small errors in image space, but that predictions are accurate in 3D space for all six degrees of freedom (6DoF).

Fast Because visual feedback is used by robots to react to changes in the environment, speed is critical. In most cases, this means that tracking algorithms should be able to process every image provided by the camera. Also, to ensure responsiveness and avoid instabilities in control algorithms, low latency is required. Given that consumer cameras typically run at a frequency of 30 Hz, this can be seen as a lower bound for the speed of a tracking algorithm. In addition, since frame-to-frame pose differences decrease if more images are considered, using all available frames further improves robustness.

Efficient In real-world robotic systems, a large number of computational processes, which range from control to planning, have to be executed in parallel. As a consequence, computational resources are often highly contested. This is especially true for mobile platforms, where both energy and space are constrained. Also, streaming images to external servers is often difficult and adds additional delay that should be avoided for highly dynamic tasks. To ensure that tracking methods can be used on mobile platforms with a limited computational budget, it is important that algorithms are as efficient as possible.

Kinematic Structures Many real-world objects consist of connected bodies that form kinematic structures. Typical examples range from various tools over furniture and appliances to complex machines. Most prominently for our task, almost every robot is a multi-body system, where individual links are connected by joints. While, in theory, individual bodies could be tracked independently, in most cases, this makes the task unfeasible. Consequently, to provide high-quality predictions for robotic end effectors, it is essential to consider kinematic information.

Proprioception Robotic systems provide a wide variety of sensory data. Prominent examples are joint sensors that measure the relative location between individual links. While this kind of proprioception can be found in almost every robot, the quality of measurements varies widely. Examples range from highly-accurate industrial robots to soft or tendon-driven mechanisms where sensors only provide a rough estimate. Nevertheless, independent of accuracy, information from joint measurements is highly valuable. Since proprioception restricts the relative pose between individual bodies and, in some cases, the camera, it significantly constrains the optimization and helps to improve robustness. In addition, if errors are small, measurements can also be used to improve accuracy.

Modular In general, a wide variety of robotic manipulators exists. Kinematic structures range from simple chains and trees, which can be found in articulated robots, to closed structures, such as parallel robots. Individual links are often connected by revolute, prismatic, or spherical joints, for which sensors might or might not provide measurements. Also, while some robots only use color cameras, others integrate depth sensors. In addition to system-dependent aspects, object characteristics can also vary widely. Examples include objects without texture, various symmetries, or surface properties that are problematic for depth cameras. To support a wide variety of different robots, kinematic structures, camera setups, and object characteristics, the tracker has to be highly modular and allow for a flexible configuration.

Usability Since tracking and pose estimation is only a small part of the full robotics stack, users might come from different domains with limited experience in computer vision. Together with the desire to quickly integrate new applications, developed tracking solutions should facilitate a fast and easy configuration of new objects. Furthermore, it is desirable to require as little data as possible. For example, while geometric 3D models are often available, creating realistic 3D scans with texture requires a lot more effort. Likewise, recording or generating training data with sufficient realism and accurate annotations is typically very time-consuming and challenging for inexperienced users. Given those considerations, in the case of similar performance, data-efficient techniques should always be preferred.

1.3. Existing Techniques

In this work, we focus on model-based 3D object tracking. In contrast to multiple object tracking (MOT) (Luo et al. 2021; Meinhardt et al. 2022; Y. Zhang et al. 2022; Zeng et al. 2022), which follows objects in image space, the goal is to provide object positions and orientations relative to the camera in 3D space at high frequency. Furthermore, compared to simultaneous localization and mapping (SLAM) (Davison 2003; Davison et al. 2007; Newcombe et al. 2011; Forster et al. 2014; Engel et al. 2018; C. Campos et al. 2021), we do not estimate the camera's pose relative to a mostly static world but assume a dynamic world for which we estimate the 6DoF poses of multiple moving objects. In addition, for kinematic structures, we predict object configurations. Together with object positions and orientations, this directly corresponds to the 6DoF poses of all individual rigid bodies in the system. In our work, we also assume that, for the estimation, object models and kinematic structures are known. Compared to category-level trackers (C. Wang et al. 2020; Weng et al. 2021; Deng et al. 2022), which do not require a model during inference, model-based techniques are often significantly more accurate and robust. Also, they fit well with model-based planning and control algorithms. As a consequence, given the current state of the art, we believe that model-based 3D object tracking approaches are best suited to provide accurate and robust visual feedback for closed-loop robotic manipulation. In the following, we provide a short summary of existing techniques. A detailed overview can be found in Chapter 2.

For model-based 3D object tracking, a wide variety of methods and techniques exist. In general, techniques can be categorized by their use of edges, keypoints, direct optimization, object regions, depth information, and deep learning. Edge-based methods fit edges from the object model to high-intensity gradients in the image (Harris and Stephens 1988; Drummond and Cipolla 2002). While such methods are able to track textureless objects, they often struggle with texture and background clutter. Keypoint-based methods, on the other hand, use characteristic points on the object's surface to estimate the pose (Vacchetti et al. 2004; Lourakis and Zabulis 2013). They typically provide a large basin of convergence and are robust to illumination changes. However, they require strong texture to provide good results. Methods based on direct optimization (Crivellaro and Lepetit 2014; Caron et al. 2014) minimize a pixel-wise error over the object silhouette. While they often have a smaller basin of convergence and are less robust to illumination changes, they require less varied texture and are typically more accurate than keypoint-based techniques.

To facilitate the tracking of textureless objects in cluttered environments, region-based approaches have become very popular (Prisacariu and Reid 2012; Tjaden et al. 2018). They fit the object model to best explain the segmentation between the object silhouette and the background based on image statistics such as color. However, while they provide excellent results in challenging conditions, most methods feature computationally expensive dense formulations that hardly run in real-time. With the development of cheap consumer depth cameras, tracking methods that minimize the distance between the object model and depth measurements have also been developed (Rusinkiewicz and Levoy 2001; Fitzgibbon 2003). Depending on the depth sensor and image quality, they are able to track textureless objects and provide accurate results. Finally, data-driven techniques that use architectures like convolutional neural networks (CNNs) have also been proposed (Garon and Lalonde 2017; Y. Li et al. 2018). While such methods hold great promise for the future, they require significant computational resources and large amounts of training data. Currently, only very few methods run in real-time. Moreover, because of limited evaluations on benchmarks, it is unclear how current deep learning-based methods compare to conventional techniques.

Finally, for the tracking of multi-body objects, various methods that combine and extend different techniques have been developed (Brox et al. 2010; Krainin et al. 2011). In general, almost all methods either use *preimposed* constraints based on the formulation of Lowe (1991) or *postimposed* constraints according to the work of Drummond and Cipolla (2002). The algorithm of Lowe (1991) uses Jacobian matrices to describe the variation of individual measurements with respect to a minimal set of parameters that describe the pose and configuration of the kinematic structure. The method of Drummond and Cipolla (2002), on the other hand, first predicts the independent 6DoF variation of individual bodies and later ensures compatibility using velocity constraints. Both formulations have been applied in combination with various tracking techniques. Nevertheless, while they work very well for tree-like topologies, they cannot be applied to closed kinematic structures. Also, while the general concepts were proven in various publications, only very few modern algorithms are available that allow the flexible configuration of kinematic structures. Two examples are *DART* (Schmidt et al. 2015b) and *SimTrack* (Pauwels and Kragic 2015). While *DART* uses depth information, *SimTrack* integrates depth and optical flow, which is a variant of direct optimization. However, with an already outdated optical flow approach and only a very limited set of considered information, both algorithms fall short of the full potential that modern tracking techniques offer.

1.4. Contribution

The following work improves and combines existing model-based 3D object tracking techniques while considering their suitability for advanced robotic manipulation. All developed approaches are integrated into a highly-modular library for multi-body, multi-modality, and multi-camera tracking that we call *M3T*. The library is open-source since 2020 and has been well-received by the community.[1] For our approach, we start with region-based tracking. Given that many robots and man-made objects have uniform surfaces without rich texture, such techniques are particularly well-suited for our domain. Existing region-based methods, however, typically feature computationally expensive dense formulations that contradict our requirements for efficiency and speed. To overcome such limitations, we develop a novel sparse formulation that bridges this gap in efficiency and, in addition, improves tracking quality.

Based on this method, we design a modular approach that allows the flexible combination of multiple modalities. We thereby consider depth, texture, and multi-region information. The depth modality minimizes the distance of surface points, while the texture modality uses keypoint features, and the multi-region approach extends region-based tracking to consider different regions on the object surface. While single-region and depth work very well for textureless objects, for which only geometric information is available, multi-region and texture are able to consider visual appearance. For the developed multi-modality tracker, we conduct a detailed evaluation that shows that our approach is highly efficient and outperforms existing methods by a considerable margin. In addition, we check the algorithm's potential for pose refinement, which is typically conducted after global 6DoF pose estimation. Finally, the evaluation not only allows us to assess our own approach but also to gain new insights into the current state of deep learning-based object tracking.

Subsequently, to consider kinematic structures, a multi-body tracking framework is developed. It facilitates the flexible configuration of joints and constraints between individual rigid bodies. The framework not only supports tree-like topologies but also allows the accurate modeling of closed kinematic structures, which are present in various robots and mechanisms. With respect to rotational constraints, particularly noteworthy are novel equations that cover the entire rotation space and which ensure that the algorithm converges in a single iteration. After an extension of the previously developed multi-modality tracker to multi-body tracking, a detailed evaluation is conducted. For this, the *Robot Tracking Benchmark (RTB)* is introduced. It is a highly-realistic synthetic dataset that features a large number of sequences and various robots. It was generated using the procedural rendering pipeline *BlenderProc* and is publicly available[2].

Finally, we extend our multi-body, multi-modality, and multi-camera 3D object tracking library *M3T* to incorporate joint measurements of varying accuracy. The resulting approach is integrated into the humanoid robot *David* (Grebenstein et al. 2011) and the system for minimal-invasive robotic surgery *MiroSurge* (Hagn et al. 2010). In both systems, the tracker continuously provides pose estimates for manipulated objects and robot end effectors. It thereby allows to close the perception-action loop. In detail, the most important contributions of our work are as follows:

[1]`https://github.com/DLR-RM/3DObjectTracking`
[2]`https://zenodo.org/record/7548537`

Sparse Region Approach A sparse formulation that introduces so-called correspondence lines, which model the probability of the object's contour location. The approach also features a highly-efficient scale-space formulation, as well as an optimization strategy that differentiates between global and local views.

Extended Segmentation Model Novel smoothed step functions for region-based tracking that allow the modeling of both local and global uncertainties. For the resulting probabilistic model, a detailed theoretical analysis is presented that shows how different parameter settings affect the posterior probability distribution of the contour location.

Multi-modality Tracking A highly-efficient, flexible tracking algorithm that combines depth, texture, and multi-region information. The previously developed single-region approach is thereby extended to multi-region tracking. Experiments compare our algorithm to state-of-the-art 3D object tracking and global pose estimation methods, assess the algorithm's potential for pose refinement, and allow us to gain new insights into the performance of deep learning-based tracking techniques.

Multi-body Tracking A unified framework that combines both the projection to a minimal parameterization and the application of pose constraints into a single formulation. It is the first approach that allows the accurate modeling of closed kinematic structures. Based on this framework, which allows the integration of any method that uses Newton-like optimization, a multi-body, multi-modality, and multi-camera tracker is developed.

Rotational Constraint Equations Equations to implement rotational constraints that directly operate on the rotation vector and cover the entire space of possible rotations. For the equations, a thorough mathematical proof is derived that shows that our constraints enforce an exact solution for which pose differences converge in a single iteration.

Robot Tracking Benchmark A synthetic dataset with a large number of sequences and various moving robots that facilitates the quantitative evaluation of multi-body tracking algorithms. The dataset contains not only photo-realistic sequences but also provides different depth qualities to simulate real-world conditions.

M3T Library A multi-body, multi-modality, and multi-camera tracking algorithm that supports a wide range of kinematic structures, object characteristics, and camera configurations. It is able to consider known and unknown occlusions, is highly efficient, and provides results in real-time. The overall framework is very modular and allows a flexible combination of different components such as cameras, links, constraints, modalities, viewers, detectors, refiners, publishers, and subscribers.

System Integration The developed tracker is integrated into the humanoid robot *David* and the *MiroSurge* system for robotic surgery. In both cases, it utilizes camera images and constrains predictions using joint measurements. By providing continuous estimates for the pose and configuration of manipulated objects and robotic end effectors, the tracker is able to facilitate new manipulation capabilities.

1.5. Outline

This work is motivated by a preceding master's thesis (Stoiber 2019) and incorporates material from multiple publications in conferences and journals that build on each other. Inspired by their causal structure, the thesis is organized as follows: In Chapter 1, we explain why 3D object tracking is essential for advanced robotic manipulation and discuss requirements that follow from this application. After a short overview of existing techniques, the contribution of this thesis is summarized. Chapter 2 presents related work and gives a detailed survey of 3D object tracking methods, categorized by their use of edges, keypoints, direct optimization, object regions, depth data, and deep learning. In addition, techniques and algorithms for multi-body tracking are discussed.

Subsequently, we start with the presentation of our own work. Chapter 3 thereby considers region-based tracking based on *SRT3D* (Stoiber et al. 2020; Stoiber et al. 2022a) and also includes modifications proposed for *ICG* (Stoiber et al. 2022b). The chapter first presents the derivation of a probabilistic correspondence line model. This is followed by the development of a highly-efficient 6DoF tracking approach that utilizes this model. After providing implementation details, a thorough evaluation is presented. In Chapter 4, a multi-modality tracker is introduced that allows the combination of depth, texture, and multi-region information. It includes developments and experiments from *ICG* and *ICG+* (Stoiber et al. 2022b; Stoiber et al. 2023a). After providing a probabilistic model for each modality, implementation details are discussed. This is followed by a thorough evaluation that distinguishes between geometry-based tracking and the additional incorporation of visual appearance. Lastly, Chapter 5 covers multi-body tracking based on *Mb-ICG* (Stoiber et al. 2023b). It starts by introducing a general framework that is independent of a particular parameterization. Afterwards, equations required for joints and constraints are derived for the axis-angle representation. For the developed constraint equations, we also analyze convergence properties. Subsequently, implementation details and extensions compared to previous algorithms are discussed. Finally, after introducing the *RTB* dataset, the chapter is concluded with a thorough evaluation that assesses the developed framework and tracker.

Given the proposed approach, Chapter 6 focuses on applications in the real world. For this, we first provide an overview of the developed open-source library *M3T* and explain its architecture. In addition, the initialization of the tracker is discussed, and an approach for the incorporation of measurements from joint sensors is presented. Based on the *M3T* library, we then consider the integration into the humanoid robot *David* and the *MiroSurge* system for minimal-invasive robotic surgery. Finally, Chapter 7 concludes this thesis with a summary of our work and a discussion of possible future research directions.

2
Related Work

2.1. Introduction

For the task of model-based 3D object tracking, a wide variety of methods and different techniques exist. Based on surveys (Lepetit and Fua 2005; Yilmaz et al. 2006), as well as recent developments, they can be categorized by their use of edges, keypoints, direct optimization, object regions, depth data, and deep learning. In the following, the current state of the art for those techniques is introduced. Most of the discussed methods thereby only consider rigid objects. Approaches that facilitate the tracking of multi-body systems with joints and constraints are discussed separately in the final section. Note that Sections 2.3, 2.5, and 2.8 are based on previous publications (Stoiber et al. 2023a; Stoiber et al. 2022a; Stoiber et al. 2023b).

2.2. Edge-based Tracking

While some early approaches fitted the object model to previously extracted edges in the image (Lowe 1992; Gennery 1992), most algorithms simply look for strong gradients around the current estimate without using global edge detection. One of the most prominent methods that established this approach is *RAPID* (Harris and Stennett 1990). It searches for strong gradients along lines perpendicular to the projected contour to find correspondence points. Based on those 2D-3D correspondences, it then optimizes the object pose. To improve the robustness of this approach, Armstrong and Zisserman (1995) suggested the detection of outliers using *RANSAC* (Fischler and Bolles 1981), while G. Simon and Berger (1998) and Drummond and Cipolla (2002) proposed the integration of robust estimators. Also, Marchand et al. (2001) incorporated a robust estimator, as well as a motion model, to first estimate a two-dimensional (2D) affine transformation. Subsequently, for the full 6DoF pose, their method iteratively minimizes a cost function based on image gradients and contour normal vectors. A similar cost function was later adopted by Bugaev et al. (2018). However, they used Basin-hopping to refine a pose that is first estimated using optical flow. In another approach, Comport et al. (2006) formulated 3D object tracking in terms of virtual visual servoing and analytically computed the distance to geometric primitives. Also, Imperoli and Pretto (2015) proposed a representation that encodes the minimum distance to an edge point in a joint location and orientation space, which can also be used for pose refinement.

To further improve robustness, the consideration of multiple edge hypotheses along search lines was also suggested. One of the first algorithms was developed by Vacchetti et al. (2004). It was later extended by Wuest et al. (2005) with the integration of Gaussian mixture models. A combined method that uses keypoint features and edges was also designed by Rosten and Drummond (2005). A different way to improve robustness in highly-cluttered scenes is the incorporation of region information. The first approach that goes in this direction was developed by Seo et al. (2014). It uses region information and optimal local searching to evaluate possible correspondences. The algorithm was later extended by G. Wang et al. (2015), who integrated global optimal searching with a graph model. The method of Huang et al. (2020) filters points in a similar way but, in addition, utilizes region-based confidence values. Apart from filtering points, the combination of region and edge information into a single energy function was proposed by J.-C. Li et al. (2021). Finally, X. Sun et al. (2021) developed a so-called contour part model, where contour patches are matched to gradients in the image.

A different direction, which, again, has the goal of improving robustness in challenging environments, is the use of particle filtering. While Isard and Blake (1998) showed that particle filters can be used for tracking, Pupilli and Calway (2006) developed the first model-based 3D object tracking method. Later, Klein and Murray (2006) implemented an approach that utilizes the graphics processing unit (GPU) to track objects in real-time. In another approach, Mörwald et al. (2009) incorporated both edges from a 3D computer-aided design (CAD) model and unmodeled edges from object texture. Also, Teulière et al. (2010) assigned multiple edge hypotheses to different classes that are integrated into a particle filter. A highly-efficient approach that utilizes particle- and pixel-level parallelism on the GPU was later developed by Brown and Capson (2012). At the same time, Choi and Christensen (2012a) integrated keypoint features to initialize particles, used *RANSAC* to validate edge correspondences, and adopted autoregressive state dynamics to guide particles effectively. Also, in a later work, Choi and Christensen (2012b) proposed the utilization of particle annealing and to use Chamfer matching of edge templates instead of keypoint features for initialization. Finally, B. Wang et al. (2019) developed a method that employs a distance transform from extracted edges and a particle filter that validates poses based on the consistency of edge directions.

2.3. Keypoint-based Tracking

To consider texture information, keypoint-based methods are very popular. In general, keypoint features mark distinct locations on the object surface and provide a descriptor that depends on the surrounding region. For 3D object tracking, most methods reconstruct 3D points, establish 2D-3D correspondences, and then optimize for the pose. Both Vacchetti et al. (2004) and Lourakis and Zabulis (2013) followed this approach. To obtain 3D model points, their methods use pre-registered reference frames and then minimize the reprojection error in the image. In addition, inspired by Shan et al. (2001), Vacchetti et al. (2004) developed a local bundle adjustment technique, which uses homographies to approximate transfer functions that project keypoints between frames. In contrast to those methods, Brox et al. (2010) proposed the utilization of a mesh model to reconstruct 3D points and minimize an error in 3D space. Similarly, Krainin et al. (2011) used depth images to determine the 3D location of keypoints. Their algorithm then optimizes the 3D point-to-point distance to estimate the object's pose.

For keypoint features, a wide variety of detectors and descriptors has been developed. Work can thereby be traced back to Moravec (1980) and Harris and Stephens (1988). Especially with the development of *SIFT* (Lowe 2004), which is both highly accurate and robust, the use of keypoint features became very popular. Approaches inspired by *SIFT* that improve computational efficiency were later presented with *SURF* (Bay et al. 2006), *DAISY* (Tola et al. 2010), or *KAZE* (Alcantarilla et al. 2012). Also, to improve the speed of feature detection, the intensity-based algorithm *FAST* (Rosten and Drummond 2005) was developed. It was further improved with the *ORB* detector (Rublee et al. 2011), which selects features using the Harris response. Together with lightweight binary descriptors such as *BRIEF* (Calonder et al. 2010), *ORB* (Rublee et al. 2011), *BRISK* (Leutenegger et al. 2011), or *FREAK* (Alahi et al. 2012), highly-efficient algorithms can be designed. In recent years, variants that use deep learning at various stages have also been developed. Prominent examples are *LIFT* (Yi et al. 2016), *DSAC* (Brachmann et al. 2017), *SuperPoint* (DeTone et al. 2018), *D2-Net* (Dusmanu et al. 2019), *SuperGlue* (Sarlin et al. 2020), or *LoFTR* (J. Sun et al. 2021). However, while those approaches show excellent results, they are often less efficient than conventional methods. Consequently, they are not well-suited to provide real-time feedback for robotic manipulation with limited computational resources. For a comprehensive survey on deep learning-based keypoint features and techniques, we refer interested readers to Ma et al. (2021).

2.4. Direct Optimization

In addition to keypoint features, direct optimization is also used to incorporate texture information. Most approaches are thereby inspired by Lucas and Kanade (1981). Their method minimizes the photometric difference between two images and is widely used for image alignment. Based on the original algorithm, numerous modifications were proposed that are summarized in a survey by Baker and Matthews (2004). While early model-based 3D object tracking approaches considered planes on the object surface (Jurie and Dhome 2002) or primitive geometries such as cylinders (La Cascia et al. 2000), the approach was later extended to arbitrary shapes (Sepp 2006). Also, instead of directly using pixel intensities, Crivellaro and Lepetit (2014) developed so-called *Descriptor Fields*, which improve performance for poorly-textured and specular objects. In contrast to this approach, Caron et al. (2014) suggested the use of mutual information between rendered and real-world images. Later, a constrained objective function that considers intensity variations using surface normals under the Lambertian assumption was proposed by Seo and Wuest (2016). Also, with *Dense Feature Fields* and dynamic template rendering, the approach of Crivellaro and Lepetit (2014) was extended by Zhong et al. (2018). Finally, combinations with region-based techniques were developed by Zhong and L. Zhang (2019), Y. Liu et al. (2020), and F. Liu et al. (2021).

Similar to direct optimization, methods that use optical flow have also been inspired by Lucas and Kanade (1981). However, instead of directly regressing the pose, algorithms first estimate the relative motion of each pixel from one frame to the next. Based on this data, they then compute the pose transformation of objects. A method that uses this approach together with region- and keypoint-based techniques was developed by Brox et al. (2010). Later, Pauwels et al. (2013) combined depth and optical flow. Also, to avoid drift, they proposed an approach they call *Augmented Reality Flow*. It computes optical flow between the current image and a rendering of the object model. The method was later extended for the tracking of articulated objects (Pauwels et al. 2014).

2.5. Region-based Tracking

Region-based methods use image statistics to differentiate between a foreground region corresponding to the object and a background region. Typically, color statistics are used to model the membership of each pixel. Based on the two regions, the goal is to find the object pose and corresponding silhouette that best explains the segmentation of the image. The potential of this technique was already demonstrated by early approaches that treated segmentation and pose tracking as independent problems (Schmaltz et al. 2012; Brox et al. 2010; Rosenhahn et al. 2007). Dambreville et al. (2008) later combined the two processes in a single energy function, leading to improved tracking robustness. Building on this approach and including the pixel-wise posterior membership of Bibby and Reid (2008), Prisacariu and Reid (2012) developed *PWP3D*, a real-time-capable algorithm that uses a level-set pose embedding. It is the foundation of almost all state-of-the-art region-based methods.

Based on *PWP3D*, multiple algorithms were proposed that incorporate additional information, extend the segmentation model, or improve efficiency. For the combination of depth and region, Kehl et al. (2017) suggested an extension of the original energy function with a term that is based on the *Iterative Closest Point (ICP)* algorithm (Besl and McKay 1992). In a different approach, Ren et al. (2017) tightly coupled region and depth information in a probabilistic formulation that uses 3D signed distance functions. To consider object texture, combinations with direct optimization using pixel intensity values (Y. Liu et al. 2020; Zhong and L. Zhang 2019) or descriptor fields (F. Liu et al. 2021) were also proposed. Also, as discussed previously, multiple methods were designed that combine region and edge information (Seo et al. 2014; G. Wang et al. 2015; Huang et al. 2020; X. Sun et al. 2021). To improve occlusion handling, Zhong et al. (2020a) suggested the use of learning-based object segmentation, while Huang et al. (2022) utilized edge-based contour constraints. Finally, the incorporation of measurements from a mobile phone's inertial sensor was suggested by Prisacariu et al. (2015).

Another approach that potentially improves tracking is the extension of the segmentation model. For this, Zhao et al. (2014) extended the appearance model of *PWP3D* with a boundary term that considers spatial distribution regularities of pixels. Later, Hexner and Hagege (2016) proposed using local appearance models that were inspired by the localized contours of Lankton and Tannenbaum (2008). The idea was further improved by Tjaden et al. (2018) with the development of temporally consistent local color histograms. Also, Zhong et al. (2020b) proposed a method that introduces polar-based region partitioning and edge-based occlusion detection.

To improve the energy function's optimization, Zhao et al. (2014) suggested a particle-filter-like stochastic technique that initializes a subsequent damped Newton method. For better efficiency, a hierarchical rendering approach that uses the Levenberg-Marquardt algorithm was developed by Prisacariu et al. (2015). In another work, Tjaden et al. (2018) proposed the use of a Gauss-Newton method to improve convergence. Apart from optimization, another idea towards better efficiency is the utilization of simplified signed distance functions in the probabilistic segmentation model (Y. Liu et al. 2020). Also, Kehl et al. (2017) suggested employing precomputed contour points to represent the object's 3D geometry and calculating the energy function along rays.

2.6. Depth-based Tracking

Depth-based methods minimize the distance between the surface of a 3D model and measurements from a depth camera. While high-quality depth cameras only recently became affordable, algorithms that utilize this kind of information have a long history. One of the most prominent methods is the *Iterative Closest Point (ICP)* approach of Besl and McKay (1992). The algorithm registers two point sets by iteratively establishing correspondences between closest points and subsequently minimizing their distance. Starting from the original *ICP* algorithm, many variants have been developed.

According to a survey from Rusinkiewicz and Levoy (2001), as well as more recent work from Pomerleau et al. (2015), most algorithms share a similar structure. Typically, they only differ with respect to point selection, matching, rejection, weighting, error metrics, or optimization techniques. For point selection, examples include methods that use all points (Besl and McKay 1992), sample points uniformly (Turk and Levoy 1994), select random points (Masuda et al. 1996), or choose points at locations with high color intensity gradients (Weik 1997). To establish correspondences, different matching algorithms have been developed. In addition to closest point selection, methods search for points along the normal vector (Chen and Medioni 1992), project source points into the destination image (Blais and Levine 1995), or perform a search close to projected points (Dorai et al. 1998). After matching, point pairs that are regarded as outliers can be rejected. Typical examples include removing pairs for which the euclidean distance or the angle between normal vectors is above a certain threshold (Pulli 1999). Also, it is possible to only consider the best n percent (Pulli 1999), eliminate pairs that are not consistent with neighboring pairs (Dorai et al. 1998), or reject points on image boundaries (Turk and Levoy 1994). To weight individual pairs, one can again use the euclidean distance (Godin et al. 1994). Other options include the compatibility of surface normals and estimates for sensor noise (Rusinkiewicz and Levoy 2001). Finally, for optimization, the point-to-point (Besl and McKay 1992) and point-to-plane (Chen and Medioni 1992) error metrics are most popular. While the first directly uses the euclidean distance, the second calculates the distance along the normal vector. The optimization is then conducted by repeatedly establishing correspondences and minimizing the error. In addition, methods exist that extrapolate in the transform space (Besl and McKay 1992) or perturb initial conditions (D. A. Simon 1996).

Based on the *ICP* approach and its variants, different tracking algorithms have been developed. Because of high efficiency and good convergence properties, many methods combine projective data association (Blais and Levine 1995) and the point-to-plane error metric (Chen and Medioni 1992). One example that uses this combination is the tracker of Salas-Moreno et al. (2013), which is employed in the *SLAM++* algorithm. It is used to estimate both the poses of individual objects and the camera. In the method, the error is computed densely over all points, and the Huber norm is utilized to consider outliers. Projective data association and the point-to-plane error metric are also used in the approach of Pauwels et al. (2013), which densely combines depth and optical flow. Finally, Tan et al. (2017) and Kehl et al. (2017) also adopted the same combination. Both methods consider depth information using a sparse set of precomputed surface points and normal vectors. In addition, similar to their previous work (Tan and Ilic 2014), Tan et al. (2017) proposed the use of random forests to predict the relative pose.

Apart from algorithms based on *ICP*, methods that utilize signed distance functions have also been developed. According to the original formulation of Fitzgibbon (2003), the space around the object is thereby voxelized. Parameters that depend on the surface distance, such as gradient vectors, can be precomputed. Instead of calculating correspondences, depth measurements are simply projected into this voxel space. Prominent examples that use this technique include the method of Ren and Reid (2012) and the articulated tracking approach *DART* by Schmidt et al. (2015b).

Finally, approaches that employ filtering techniques instead of gradient-based optimization have also been developed. Examples include the method of Wüthrich et al. (2013), which uses a Rao-Blackwellised particle filter to estimate the object pose. The approach was later extended to articulated objects by Cifuentes et al. (2017) using a so-called *Coordinate Particle Filter* (Wüthrich et al. 2015). In another approach, Choi and Christensen (2013) proposed the combination of photometric and geometric features to determine the likelihood of each particle. Also, to make tracking fast and reliable even with a small number of particles, Krull et al. (2015) used the concept of 3D object coordinates. Furthermore, Issac et al. (2016) adopted robust Gaussian filtering (Wüthrich et al. 2016) to effectively consider outliers.

2.7. Deep Learning-based Tracking

While deep learning has proven highly successful for global 6DoF pose estimation (Haugaard and Buch 2022; Lipson et al. 2022; G. Wang et al. 2021; He et al. 2021; Labbé et al. 2020; Hodaň et al. 2020; Sundermeyer et al. 2018), deep learning-based tracking methods were only recently proposed. Many approaches are inspired by pose refinement and predict the relative pose between object renderings and subsequent images. One of the first methods that follows this approach was developed by Garon and Lalonde (2017). It regresses the pose based on color and depth data. To make predictions robust to visual ambiguity and symmetry, Manhardt et al. (2018) proposed a loss function that evaluates the alignment of object contours. At the same time, Y. Li et al. (2018) developed *DeepIM*. The method introduces both a new loss function and an untangled pose representation that is independent of the object's coordinate frame and size. Both methods only utilize red-green-blue (RGB) images. Later, Marougkas et al. (2020) developed a method that uses red-green-blue-depth (RGB-D) images and introduced multiple parallel soft spatial attention modules to handle background clutter and occlusions. Also, Wen et al. (2020) proposed *se(3)-TrackNet*, which employs Lie algebra to represent 3D orientations and a new loss function. The incorporation of inertial data from an inertial measurement unit (IMU) sensor was investigated by R. Ge and Loianno (2021).

In addition to those render-and-compare algorithms, Deng et al. (2021) developed *PoseRBPF*. It uses a Rao-Blackwellized particle filter that decouples rotation and translation and encodes the rotation using pose-representative latent codes (Sundermeyer et al. 2018). In another approach by Piga et al. (2021), deep learning was adopted to segment point clouds, and an *Unscented Kalman Filter (UKF)* was used to track objects based on depth information. Also, Zheng et al. (2022) proposed the incorporation of a prior that is based on the object's motion history into a temporally primed tracking framework. Finally, Piga et al. (2022) combined real-time optical flow with low-frequency deep learning-based instance segmentation and global 6DoF pose estimation. Using this combination, they synchronized predictions with a Kalman filtering approach and were able to provide results at high framerates.

2.8. Multi-body Tracking

In contrast to the previously provided survey on tracking techniques, the following section gives a detailed overview on multi-body object tracking. We thereby focus on gradient-based approaches and rigid models. Elastic objects such as human hands or bodies are only briefly discussed. While early methods (Hogg 1983; Gavrila and Davis 1996) employed search algorithms to determine the pose and configuration of objects, Lowe (1991) proposed the use of nonlinear least-squares optimization. He showed that the approach is both robust and efficient, and allows to determine the state of articulated objects for large frame-to-frame motion. Similar techniques were later adopted for model-based hand tracking (Rehg and Kanade 1994) and to recover human body configurations (Bregler and Malik 1998). Also, Nickels and Hutchinson (2001) proposed the tracking of articulated objects using feature points in an *Extended Kalman Filter (EKF)*. Finally, a more formal derivation motivated by the Lagrange-d'Alembert formulation in classical physics was presented by Comport et al. (2007). In all those methods, Jacobian matrices that describe the variation of individual points or measurements for a minimal set of parameters are derived.

In contrast to the use of a minimal parameterization, Drummond and Cipolla (2002) proposed an approach that adopts Lagrange multipliers in combination with a Lie group formalism. Their method first predicts the 6DoF pose variation of all rigid bodies independently and later ensures compatibility using velocity constraints. A detailed comparison of this *postimposed* approach compared to *preimposed* methods using Jacobians was presented by T. E. d. Campos et al. (2006). Their experiments demonstrate that results for the two techniques are identical. Also, they showed that while *postimposed* constraints are very efficient for simple kinematic chains and allow temporary constraints, *preimposed* methods scale better for more complex tree-like kinematic structures. In contrast to those mathematically exact constraints, approaches that minimize the Mahalanobis distance (Demirdjian et al. 2003) or that introduce soft revolute joints (Mündermann et al. 2007) were also proposed.

Based on the presented principles to model kinematic structures, various methods that employ different tracking techniques have been developed. Approaches that consider depth information using the *ICP* algorithm were proposed by Dewaele et al. (2004) and Pellegrini et al. (2008). Later, Brox et al. (2010) fused information from region fitting, dense optical flow, and *SIFT* features. Similarly, Krainin et al. (2011) combined *SIFT* features and dense color information with *ICP*-like depth measurements. Information from depth sensors and joint encoders was utilized by Klingensmith et al. (2013). Also, Bohg et al. (2014) used joint positions that are predicted using pixel-wise part classification on depth images. Similarly, Rauch et al. (2018) employed pixel-wise part classification but directly considered depth values instead of joint positions. Lately, an algorithm that fuses region and dense color information was presented by Y. Liu and Namiki (2021). With *DART*, Schmidt et al. (2015b) developed a general user-friendly method that extends 6DoF tracking with signed distance functions to articulated objects. Later, Schmidt et al. (2015a) further extended it to avoid interpenetration of bodies and to make use of contact information from sensors. The integration of *DART* with a tactile sensor was also studied by Izatt et al. (2017). A similarly general method that uses dense optical flow and *ICP*-like depth measurements was developed by Pauwels et al. (2014). With the extension to multiple movable cameras, the algorithm was later renamed to *SimTrack* (Pauwels and Kragic 2015). Except for *SimTrack*, which implements the *postimposed* constraints of Drummond and Cipolla (2002), most methods use *preimposed* formulations similar to the approach of Lowe (1991).

Apart from gradient-based methods, other optimization techniques were also adopted for the tracking of multi-body objects. In the work of Hebert et al. (2012), a *UKF* was used to fuse different sources of information. A method that employs a *Random Forest* to directly regress to joint angles was proposed by Widmaier et al. (2016). Also, Cifuentes et al. (2017) used the *Coordinate Particle Filter* (Wüthrich et al. 2015) to track robot manipulators, considering information from both depth images and joint measurements. In another approach, Zuo et al. (2019) employed deep learning to recognize 2D keypoints on a robot and use them to infer its configuration. Finally, Labbé et al. (2021) proposed *RoboPose*, a render-and-compare strategy to estimate the joint angles of a known robot from a single image. In addition to methods for rigid objects, a large amount of work on the tracking of human hands and bodies exists. While deep learning has become highly popular for those tasks (Zimmermann and Brox 2017; Kanazawa et al. 2018; L. Ge et al. 2019), some model-based methods still use the original formulation of Lowe to consider the underlying kinematic structure (Tan et al. 2016; Taylor et al. 2016; Han et al. 2020). However, because techniques like keypoint detection and regularization are highly optimized for their respective domains, there is no straightforward application of such algorithms to arbitrary multi-body systems.

3 Region-based Tracking

3.1. Introduction

In the following chapter, we focus on region-based tracking. In general, such techniques use image statistics to differentiate between a foreground region that corresponds to the object and a background region. Based on the two regions, the goal is to find the object pose and corresponding silhouette that best explains the segmentation of the image. The big advantage of such approaches is that they are able to reliably track a wide variety of objects in cluttered scenes, requiring only a monocular RGB camera and a textureless 3D object model. The only main assumption is that the object and background are distinguishable. As a consequence, region-based techniques are perfectly suited for textureless objects. Also, methods are typically robust to motion blur, making it possible to track fast-moving objects. However, while those properties are perfect for the tracking of textureless robots, most methods feature computationally expensive dense formulations that contradict our requirements for efficiency and speed.

In the following, we, therefore, present a sparse region-based approach that bridges this gap in efficiency. Our method considers image information sparsely along so-called correspondence lines that model the probability of the object's contour location. To further improve efficiency, a discrete scale-space formulation is developed. In addition, we introduce smoothed step functions that consider a defined global and local uncertainty and analyze their effect on the posterior probability. The joint posterior probability for the object pose is then described using a pre-rendered sparse viewpoint model. Subsequently, the function is maximized using Newton optimization with Tikhonov regularization. We thereby differentiate between global and local optimization, using a novel approximation for the first-order derivative employed in the Newton method. In multiple experiments on the *RBOT* (Tjaden et al. 2018) and *OPT* (Wu et al. 2017) datasets, it is demonstrated that the resulting algorithm improves the state of the art by a significant margin, both in terms of runtime and quality. An illustration of the tracking process with converging correspondence lines is shown in Fig. 3.1.

Note that the chapter is based on *SRT3D* (Stoiber et al. 2022a) and a corresponding preceding publication (Stoiber et al. 2020). In addition, it includes modifications proposed for *ICG* (Stoiber et al. 2022b), which include the re-scaling of posterior probability distributions and the handling of occlusions. All experiments were conducted with the latest implementation in the *M3T* tracking library.

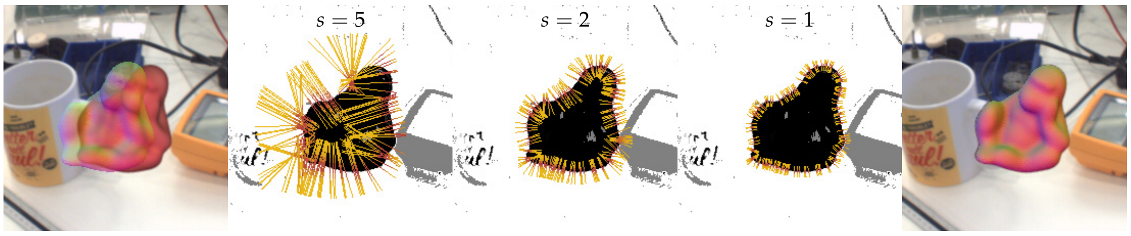


Figure 3.1.: Optimization process of the developed region-based method featuring the *Ape* object from the *RBOT* dataset (Tjaden et al. 2018). The image on the left shows a rendered overlay of the object model for the initial pose. The estimated pose after the optimization is visualized in the image on the right. The three illustrations in the middle show yellow correspondence lines for different scales s. High probabilities for the contour location are illustrated in red. Pixel-wise posterior probabilities that describe the probability that a pixel belongs to the background are encoded in grayscale images. Note that during tracking, pixel-wise posteriors are only calculated along correspondence lines.

3.2. Correspondence Line Model

The following section provides a detailed derivation of the probabilistic correspondence line model, which describes the probability of the object's contour location. We thereby start with a geometric definition of correspondence lines. This is followed by a probabilistic model that considers the segmentation of a correspondence line into foreground and background. To improve computational efficiency, we then extend this model and provide a discrete scale-space formulation. Finally, novel smoothed step functions are introduced, and their effect on the contour location's posterior probability is discussed.

3.2.1. Correspondence Lines

In contrast to most state-of-the-art algorithms, we do not consider image information densely over the entire image. Instead, inspired by *RAPID* (Harris and Stennett 1990), pixel values are processed sparsely along so-called correspondence lines. The name correspondence line is motivated by the term correspondence point used in *ICP* (Besl and McKay 1992). Similar to *ICP*, correspondences are first defined, and the optimization with respect to them is then conducted in a second step. While for *ICP*, individual 3D points are used as data, multiple pixel values along a line are considered in our case. A visualization of a single correspondence line is shown in Fig. 3.2.

Similar to the commonly used image definition $\boldsymbol{I}\colon \boldsymbol{\Omega} \to \{0,\dots,255\}^3$, we formally denote a correspondence line as a map $\boldsymbol{l} : \omega \to \{0,\dots,255\}^3$. In this notation, $\boldsymbol{\Omega} \subset \mathbb{R}^2$ describes the image domain while $\omega \subset \mathbb{R}$ is considered the correspondence line domain. Image values $\boldsymbol{y}$, which are typically accessed using the image coordinate $\boldsymbol{x} = \begin{bmatrix} x & y \end{bmatrix}^\top$

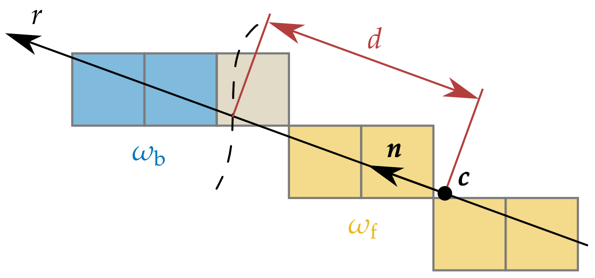


Figure 3.2.: Correspondence line in an image defined by a center $\boldsymbol{c}$ and a normal vector $\boldsymbol{n}$. The illustration shows pixels along the correspondence line as well as the foreground region ω_f in yellow and the background region ω_b in blue. The contour distance d points from the correspondence line center to an estimated contour, indicated by a dashed line.

and the image function $\boldsymbol{y} = \boldsymbol{I}(\boldsymbol{x})$, are described using the line coordinate r and the correspondence line function $\boldsymbol{y} = \boldsymbol{l}(r)$. Correspondence lines are located in the image and remain fixed once they have been established. The location and orientation of each correspondence line is defined by a center $\boldsymbol{c} = \begin{bmatrix} c_x & c_y \end{bmatrix}^\top \in \mathbb{R}^2$ in image coordinates and a normal vector $\boldsymbol{n} = \begin{bmatrix} n_x & n_y \end{bmatrix}^\top \in \mathbb{R}^2$, with $\|\boldsymbol{n}\|_2 = 1$. Using this definition, the relation between an image $\boldsymbol{I}$ and a correspondence line $\boldsymbol{l}$ is expressed as follows

$$\boldsymbol{l}(r) = \boldsymbol{I}(\boldsymbol{c} + r\boldsymbol{n}), \tag{3.1}$$

where image coordinates in $\boldsymbol{I}$ are rounded to the center of the next closest pixel.

3.2.2. Probabilistic Model

Inspired by the generative model of Bibby and Reid (2008), we derive a probabilistic model for the segmentation of a correspondence line into a foreground region ω_f and a background region ω_b. Note that this is the one-dimensional (1D) equivalent of the segmentation of a 2D image into the regions Ω_f and Ω_b. We assume that there is only one transition between foreground and background. The location of this transition relative to the correspondence line center $\boldsymbol{c}$ is described by the contour distance $d \in \mathbb{R}$, which is visualized in Fig. 3.2.

To derive the probabilistic model, we first describe the formation process for a single pixel on the correspondence line. The joint probability distribution is thereby written as

$$p(r, \boldsymbol{y}, d, m) = p(r \mid d, m)p(\boldsymbol{y} \mid m)p(m)p(d), \tag{3.2}$$

where $m \in \{m_\mathrm{f}, m_\mathrm{b}\}$ is the model parameter that can denote either foreground or background. If we condition this distribution on the image value $\boldsymbol{y}$, we obtain

$$p(r, d, m \mid \boldsymbol{y}) = p(r \mid d, m)p(m \mid \boldsymbol{y})p(d). \tag{3.3}$$

Following Bibby and Reid (2008), we use Bayes' theorem and the marginalization over m to calculate the pixel-wise posterior probability

$$p(m_i \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid m_i)p(m_i)}{\sum_{j\in\{\mathrm{f,b}\}} p(\boldsymbol{y} \mid m_j)p(m_j)}, \quad i \in \{\mathrm{f,b}\}, \tag{3.4}$$

where $p(\boldsymbol{y} \mid m_\mathrm{f})$ and $p(\boldsymbol{y} \mid m_\mathrm{b})$ are probability distributions that describe how likely it is that a specific color value is part of the foreground region or the background region, respectively. The two distributions can be estimated by calculating two color histograms, one over the foreground region and one over the background region. A detailed explanation of their computation is given in Section 3.4.2. Using the knowledge that foreground and background are equally likely along the correspondence line, i.e. $p(m_\mathrm{f}) = p(m_\mathrm{b})$, we obtain

$$p(m_i \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid m_i)}{p(\boldsymbol{y} \mid m_\mathrm{f}) + p(\boldsymbol{y} \mid m_\mathrm{b})}, \quad i \in \{\mathrm{f,b}\}. \tag{3.5}$$

Finally, based on Eq. (3.3), we are able to marginalize over m and condition on r to express the posterior probability for the contour distance d as

$$p(d \mid r,\boldsymbol{y}) = \frac{1}{p(r)} \sum_{i\in\{\mathrm{f,b}\}} p(r \mid d, m_i)p(m_i \mid \boldsymbol{y})p(d). \tag{3.6}$$

To calculate the posterior probability over the entire correspondence line domain ω, we assume pixel-wise independence and, based on Eq. (3.6), write

$$p(d \mid \omega,\boldsymbol{l}) \propto \prod_{r\in\omega} \sum_{i\in\{\mathrm{f,b}\}} p(r \mid d, m_i)p(m_i \mid \boldsymbol{l}(r)). \tag{3.7}$$

Note that $p(r)$ and $p(d)$ are considered to be uniform and constant and are therefore dropped. Also, while pixel-wise independence does not hold in general, it is a well-established approximation that allows us to avoid ill-defined assumptions for spatial regularities and is close enough to reality to yield good results. The conditional line coordinate probability $p(r \mid d, m)$ will be discussed in Section 3.2.4. Similar to the probabilistic model of Bibby and Reid (2008), which describes the probability of a shape kernel given information from an image, Eq. (3.7) provides the probability of the contour distance d given data from a correspondence line.

3.2.3. Discrete Scale-space Formulation

Estimating the distribution of posterior probabilities $p(d \mid \omega,\boldsymbol{l})$ is computationally expensive since, for each distance d, the product in Eq. (3.7) has to be computed over the entire domain ω. This results in quadratic complexity for the calculation of the entire distribution. In contrast, pixel-wise posterior probabilities $p(m \mid \boldsymbol{y})$ are used in the posterior probability calculation of multiple distances d, leading to linear complexity. Consequently, shifting computation from the calculation of the distribution to the

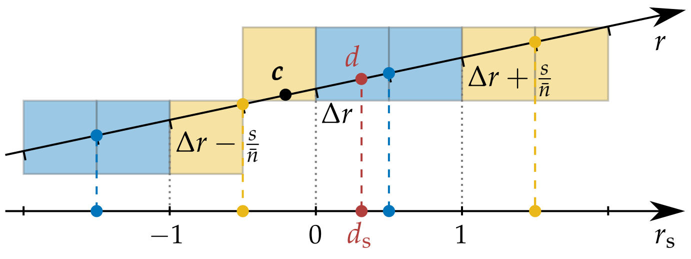


Figure 3.3.: Example of the relation between the unscaled space r along the correspondence line and the scale space r_s. Neighboring pixels that are combined into segments are visualized by the same color in blue or yellow. Blue and yellow dots indicate the center of each segment and the corresponding discretized value in the scale space. An example of the contour distance is illustrated in red. The offset Δr is chosen in a way that ensures that discretized values in the scale space are the same for all correspondence lines. In this example, Δr points to the closest edge between pixels.

calculation of pixel-wise posterior probabilities allows us to improve computational efficiency. Also, it is advantageous to normalize correspondence lines in a way that ensures that a line coordinate pointing to a segment center for one correspondence line points to a segment center for all correspondence lines. This uniformity can be used in the precalculation of values for the conditional line coordinate probability to further improve efficiency.

In the following, we thus develop a discrete scale-space formulation that combines multiple pixels into segments. In addition, the formulation projects from the continuous space along the correspondence line into a discrete space that is independent of a correspondence line's location and orientation. An illustration of this transformation is shown in Fig. 3.3. Both line coordinates and contour distances are projected as follows

$$r_\mathrm{s} = (r - \Delta r)\frac{\bar{n}}{s}, \tag{3.8}$$

$$d_\mathrm{s} = (d - \Delta r)\frac{\bar{n}}{s}, \tag{3.9}$$

with $s \in \mathbb{N}^+$ the scale that describes the number of pixels combined into a segment, $\bar{n} = \max(|n_x|, |n_y|)$ the major absolute normal component that projects a correspondence line to the closest horizontal or vertical image coordinate, and $\Delta r \in \mathbb{R}$ the offset from the correspondence line center $\boldsymbol{c}$ to a defined pixel location.

Based on Eq. (3.7), the posterior probability in the discrete scale space is calculated as

$$p(d_\mathrm{s} \mid \omega_\mathrm{s}, \boldsymbol{l}_\mathrm{s}) \propto \prod_{r_\mathrm{s} \in \omega_\mathrm{s}} \sum_{i \in \{\mathrm{f},\mathrm{b}\}} p(r_\mathrm{s} \mid d_\mathrm{s}, m_i) p(m_i \mid \boldsymbol{l}_\mathrm{s}(r_\mathrm{s})), \tag{3.10}$$

where ω_s is the scaled correspondence line domain and $\boldsymbol{s} = \boldsymbol{l}_\mathrm{s}(r_\mathrm{s})$ a set-valued function that maps from the scaled line coordinate r_s to the segment $\boldsymbol{s}$, which is a set of the closest

s pixel values $\boldsymbol{y}$. Similar to pixel-wise posteriors in Eq. (3.5) and assuming pixel-wise independence, segment-wise posteriors are defined as

$$p(m_i \mid \boldsymbol{s}) = \frac{\prod_{\boldsymbol{y} \in \boldsymbol{s}} p(\boldsymbol{y} \mid m_i)}{\prod_{\boldsymbol{y} \in \boldsymbol{s}} p(\boldsymbol{y} \mid m_\mathrm{f}) + \prod_{\boldsymbol{y} \in \boldsymbol{s}} p(\boldsymbol{y} \mid m_\mathrm{b})}, \quad i \in \{\mathrm{f}, \mathrm{b}\}. \tag{3.11}$$

The derived formulation allows to efficiently cover the correspondence line domain ω, using the scale parameter s to set the segment size and to adjust between accuracy and efficiency. In the following, we will again drop the index s for all variables to simplify the notation. Note, however, that all definitions and derivations are valid both for the original space and for the discrete scale-space formulation.

3.2.4. Smoothed Step Functions

To model the conditional probabilities of the line coordinate $p(r \mid d, m_\mathrm{f})$ and $p(r \mid d, m_\mathrm{b})$, most state-of-the-art algorithms (Zhong et al. 2020b; Tjaden et al. 2018) use the following smoothed step functions that are based on the arctangent

$$h_\mathrm{f}(x) = \frac{1}{2} - \frac{1}{\pi} \tan^{-1}\left(\frac{x}{s_\mathrm{h}}\right), \tag{3.12}$$

$$h_\mathrm{b}(x) = \frac{1}{2} + \frac{1}{\pi} \tan^{-1}\left(\frac{x}{s_\mathrm{h}}\right), \tag{3.13}$$

where $x = r - d$ describes the distance from the line coordinate r to the contour and the slope parameter $s_\mathrm{h} \in \mathbb{R}^+$ models a local uncertainty with respect to the exact location of the foreground and background transition. Considering the plots of those functions in Fig. 3.4, one notices that the functions quickly converge towards either zero or one for increasing absolute values of $x = r - d$. Except for a small area around zero, the functions assume that, given the model m and the contour distance d, one knows perfectly on which side of the contour the line coordinate r lies. In the following, we will argue that for real-world applications, this assumption is wrong.

While the pixel-wise posterior probability in Eq. (3.5) provides very good predictions for the model m, it is still an imperfect simplification of the real world. Typical effects that are not considered by the statistical model are image noise or fast appearance changes that can lead to pixel colors that are not yet present in the color histograms. Another effect originates from pixels that are wrongly classified due to imperfect segmentation and that are then assigned to the wrong color histograms. Finally, there also remains the question if a statistical model that purely relies on pixel colors is sufficient to capture all the statistical effects in the real world and is able to perfectly predict the model m.

To take those limitations into account and consider a constant global uncertainty in

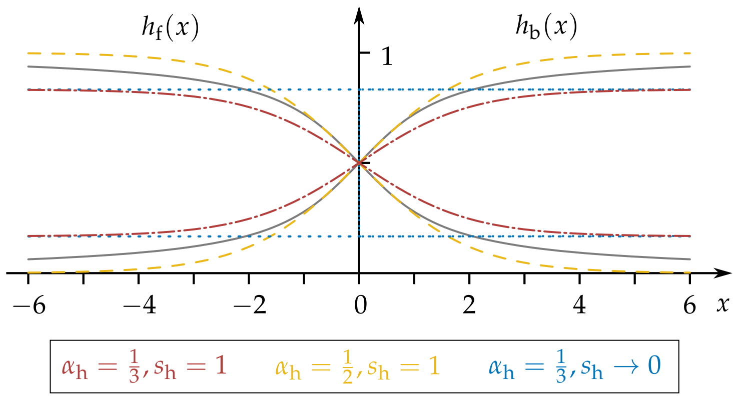


Figure 3.4.: Smoothed step functions h_f and h_b that model the conditional line coordinate probabilities $p(r \mid d, m_f)$ and $p(r \mid d, m_b)$. The functions from Eqs. (3.12) and (3.13) used by Zhong et al. (2020b) and Tjaden et al. (2018) are illustrated by gray lines for a slope parameter $s_h = 1$. For the proposed functions in Eqs. (3.14) and (3.15), dash-dotted red lines correspond to $\alpha_h = \frac{1}{3}$ and $s_h = 1$. Also, dotted blue lines show the functions for $\alpha_h = \frac{1}{3}$ and $s_h \to 0$, while dashed yellow lines illustrate the proposed functions for $\alpha_h = \frac{1}{2}$ and $s_h = 1$.

addition to local uncertainty, we propose the following functions

$$h_f(x) = \frac{1}{2} - \alpha_h \tanh\left(\frac{x}{2s_h}\right), \tag{3.14}$$

$$h_b(x) = \frac{1}{2} + \alpha_h \tanh\left(\frac{x}{2s_h}\right), \tag{3.15}$$

with the amplitude parameter $\alpha_h \in [0, 0.5]$ and the slope parameter $s_h \in \mathbb{R}^+$. Note that we also replace the arctangent with the hyperbolic tangent function. In the next section, we will prove that, under some conditions, this results in a Gaussian distribution for the posterior probability. Examples of the proposed functions are shown in Fig. 3.4.

In addition to viewing α_h as a simple amplitude parameter, we are able to demonstrate that there is also another interpretation. For this, we assume that the model m is extended with a third class m_n. This class considers external effects that are independent of the foreground and background model m_f and m_b. For this scenario, we can show that $p(m_f) = p(m_b) = \alpha_h$ and that $p(m_n) = 1 - 2\alpha_h$. Following this interpretation, the amplitude parameter, therefore, allows us to set the probability that a pixel's color is generated by the foreground or background model in contrast to some other effects that are considered as noise. This again shows that the amplitude parameter α_h is able to model a constant, global uncertainty. Note that in this scenario, smoothed step functions with $\alpha_h = \frac{1}{2}$ are used, and a constant function $p(r \mid d, m_n) = \frac{1}{2}$ is adopted for the noise model. A detailed derivation of this extended model and a proof of its equivalence to the use of the functions in Eqs. (3.14) and (3.15) is given in Appendix A.

3.2.5. Posterior Probability Distribution

Given the smoothed step functions h_f and h_b that model the conditional line coordinate probabilities $p(r \mid d, m_\mathrm{f})$ and $p(r \mid d, m_\mathrm{b})$, the final expression of the posterior probability distribution from Eq. (3.7) can be written as

$$p(d \mid \omega, \boldsymbol{l}) \propto \prod_{r \in \omega} h_\mathrm{f}(r-d) p_\mathrm{f}(r) + h_\mathrm{b}(r-d) p_\mathrm{b}(r), \tag{3.16}$$

with the abbreviations $p_\mathrm{f}(r) = p(m_\mathrm{f} \mid \boldsymbol{l}(r))$ and $p_\mathrm{b}(r) = p(m_\mathrm{b} \mid \boldsymbol{l}(r))$. In the following, we provide a detailed analysis to understand how the slope parameter s_h and the amplitude parameter α_h affect this distribution. We thereby assume a contour at the correspondence line center and step functions for the pixel-wise posteriors p_f and p_b. Note that the assumption of step functions corresponds well with real-world experiments that show that, in most cases, there is a distinct split between foreground and background.

For the analysis, we write the posterior probability distribution in continuous form for an infinite correspondence line and infinitesimally small pixels

$$p(d \mid \omega, \boldsymbol{l}) \propto \prod_{r=-\infty}^{\infty} \big(h_\mathrm{f}(r-d) p_\mathrm{f}(r) + h_\mathrm{b}(r-d) p_\mathrm{b}(r) \big)^{\mathrm{d}r}. \tag{3.17}$$

We then start with the calculation of the first-order derivative of the logarithmic posterior with respect to the contour distance d. Based on a detailed derivation given in Appendix B, the obtained closed-form solution is

$$\frac{\partial \ln \big(p(d \mid \omega, \boldsymbol{l}) \big)}{\partial d} = -2 \tanh^{-1} \left(2\alpha_\mathrm{h} \tanh \left(\frac{d}{2 s_\mathrm{h}} \right) \right). \tag{3.18}$$

A visualization of this function for different slope and amplitude parameters α_h and s_h is given in Fig. 3.5. The plot shows that the amplitude parameter α_h controls not only the amplitude of h_f and h_b but also the amplitude of the first-order derivative. For $\alpha_\mathrm{h} = \frac{1}{2}$, the first-order derivative converges to a linear function. At the same time, the parameter s_h affects both the slope of h_f and h_b and the slope of the first-order derivative. For $s_\mathrm{h} \to 0$ it leads to a perfect step function.

For the two edge cases with $\alpha_\mathrm{h} = \frac{1}{2}$ and $s_\mathrm{h} \to 0$, Eq. (3.18) can be simplified, and we are able to calculate a closed-form solution for the posterior probability distribution. In the case of $\alpha_\mathrm{h} = \frac{1}{2}$, the posterior probability distribution results in a perfect Gaussian

$$p(d \mid \omega, \boldsymbol{l}) = \frac{1}{\sqrt{2\pi s_\mathrm{h}}} \exp \left(-\frac{d^2}{2 s_\mathrm{h}} \right), \tag{3.19}$$

where the slope parameter s_h is equal to the variance. In the case of $s_\mathrm{h} \to 0$, which leads to sharp step functions for h_f and h_b, the posterior probability distribution becomes a perfect Laplace distribution

$$p(d \mid \omega, \boldsymbol{l}) = \frac{1}{2b} \exp \left(-\frac{|d|}{b} \right), \quad b = \frac{1}{2 \tanh^{-1}(2\alpha_\mathrm{h})}, \tag{3.20}$$

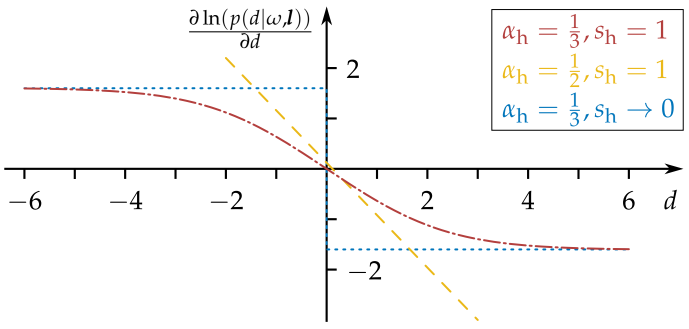


Figure 3.5.: First-order derivatives of the log-posterior with respect to the contour distance d for different slope and amplitude parameters s_h and α_h. The dash-dotted red line shows the derivative for $\alpha_h = \frac{1}{3}$ and $s_h = 1$, which yields a function with a smooth transition from an upper bound to a lower bound. The dashed yellow line shows the function for $\alpha_h = \frac{1}{2}$ and $s_h = 1$. This produces a linear first-order derivative. Finally, using $\alpha_h = \frac{1}{3}$ and $s_h \to 0$ results in a perfect step function illustrated by the dotted blue line.

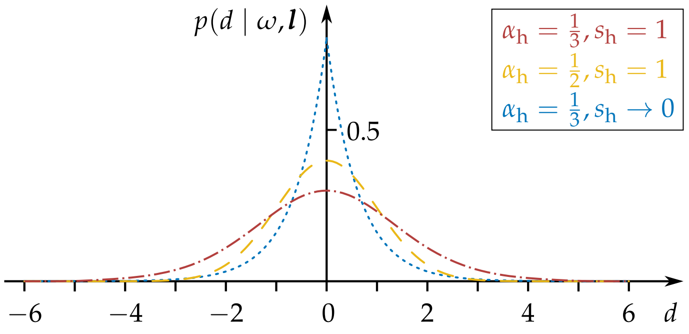


Figure 3.6.: Posterior probability distributions for different slope and amplitude parameters s_h and α_h. The dash-dotted red line shows the function for $\alpha_h = \frac{1}{3}$ and $s_h = 1$, which leads to a very flat distribution. Note that the function was computed numerically. Using $\alpha_h = \frac{1}{2}$ and $s_h = 1$ results in a perfect Gaussian distribution shown by the dashed yellow line. The parameters $\alpha_h = \frac{1}{3}$ and $s_h \to 0$, on the other hand, yield a perfect Laplace distribution for the posterior probability that is illustrated by a dotted blue line.

where $b \in \mathbb{R}^+$ is the scale parameter of the Laplace distribution that depends on α_h. A detailed derivation of the two functions is provided in Appendix C. Examples for both distributions, as well as a mixed posterior distribution with $s_h = 1$ and $\alpha_h = \frac{1}{3}$, are visualized in Fig. 3.6. The plot shows that while the Laplace distribution has a pronounced peak, the Gaussian distribution has a smoothed maximum for which nearby values have similarly high probabilities. This coincides with our intuition that the slope parameter s_h controls local uncertainty, allowing multiple values d to be almost equally likely. At the same time, the amplitude parameter α_h controls the size of the peak compared to its surroundings, effectively controlling global uncertainty. Combining the two parameters in a mixed distribution results in a function that is able to consider both local and global uncertainty simultaneously. Given the detailed knowledge about correspondence lines and the posterior probability distribution, we are now able to develop a sparse approach to region-based 3D object tracking.

3.3. 3D Object Tracking

In this section, we first define basic mathematical concepts. This is followed by the description of a sparse viewpoint model, which avoids the rendering of the 3D model during tracking. By combining this representation of the geometry with the correspondence line model developed in the previous section, we are able to formulate a joint posterior probability with respect to the object pose. This probability function is then maximized using Newton optimization with Tikhonov regularization. Finally, we define the required gradient vector and Hessian matrix for the Newton method. We thereby differentiate between global and local optimization to ensure both fast convergence and high accuracy.

3.3.1. Preliminaries

In the following work, we define 3D model points as $\boldsymbol{X} = \begin{bmatrix} X & Y & Z \end{bmatrix}^\top \in \mathbb{R}^3$ and use the tilde notation to write the homogeneous form $\widetilde{\boldsymbol{X}} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^\top$. For the projection of a 3D model point $\boldsymbol{X}$ into the image space, we assume an undistorted image and use the pinhole camera model

$$\boldsymbol{x} = \boldsymbol{\pi}(\boldsymbol{X}) = \begin{bmatrix} \frac{X}{Z} f_x + p_x \\ \frac{Y}{Z} f_y + p_y \end{bmatrix}, \tag{3.21}$$

with f_x and f_y the focal lengths and p_x and p_y the principal point coordinates given in units of pixels. The inverse operation, which is the reconstruction of a 3D model point from an image coordinate $\boldsymbol{x}$ and corresponding depth value d_Z along the optical axis, can be written as

$$\boldsymbol{X} = \boldsymbol{\pi}^{-1}(\boldsymbol{x}, d_Z) = d_Z \begin{bmatrix} \frac{x - p_x}{f_x} \\ \frac{y - p_y}{f_y} \\ 1 \end{bmatrix}. \tag{3.22}$$

To describe the relative pose between the model reference frame M and the camera reference frame C, we use the homogeneous matrix ${}_{\mathrm{C}}\boldsymbol{T}_{\mathrm{M}} \in \mathbb{SE}(3)$. For the transformation of a 3D model point, we can then write

$${}_{\mathrm{C}}\widetilde{\boldsymbol{X}} = {}_{\mathrm{C}}\boldsymbol{T}_{\mathrm{M}}\, {}_{\mathrm{M}}\widetilde{\boldsymbol{X}} = \begin{bmatrix} {}_{\mathrm{C}}\boldsymbol{R}_{\mathrm{M}} & {}_{\mathrm{C}}\boldsymbol{t}_{\mathrm{M}} \\ \boldsymbol{0} & 1 \end{bmatrix} {}_{\mathrm{M}}\widetilde{\boldsymbol{X}}, \tag{3.23}$$

where ${}_{\mathrm{C}}\widetilde{\boldsymbol{X}}$ and ${}_{\mathrm{M}}\widetilde{\boldsymbol{X}}$ are 3D model points written in the camera reference frame C and the model reference frame M, respectively, and where ${}_{\mathrm{C}}\boldsymbol{R}_{\mathrm{M}} \in \mathbb{SO}(3)$ and ${}_{\mathrm{C}}\boldsymbol{t}_{\mathrm{M}} \in \mathbb{R}^3$ are the

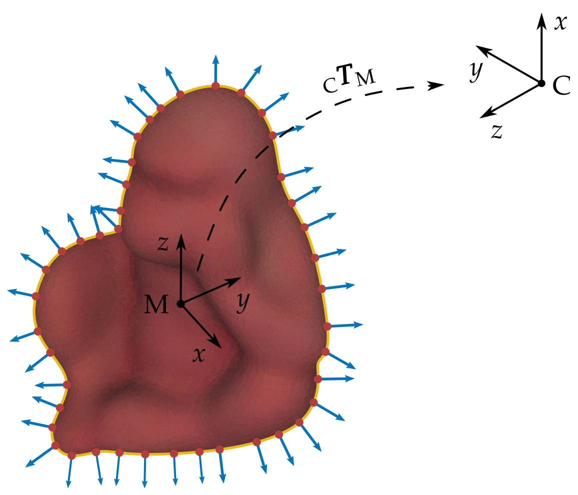


Figure 3.7.: Illustration of a 2D rendering computed from a 3D mesh model. The model reference frame M is shown at the center of the object, while a camera reference frame C is shown at the upper right corner of the image. The transformation from the model to the camera reference frame that is described by ${}_{\mathrm{C}}\boldsymbol{T}_{\mathrm{M}}$ is indicated by a dashed arrow. The contour of the rendered model is highlighted by a yellow line. Red points and blue arrows illustrate 2D contour points and approximated normal vectors.

rotation matrix and the translation vector that define the transformation from M to C. An illustration of the two reference frames and a homogeneous transformation matrix is given in Fig. 3.7.

For small variations, the axis-angle representation, which is a minimal representation, is used. With the exponential map, the rotation matrix writes as

$$\boldsymbol{R} = \exp([\boldsymbol{r}]_\times) = \boldsymbol{I} + [\boldsymbol{r}]_\times + \frac{1}{2!}[\boldsymbol{r}]_\times^2 + \frac{1}{3!}[\boldsymbol{r}]_\times^3 + ..., \tag{3.24}$$

where $[\boldsymbol{r}]_\times$ is the skew-symmetric cross-product matrix of $\boldsymbol{r} \in \mathbb{R}^3$. By neglecting higher-order terms of the series expansion, Eq. (3.24) can be linearized. We are then able to write the linear variation of a 3D model point in the camera reference frame C as

$${}_{\mathrm{C}}\widetilde{\boldsymbol{X}}(\boldsymbol{\theta}) = \begin{bmatrix} {}_{\mathrm{C}}\boldsymbol{R}_{\mathrm{M}} & {}_{\mathrm{C}}\boldsymbol{t}_{\mathrm{M}} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{I} + [\boldsymbol{\theta}_{\mathrm{r}}]_\times & \boldsymbol{\theta}_{\mathrm{t}} \\ \boldsymbol{0} & 1 \end{bmatrix} {}_{\mathrm{M}}\widetilde{\boldsymbol{X}}, \tag{3.25}$$

with the rotational variation $\boldsymbol{\theta}_{\mathrm{r}} \in \mathbb{R}^3$, the translational variation $\boldsymbol{\theta}_{\mathrm{t}} \in \mathbb{R}^3$, and the full variation vector $\boldsymbol{\theta}^\top = \begin{bmatrix} \boldsymbol{\theta}_{\mathrm{r}}^\top & \boldsymbol{\theta}_{\mathrm{t}}^\top \end{bmatrix}$. Note that, since the object is typically moved significantly more than the camera, it is more natural to variate 3D points in the model reference frame M instead of the camera reference frame C. Also, the variation in the model reference frame has the advantage that a simple extension of the algorithm to multiple cameras is possible.

3.3.2. Sparse Viewpoint Model

In contrast to most state-of-the-art methods, we do not use the 3D geometry in the form of a mesh model directly. Instead, similar to Tan et al. (2017), we employ a representation that we call a sparse viewpoint model. To create this model, the 3D geometry is rendered from a number of n_v viewpoints all around the object. Virtual cameras are thereby placed on the vertices of a geodesic grid that surrounds the object. For each rendering, n_c points $\boldsymbol{x}_i \in \mathbb{R}^2$ are randomly sampled from the contour of the model. Subsequently, the vectors $\boldsymbol{n}_i \in \mathbb{R}^2$ that are normal to the contour are approximated for each point. Note that $\|\boldsymbol{n}_i\|_2 = 1$. An illustration of a rendering with sampled 2D contour points and normal vectors is shown in Fig. 3.7. Based on those 2D entities, 3D vectors with respect to the model reference frame are then reconstructed as follows

$$ {}_\mathrm{M}\widetilde{\boldsymbol{X}}_i = {}_\mathrm{M}\boldsymbol{T}_\mathrm{C}\, \widetilde{\boldsymbol{\pi}}^{-1}(\boldsymbol{x}_i, d_{Zi}), \tag{3.26} $$

$$ {}_\mathrm{M}\boldsymbol{N}_i = {}_\mathrm{M}\boldsymbol{R}_\mathrm{C} \begin{bmatrix} \boldsymbol{n}_i \\ 0 \end{bmatrix}, \tag{3.27} $$

where the tilde notation in $\widetilde{\boldsymbol{\pi}}^{-1}$ indicates that the 3D model point is returned in homogeneous form based on Eq. (3.22) and d_{Zi} is the depth value from the rendering. Note that in this case, C denotes the reference frame of the virtual camera from which the rendering was created. In addition to those vectors, we also compute the orientation vector ${}_\mathrm{M}\boldsymbol{v} = {}_\mathrm{M}\boldsymbol{R}_\mathrm{C}\, \boldsymbol{e}_Z$ that points from the camera to the model center, where $\boldsymbol{e}_Z = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$. The computed 3D model points, normal vectors, and the orientation vector are then stored for each view.

The sparse viewpoint model allows for a highly-efficient representation of the model contour. Given a specific pose with ${}_\mathrm{M}\boldsymbol{R}_\mathrm{C}$ and ${}_\mathrm{C}\boldsymbol{t}_\mathrm{M}$, the process of rendering the model and computing the contour reduces to a simple search for the closest precomputed view

$$ i_\mathrm{v} = \underset{i \in \{1,\dots,n_\mathrm{v}\}}{\arg\max} \left({}_\mathrm{M}\boldsymbol{v}_i^\top\, {}_\mathrm{M}\boldsymbol{R}_\mathrm{C}\, {}_\mathrm{C}\boldsymbol{t}_\mathrm{M} \right), \tag{3.28} $$

and the subsequent projection of the corresponding 3D model points and normal vectors into the image. Note that this high efficiency is especially important during the optimization of the joint posterior probability, where the pose changes in each iteration.

3.3.3. Joint Posterior Probability

In the following, we combine the correspondence line model from Section 3.2 with the developed sparse viewpoint model to define a joint posterior probability with respect to the pose variation. However, before probabilities can be calculated, the location and orientation of correspondence lines need to be defined. For this, 3D model points and normal vectors from the closest view of the sparse viewpoint model are projected into the image using the following equations

$$\boldsymbol{c}_i = \boldsymbol{\pi}\big({}_{\mathrm{C}}\boldsymbol{T}_{\mathrm{M}}\,{}_{\mathrm{M}}\widetilde{\boldsymbol{X}}_i\big), \tag{3.29}$$

$$\boldsymbol{n}_i \propto \big({}_{\mathrm{C}}\boldsymbol{R}_{\mathrm{M}}\,{}_{\mathrm{M}}\boldsymbol{N}_i\big)_{2\times 1}, \tag{3.30}$$

where the normal vector $\boldsymbol{n}_i$ is normalized to $\|\boldsymbol{n}_i\|_2 = 1$ and $()_{2\times1}$ denotes the first two elements of a vector.

Once all correspondence lines have been defined, we are able to variate the current pose and calculate contour distances d_i with respect to the pose variation vector $\boldsymbol{\theta}$. Contour distances are thereby calculated as the distances along normal vectors $\boldsymbol{n}_i$ from correspondence line centers $\boldsymbol{c}_i$ to projected 3D model points $\boldsymbol{X}_i$

$$d_i(\boldsymbol{\theta}) = \boldsymbol{n}_i^\top\big(\boldsymbol{\pi}({}_{\mathrm{C}}\boldsymbol{X}_i(\boldsymbol{\theta})) - \boldsymbol{c}_i\big). \tag{3.31}$$

Also, the same 3D model points $\boldsymbol{X}_i$ are used as in the definition of correspondence lines. Note, however that 3D model points ${}_{\mathrm{C}}\boldsymbol{X}_i$ and contour distances d_i also depend on the current pose estimate ${}_{\mathrm{C}}\boldsymbol{T}_{\mathrm{M}}$, which might be different from the pose that was used to define correspondence lines. An example of multiple correspondence lines with variated contour distances is shown in Fig. 3.8.

Finally, assuming a number of n_{c} independent correspondence lines and using the discrete scale-space formulation from Section 3.2.3, the joint posterior probability can be calculated as

$$p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{D}}) \propto \prod_{i=1}^{n_{\mathrm{c}}} p\big(d_{\mathrm{s}i}(\boldsymbol{\theta}) \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i}\big)^{\frac{s_{\mathrm{h}} s^2}{\sigma_{\mathrm{r}}{}^2 \bar{n}_i^2}}, \tag{3.32}$$

where $\boldsymbol{\mathcal{D}}$ describes the data from all correspondence lines. Note that the transformation of contour distances d_i from the original space to the discrete scale space is given by Eq. (3.9). In the proposed formulation, we make individual correspondence lines independent of scale and slope parameters. For this, the derivations from Section 3.2.5 are used, which show that, for the Gaussian case, the variance of the posterior probability distribution is equal to the slope parameter s_{h}. Following the interpretation that s_{h} corresponds to a variance in scale space, the unscaled variance in units of pixels is $\sigma^2 = {}^{s_{\mathrm{h}} s^2}/_{\bar{n}_i^2}$. Finally, to define our own standard deviation σ_{r} that is independent of those parameters, we have to scale the posterior probability in scale space with ${}^{s_{\mathrm{h}} s^2}/_{\sigma_{\mathrm{r}}{}^2 \bar{n}_i^2}$. The user-defined parameter σ_{r} thereby gives us the possibility to define the confidence for each iteration of the optimization, as well as with respect to other modalities. In summary, the developed formulation describes how well the current pose estimate explains the segmentation of the image into a foreground region, which corresponds to the object, and a background region.

3.3.4. Optimization

To maximize the joint posterior probability, we estimate the variation vector $\hat{\boldsymbol{\theta}}$ and iteratively update the pose. For a single iteration, the variation vector is calculated using

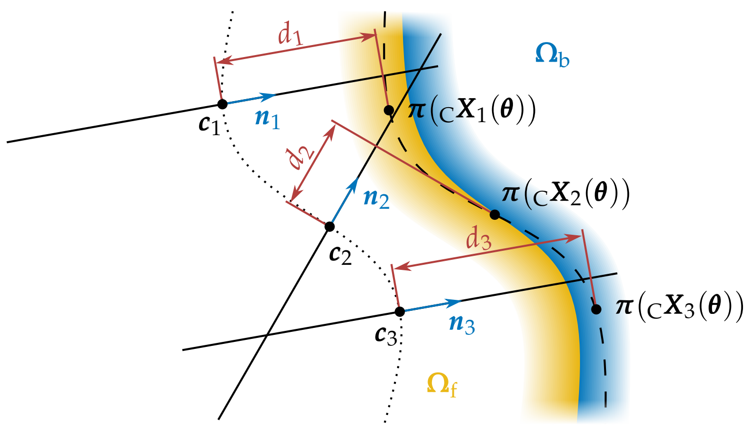


Figure 3.8.: Correspondence lines defined by a center $\boldsymbol{c}_i$ and a normal vector $\boldsymbol{n}_i$. Variated contour distances d_i are measured along the correspondence lines from the centers $\boldsymbol{c}_i$ to the projected 3D model points $_C\boldsymbol{X}_i$ that depend all on the same pose variation $\boldsymbol{\theta}$. The object contour of the original pose estimate, which was used to define the correspondence lines, is indicated by a dotted line. The current estimate of the contour that depends on the pose variation vector $\boldsymbol{\theta}$ is shown by a dashed line. The ground truth segmentation that we try to estimate is given by the foreground region $\boldsymbol{\Omega}_\mathrm{f}$ in yellow and the background region $\boldsymbol{\Omega}_\mathrm{b}$ in blue. Note that while those contours are illustrated as continuous lines, in our method, they are represented by points and normal vectors from the closest view of the sparse viewpoint model.

the Newton method with Tikhonov regularization

$$\hat{\boldsymbol{\theta}} = \left(-\boldsymbol{H} + \begin{bmatrix} \lambda_\mathrm{r}\boldsymbol{I}_3 & \boldsymbol{0} \\ \boldsymbol{0} & \lambda_\mathrm{t}\boldsymbol{I}_3 \end{bmatrix} \right)^{-1} \boldsymbol{g}, \tag{3.33}$$

where $\boldsymbol{g}$ is the gradient vector, $\boldsymbol{H}$ is the Hessian matrix, $\boldsymbol{I}_3$ the 3×3 identity matrix, and λ_r and λ_t are the regularization parameters for rotation and translation, respectively. The gradient vector and the Hessian matrix are defined as the first- and second-order derivatives of the joint log-posterior

$$\boldsymbol{g}^\top = \frac{\partial}{\partial \boldsymbol{\theta}} \ln \big(p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{D}}) \big) \Big|_{\boldsymbol{\theta}=\boldsymbol{0}}, \tag{3.34}$$

$$\boldsymbol{H} = \frac{\partial^2}{\partial \boldsymbol{\theta}^2} \ln \big(p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{D}}) \big) \Big|_{\boldsymbol{\theta}=\boldsymbol{0}}. \tag{3.35}$$

Using the logarithm has the advantage that scaling terms vanish and products turn into summations. Note that the Hessian represents the curvature of the distribution at a specific location, which for Gaussian probability functions is constant and directly corresponds to the negative inverse variance. Given this probabilistic interpretation, it can be argued that regularization parameters correspond to a prior probability. This prior controls how much we believe in the previous pose compared to the current estimate described by the gradient and Hessian. Consequently, for directions in which the Hessian indicates high uncertainty, the regularization helps to keep the optimization stable and to avoid pose changes that are not supported by sufficient data.

Finally, given a robust estimate for the variation vector, the predicted pose can be updated as follows

$$ {}_{\mathrm{C}}\boldsymbol{T}_{\mathrm{M}}^{+} = {}_{\mathrm{C}}\boldsymbol{T}_{\mathrm{M}} \begin{bmatrix} \exp([\hat{\boldsymbol{\theta}}_{\mathrm{r}}]_{\times}) & \hat{\boldsymbol{\theta}}_{\mathrm{t}} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{3.36} $$

Because of the exponential map, no orthonormalization is necessary. By iteratively repeating this process, we are able to optimize towards the pose that best explains the segmentation of the image.

3.3.5. Gradient and Hessian Approximation

In the following, the gradient vector and the Hessian matrix are approximated in a way that ensures both fast convergence and high accuracy. Using the chain rule, we write

$$ \boldsymbol{g}^{\top} = \sum_{i=1}^{n} \frac{s_{\mathrm{h}} s^2}{\sigma_{\mathrm{r}}{}^2 \bar{n}_i^2} \frac{\partial \ln\big(p(d_{\mathrm{s}i} \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})\big)}{\partial d_{\mathrm{s}i}} \frac{\partial d_{\mathrm{s}i}}{\partial {}_{\mathrm{C}}\boldsymbol{X}_i} \frac{\partial {}_{\mathrm{C}}\boldsymbol{X}_i}{\partial \boldsymbol{\theta}} \bigg|_{\boldsymbol{\theta}=\mathbf{0}}, \tag{3.37} $$

$$ \boldsymbol{H} \approx \sum_{i=1}^{n} \frac{s_{\mathrm{h}} s^2}{\sigma_{\mathrm{r}}{}^2 \bar{n}_i^2} \frac{\partial^2 \ln\big(p(d_{\mathrm{s}i} \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})\big)}{\partial d_{\mathrm{s}i}{}^2} \left(\frac{\partial d_{\mathrm{s}i}}{\partial {}_{\mathrm{C}}\boldsymbol{X}_i} \frac{\partial {}_{\mathrm{C}}\boldsymbol{X}_i}{\partial \boldsymbol{\theta}} \right)^{\top} \left(\frac{\partial d_{\mathrm{s}i}}{\partial {}_{\mathrm{C}}\boldsymbol{X}_i} \frac{\partial {}_{\mathrm{C}}\boldsymbol{X}_i}{\partial \boldsymbol{\theta}} \right) \bigg|_{\boldsymbol{\theta}=\mathbf{0}}. \tag{3.38} $$

Note that for the Hessian matrix, second-order partial derivatives with respect to $d_{\mathrm{s}i}$ and ${}_{\mathrm{C}}\boldsymbol{X}_i$ are neglected. Resulting errors are left to the iterative nature of the optimization. Using Eq. (3.25), the first-order derivative of the 3D model point ${}_{\mathrm{C}}\boldsymbol{X}_i$ is calculated as

$$ \frac{\partial {}_{\mathrm{C}}\boldsymbol{X}_i}{\partial \boldsymbol{\theta}} = {}_{\mathrm{C}}\boldsymbol{R}_{\mathrm{M}} \begin{bmatrix} -[{}_{\mathrm{M}}\boldsymbol{X}_i]_{\times} & \boldsymbol{I}_3 \end{bmatrix}. \tag{3.39} $$

With respect to the scaled contour distance $d_{\mathrm{s}i}$, both Eqs. (3.9) and (3.31) are used to write

$$ \frac{\partial d_{\mathrm{s}i}}{\partial {}_{\mathrm{C}}\boldsymbol{X}_i} = \frac{\bar{n}_i}{s} \frac{1}{{}_{\mathrm{C}}Z_i^2} \begin{bmatrix} {}_{\mathrm{C}}Z_i n_{xi} f_x & {}_{\mathrm{C}}Z_i n_{yi} f_y & -{}_{\mathrm{C}}X_i n_{xi} f_x - {}_{\mathrm{C}}Y_i n_{yi} f_y \end{bmatrix}. \tag{3.40} $$

For the calculation of the required first- and second-order derivatives of the log-posterior, we differentiate between global and local optimization.

In the case of global optimization, the posterior probability distribution of individual correspondence lines is approximated by a normal distribution $\mathcal{N}(d_{\mathrm{s}i} \mid \mu_i, \sigma_i{}^2)$. The required mean and standard deviation μ_i and σ_i are thereby estimated from a set of discretized contour distances $d_{\mathrm{s}i}$ and their corresponding probability values. An example of the approximation of a discrete posterior probability distribution is shown in Fig. 3.9. Based on the normal distribution, the first- and second-order derivatives are calculated

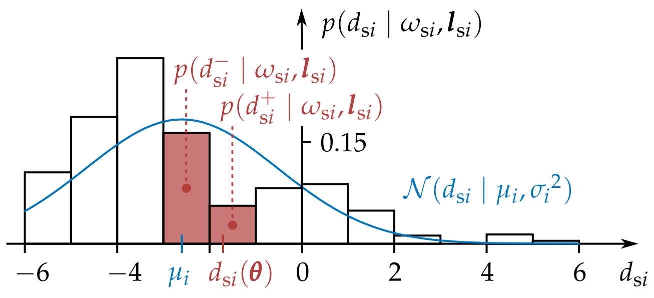


Figure 3.9.: Discrete posterior probability distribution with noisy measurements. For global optimization, the distribution is approximated by a normal distribution $\mathcal{N}(d_{si} \mid \mu_i, \sigma_i{}^2)$. The normal distribution and its mean μ_i are illustrated in blue. In the case of local optimization, only two discrete probability values that are closest to the current estimate of the contour distance $d_{si}(\boldsymbol{\theta})$ are considered. The two discrete probability values $p(d_{si}^- \mid \omega_{si}, \boldsymbol{l}_{si})$ and $p(d_{si}^+ \mid \omega_{si}, \boldsymbol{l}_{si})$, which are used to approximate the first-order derivative, are colored in red.

as follows

$$\frac{\partial \ln\big(p(d_{si} \mid \omega_{si}, \boldsymbol{l}_{si})\big)}{\partial d_{si}} \approx -\frac{1}{\sigma_i{}^2}(d_{si} - \mu_i), \tag{3.41}$$

$$\frac{\partial^2 \ln\big(p(d_{si} \mid \omega_{si}, \boldsymbol{l}_{si})\big)}{\partial {d_{si}}^2} \approx -\frac{1}{\sigma_i{}^2}. \tag{3.42}$$

The approximated derivatives direct the optimization towards the mean μ_i, using the variance $\sigma_i{}^2$ to consider uncertainty. Note that while in the real world, the mean does not exactly coincide with the maximum, it is typically quite close. At the same time, using the approximation has the advantage of fast convergence, and that the optimization avoids local minima resulting from invalid pixel-wise posteriors and image noise.

Once the optimization is closer to the maximum, the global mean is not a good enough estimate, and more detailed refinement is required. In such cases, the algorithm switches to local optimization. We thereby use the probability values of the two discrete contour distances d_{si}^- and d_{si}^+ that are closest to the current estimate $d_{si}(\boldsymbol{\theta})$ and approximate the first-order derivatives using a weighting term $\alpha_s/\sigma_i{}^2$ and finite differences

$$\frac{\partial \ln\big(p(d_{si} \mid \omega_{si}, \boldsymbol{l}_{si})\big)}{\partial d_{si}} \approx \frac{\alpha_s}{\sigma_i{}^2} \ln\left(\frac{p(d_{si}^+ \mid \omega_{si}, \boldsymbol{l}_{si})}{p(d_{si}^- \mid \omega_{si}, \boldsymbol{l}_{si})}\right). \tag{3.43}$$

For second-order derivatives, again, the global approximation from Eq. (3.42) is used. Note that weighting the first-order derivative with the variance $\sigma_i{}^2$ improves robustness because correspondence lines with high uncertainty are considered less important. Simultaneously, the step size α_s helps to balance the weight and specifies how far the optimization proceeds, directly scaling the variation vector $\hat{\boldsymbol{\theta}}$. The same first- and second-order derivatives can also be derived using inverse-variance weighting and a constant curvature of $1/\alpha_\mathrm{s}$ for the second-order derivative. A detailed derivation of this interpretation is given in Appendix D.

Finally, apart from the choice of derivatives, the parameterization of smoothed step functions and the corresponding shape of posterior probability distributions significantly influences the optimization. To study this effect, we consider the first-order derivatives of the log-posteriors that are shown in Fig. 3.5. While for Gaussian distributions, linear first-order derivatives lead to the estimation of the weighted mean over all correspondence lines, for Laplace distributions, binary derivatives guide the optimization towards the weighted median. Note that this again corresponds well to the interpretation of local and global uncertainty modeled by the slope parameter s_h and the amplitude parameter α_h. If only local uncertainty exists, it is advantageous to consider the magnitude of errors in the contour distance and optimize for the mean. At the same time, in the case of global noise, it is reasonable to only consider the direction of errors and conduct the optimization with respect to the median.

3.4. Implementation

The following section provides details for the implementation of our approach in the *M3T* algorithm and specifies parameter values for the evaluation conducted in Section 3.5. First, we start with the generation of the sparse viewpoint model and the calculation of color histograms. This is followed by a description of the tracking process. Finally, we explain how occlusions can be considered. All mentioned parameter values are carefully chosen to maximize tracking quality while not requiring unreasonable amounts of computation.

3.4.1. Sparse Viewpoint Model

For the sparse viewpoint model, $n_\mathrm{v} = 2562$ different views are considered. They are generated by subdividing the triangles of an icosahedron 4 times, resulting in an angle of approximately 4° between neighboring views. Virtual cameras that are used for the rendering are placed at a distance of 0.8 m to the object center. For all views, the orientation vector ${}_\mathrm{M}\boldsymbol{v}$ and a constant number of $n = 200$ model points ${}_\mathrm{M}\boldsymbol{X}_i$ and normal vectors ${}_\mathrm{M}\boldsymbol{N}_i$ are computed. In addition, for each point and view, we also compute so-called continuous distances for the foreground and background. Continuous distances describe the distance from the 2D model point $\boldsymbol{x}_i$ along the line defined by the normal vector $\boldsymbol{n}_i$ for which the foreground and background are not interrupted by each other. After their computation in the rendered image, they are converted and stored in meters. The values are later used by the tracker to disable individual correspondence lines for which continuous distances are below a certain threshold, and the assumption that only a single transition between foreground and background is present in the correspondence line is not sufficiently fulfilled.

3.4.2. Color Histograms

For the estimation of the color probability distributions $p(\boldsymbol{y} \mid m_\text{f})$ and $p(\boldsymbol{y} \mid m_\text{b})$, color histograms are used. Each dimension of the RGB color space is discretized by 32 equidistant bins, leading to a total of $32^3 = 32768$ values. The computation of the color histograms is started from the current pose estimate or from an initial pose, provided, for example, by a global 6DoF pose estimation pipeline. Based on this pose, 3D model points and normal vectors are projected into the image using Eqs. (3.29) and (3.30). In both the positive and negative directions of the normal vector, the first 20 pixels are then considered. Pixel colors along this line are assigned to either the foreground or background histogram, depending on which side of the projected model point they are. Note that fewer than 20 pixels are considered if a transition between foreground and background occurs within a shorter distance. Also, in cases where the contour location is more uncertain, it might be reasonable to use an offset from the center.

Due to motion or dynamic illumination, color statistics of both the foreground and background are continuously changing during tracking. To take those changes into account while at the same time considering previous observations, we use online adaptation. Based on Bibby and Reid 2008, we update the histograms as follows

$$p_t(\boldsymbol{y} \mid m_i) = \alpha_i p(\boldsymbol{y} \mid m_i) + (1 - \alpha_i) p_{t-1}(\boldsymbol{y} \mid m_i), \quad i \in \{\text{f}, \text{b}\}, \tag{3.44}$$

where $\alpha_\text{f} = 0.2$ and $\alpha_\text{b} = 0.2$ are the learning rates for the foreground and background, respectively. Note that $p(\boldsymbol{y} \mid m_i)$ denotes the observed histogram, while $p_t(\boldsymbol{y} \mid m_i)$ and $p_{t-1}(\boldsymbol{y} \mid m_i)$ are the adapted histograms of the current and previous time steps, respectively. For initialization, we directly use the observed histograms instead of blending them with previous values.

3.4.3. Tracking Process

To start tracking, an initial pose is required, which is typically obtained either from global object detection and 6DoF pose estimation methods or from dataset annotations. Based on this prediction and the corresponding camera image, color histograms for the foreground and background are initialized. After initialization, a tracking step is executed for each new image that is streamed from the camera. An overview of computation that is performed in a typical tracking step is given in Algorithm 3.1.

Starting from a new image and the previous pose estimate ${}_\text{C}\boldsymbol{T}_\text{M}$, we first retrieve the closest view of the sparse viewpoint model. Model points ${}_\text{M}\boldsymbol{X}_i$ and normal vectors ${}_\text{M}\boldsymbol{N}_i$ are then projected into the image plane to define correspondence lines. After that, continuous distances from the sparse viewpoint model are used to reject correspondence lines with distances that are below 3 segments. For the remaining correspondence lines, the posterior probability distribution $p(d_{\text{s}i} \mid \omega_{\text{s}i}, \boldsymbol{l}_{\text{s}i})$ is evaluated at 12 discrete values $d_{\text{s}i} \in \{-5.5, -4.5, \dots, 5.5\}$. In the calculation, we use 8 precomputed values for the smoothed step functions h_f and h_b, corresponding to $x \in \{-3.5, -2.5, \dots, 3.5\}$. Also, a minimal offset Δr_i is chosen such that the line coordinates r_i point to pixel centers while

Algorithm 3.1 Tracking Step

```
1:  Update camera image
2:  for $i = 1, 2, \dots, 7$ do
3:      Optional: Render depth image for occlusion handling
4:      Find closest view of the sparse viewpoint model
5:      Define correspondence lines in the image
6:      Compute discrete distributions $p(d_{si} \mid \omega_{si}, \boldsymbol{l}_{si})$
7:      for $j = 1, 2$ do
8:          Calculate gradient $\boldsymbol{g}$ and Hessian $\boldsymbol{H}$
9:          Estimate variation $\hat{\boldsymbol{\theta}}$ and update pose ${}_{\mathrm{C}}\boldsymbol{T}_{\mathrm{M}}$
10:     end for
11: end for
12: Update color histograms $p(\boldsymbol{y} \mid m_{\mathrm{f}})$ and $p(\boldsymbol{y} \mid m_{\mathrm{b}})$
```

the scaled line coordinates r_{si} ensure matching values for $x = r_{si} - d_{si}$. In our case, this means that $r_{si} \in \mathbb{Z}$. Having computed the distributions, two iterations of the regularized Newton optimization are executed. For the first iteration, the global optimization is used to quickly converge towards a rough pose estimate. In the second iteration, the local optimization is employed to refine this pose, using a step size of $\alpha_{\mathrm{s}} = 1.3$. As regularization parameters, we define $\lambda_{\mathrm{r}} = 1000$ and $\lambda_{\mathrm{t}} = 30000$.

To find the final pose, the process is repeated 7 times. We thereby choose larger scales for the first iteration and decrease the scale with each consecutive iteration. This choice has the effect that a large area with low resolution is considered in the beginning, while short lines with high resolution are used in later iterations. An example of correspondence lines at different scales is shown in Fig. 3.1. Similarly, the standard deviation σ_{r} is also adjusted with each iteration. This allows us to define our confidence in the data and the previous estimate, which typically improves with each iteration. In addition to the scale s and the uncertainty σ_{r}, both the amplitude parameter α_{h} and the slope parameter s_{h} depend on the considered sequence and model quality. We, therefore, adjust those parameters for every dataset and provide them in the evaluation section. Finally, having estimated the pose for an image, the prediction is used to update the color histograms. After that, the tracker waits for a new image to arrive.

3.4.4. Occlusion Handling

While the presented algorithm itself is already quite robust to occlusions, tracking results can be further improved by explicitly considering them. For this, we use either images from depth cameras that are close to the color camera or depth renderings that contain all known objects in the environment. By comparing the expected positions of 3D model points to depth values, occlusions can be detected, and correspondence lines can be disabled. For renderings, we simply check if the depth of the model point minus a user-defined threshold of 30 mm is smaller than the depth value at the corresponding

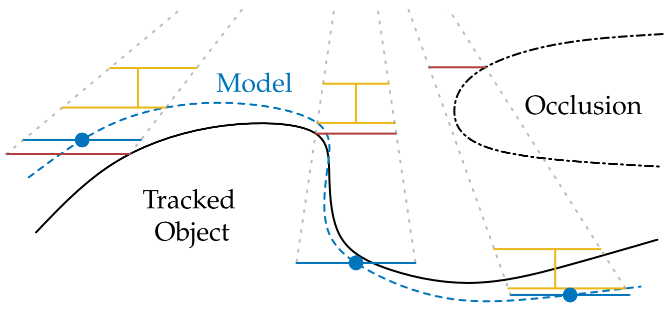


Figure 3.10.: Visualization of the occlusion handling strategy for depth camera images. For each blue model point, the considered region is defined by a blue line with a fixed length and a dotted gray cone. The lower yellow line in each cone visualizes the depth value that is calculated from the model offset, while the upper yellow line adds the user-defined threshold. Red lines indicate minimum depth measurements from the camera. For the right point, an occlusion is detected since the red depth measurement is smaller than the expected minimum allowed value in yellow.

image coordinate. For depth cameras, the process is more involved. The main reason is that depth cameras do not provide perfect images but often produce areas without measurements. To make the occlusion handling more robust, we have to consider a bigger region. We, therefore, first compute the minimum depth based on 25 image values within a quadratic region that corresponds to a patch of $20 \times 20\,\mathrm{mm}$. Similarly, during the generation of the sparse viewpoint model, an offset between the depth of the sampled model point and the minimum depth within a quadratic region of $20 \times 20\,\mathrm{mm}$ is calculated. Finally, the depth of the model point minus the precomputed offset and a threshold of $30\,\mathrm{mm}$ is compared to the minimum measurement from the depth camera. If it is smaller, the respective correspondence line is rejected. An illustration of this occlusion handling strategy is shown in Fig. 3.10.

3.5. Evaluation

In this section, we present an extensive evaluation of our approach, which was implemented in the *M3T* library. Both the *RBOT* (Tjaden et al. 2018) and *OPT* (Wu et al. 2017) datasets are used to compare our method to the state of the art in region-based tracking, as well as other techniques. We thereby evaluate the quality of the predicted poses and the speed of the algorithm. Also, a detailed analysis is conducted that assesses the importance of different parameters introduced in Sections 3.2 and 3.3. Finally, we discuss essential design considerations and remaining limitations. Videos that show the performance of a previous version of our approach are available online.[1,2]

[1] `https://www.youtube.com/watch?v=lwhxSRpwn3Y`
[2] `https://www.youtube.com/watch?v=TkS0Wkd_01A`

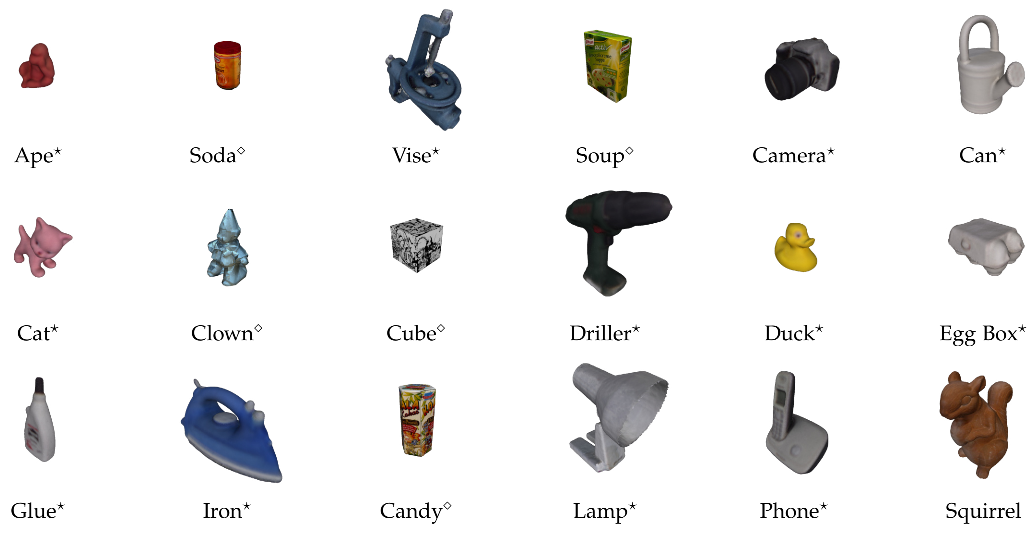


Figure 3.11.: Overview of all objects in the *RBOT* dataset (Tjaden et al. 2018). Objects from the *LINEMOD* dataset (Hinterstoisser et al. 2013) and *Rigid Pose* dataset (Pauwels et al. 2013) are marked with * and $^{\diamond}$, respectively.

3.5.1. RBOT Dataset

In the following, we first introduce the *RBOT* dataset, discuss the conducted experiments, and finally compare our results to the current state of the art. The *RBOT* dataset consists of 18 objects that are shown in Fig. 3.11. For each object, four sequences exist: a *regular* version, one with *dynamic light*, a sequence with both dynamic light and Gaussian *noise*, and one with dynamic light and an additional squirrel object that leads to *occlusion*. An example image for each sequence is shown in Fig. 3.12. Each sequence consists of 1001 semi-synthetic monocular images, where objects were rendered into real-world images that are recorded from a hand-held camera moving around a cluttered desk.

In the evaluation, experiments are performed as defined by Tjaden et al. (2018). The required translational and rotational errors are calculated as

$$e_{\mathrm{t}}(t_k) = \left\Vert {}_{\mathrm{C}}\boldsymbol{t}_{\mathrm{M}}(t_k) - {}_{\mathrm{C}}\boldsymbol{t}_{\mathrm{M_{GT}}}(t_k) \right\Vert_2, \tag{3.45}$$

$$e_{\mathrm{r}}(t_k) = \cos^{-1}\left(\frac{\operatorname{trace}\left({}_{\mathrm{C}}\boldsymbol{R}_{\mathrm{M}}(t_k)^\top {}_{\mathrm{C}}\boldsymbol{R}_{\mathrm{M_{GT}}}(t_k)\right) - 1}{2}\right), \tag{3.46}$$

where ${}_{\mathrm{C}}\boldsymbol{R}_{\mathrm{M_{GT}}}(t_k)$ and ${}_{\mathrm{C}}\boldsymbol{t}_{\mathrm{M_{GT}}}(t_k)$ are the ground-truth (GT) rotation matrix and translation vector for the frame $k \in \{0, \dots, 1000\}$. A pose is considered successful if both $e_{\mathrm{t}}(t_k) < 5\,\mathrm{cm}$ and $e_{\mathrm{r}}(t_k) < 5°$. After initializing the tracker with the ground-truth pose at t_0, it runs until either the recorded sequence ends or tracking is unsuccessful. In the case of unsuccessful tracking, the algorithm is re-initialized with the ground-truth pose at t_k. For the *Occlusion* sequence, the method is evaluated with and without occlusion modeling. In the case of occlusion modeling, where other objects are considered using a

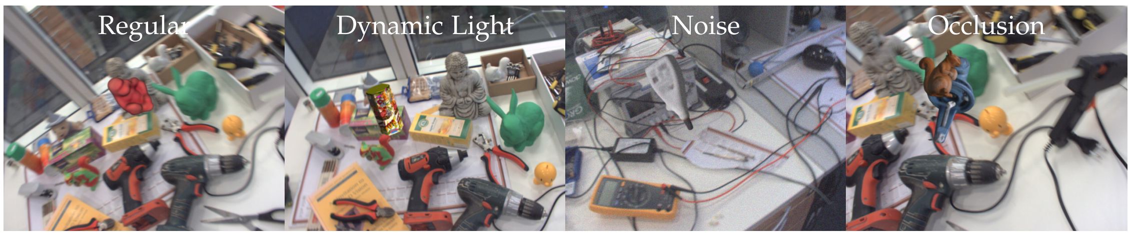


Figure 3.12.: Images from the *RBOT* dataset (Tjaden et al. 2018) with one example image for the *Regular*, *Dynamic Light*, *Noise*, and *Occlusion* sequence. Sequences show the *Ape*, *Candy*, *Glue*, and *Vise* objects, respectively. In addition, the *Occlusion* sequence features a squirrel object that occludes the *Vise*.

depth rendering, both objects are tracked simultaneously. Unsuccessful tracking of the occluding squirrel object is not considered in the reported tracking success. Finally, for our algorithm, we define the amplitude parameter $\alpha_\mathrm{h} = 0.36$ and the slope parameter $s_\mathrm{h} \rightarrow 0$ for the smoothed step functions. Also, we use the scale $s = \{5, 2, 2, 1\}$ and the uncertainty $\sigma_\mathrm{r} = \{20, 7, 3, 1.5\}$, where the standard deviation is defined in units of pixels. Note that the given sets define the value of each iteration, with the last value being used for all remaining iterations. A detailed analysis of the slope and amplitude parameters is provided in Section 3.5.3.

Results of the evaluation are shown in Table 3.1. Our approach is compared to the state of the art in region-based tracking. Also, we include algorithms of Huang et al. (2020), J.-C. Li et al. (2021), X. Sun et al. (2021), and Huang et al. (2022) that combine edge and region information. In addition, the method of F. Liu et al. (2021), which uses descriptor fields and region-based techniques, is also considered. The comparison shows that *M3T* performs significantly better than all previous methods, achieving superior results for most objects and performing best on average. This difference becomes even larger for purely region-based methods, with our algorithm performing best for almost all objects and sequences. Considering the average success rate, our approach performs more than six percentage points better than the combined method of X. Sun et al. (2021), about seven percentage points better than Huang et al. (2022), and more than 15 percentage points better than the next best, dense, region-based approach by Zhong and L. Zhang (2019). The superior tracking success compared to all other region-based methods is especially interesting since, in contrast to them, we do not use an advanced segmentation model. In theory, this should be a significant disadvantage.

In addition to tracking success, we also compare average runtimes. A summary for the different algorithms is given in Table 3.2. The evaluation of *M3T* was conducted on a computer with an *Intel Core i9-11900K* central processing unit (CPU) and a *NVIDIA RTX A5000* GPU. In the experiments, an average runtime of 0.9 ms for the case without occlusion modeling and an average execution time of 5.1 ms for the *Modeled Occlusion* scenario were obtained. Note that in the case of occlusion modeling, depth images have

Table 3.1.: Tracking success rates for state-of-the-art approaches on the *RBOT* dataset (Tjaden et al. 2018). Methods that are not purely region-based are indicated by a *. Results are from the corresponding publications. The best results are highlighted in bold, while the second-best values are underlined.

Approach	Ape	Soda	Vise	Soup	Camera	Can	Cat	Clown	Cube	Driller	Duck	Egg Box	Glue	Iron	Candy	Lamp	Phone	Squirrel	**Average**
Regular																			
Tjaden et al. (2018)	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	79.9
Zhong et al. (2020b)	88.8	41.3	94.0	85.9	86.9	89.0	98.5	93.7	83.1	87.3	86.2	78.5	58.6	86.3	57.9	91.7	85.0	96.2	82.7
Huang et al. (2020)*	91.9	44.8	**99.7**	89.1	89.3	90.6	97.4	95.9	83.9	97.6	91.8	84.4	59.0	92.5	74.3	97.4	86.4	99.7	86.9
F. Liu et al. (2021)*	93.7	39.3	98.4	91.6	84.6	89.2	97.9	95.9	86.3	95.1	93.4	77.7	61.5	87.8	65.0	95.2	85.7	**99.8**	85.5
J.-C. Li et al. (2021)*	92.8	42.6	96.8	87.5	90.7	86.2	99.0	96.9	86.8	94.6	90.4	87.0	57.6	88.7	59.9	96.5	90.6	99.5	85.8
X. Sun et al. (2021)*	93.0	55.2	99.3	85.4	96.1	93.9	98.0	95.6	79.5	**98.2**	89.7	89.1	66.5	91.3	60.6	98.6	95.6	99.6	88.1
Huang et al. (2022)*	94.6	49.4	99.5	91.0	93.7	96.0	97.8	96.6	90.2	**98.2**	93.4	90.3	64.4	94.0	79.0	**98.8**	92.9	**99.8**	89.9
M3T (Ours)	**99.0**	**68.1**	99.4	**96.7**	**96.8**	**96.9**	**99.9**	**98.7**	**95.6**	97.6	**98.1**	**95.8**	**81.8**	**95.2**	**92.6**	97.4	**96.2**	**99.8**	**94.8**
Dynamic Light																			
Tjaden et al. (2018)	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	81.2
Zhong et al. (2020b)	89.7	40.2	92.7	86.5	86.6	89.2	98.3	93.9	81.8	88.4	83.9	76.8	55.3	79.3	54.7	88.7	81.0	95.8	81.3
Huang et al. (2020)*	91.8	42.3	98.9	89.9	91.3	87.8	97.6	94.5	84.5	98.1	91.9	86.7	66.2	90.9	73.2	97.1	89.2	99.6	87.3
F. Liu et al. (2021)*	93.5	38.2	98.4	88.8	87.0	88.5	98.1	94.4	85.1	95.1	92.7	76.1	58.1	79.6	62.1	93.2	84.7	99.6	84.1
J.-C. Li et al. (2021)*	93.5	43.1	96.6	88.5	92.8	86.0	99.6	95.5	85.7	96.8	91.1	90.2	68.4	86.8	59.7	96.1	91.5	99.2	86.7
X. Sun et al. (2021)*	93.8	55.9	**99.6**	85.6	97.7	93.7	97.7	96.5	78.3	98.6	91.0	91.6	72.1	90.7	63.0	**98.9**	94.4	**100.0**	88.8
Huang et al. (2022)*	94.3	48.3	99.5	90.1	94.6	96.1	97.9	97.3	90.9	**99.1**	92.9	91.5	72.6	94.7	80.0	98.3	95.2	99.8	90.7
M3T (Ours)	**99.5**	**68.7**	99.4	**96.8**	**98.1**	**98.5**	**100.0**	**98.8**	**95.5**	97.9	**97.5**	**97.2**	**87.1**	**95.3**	**90.6**	97.9	**95.5**	99.8	**95.2**
Noise																			
Tjaden et al. (2018)	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	56.6
Zhong et al. (2020b)	79.3	35.2	82.6	86.2	65.1	56.9	96.9	67.0	37.5	75.2	85.4	35.2	18.9	63.7	35.4	64.6	66.3	93.2	63.6
Huang et al. (2020)*	89.0	45.0	89.5	90.2	68.9	38.3	95.9	72.8	20.1	85.5	92.2	26.8	15.8	66.2	52.2	58.3	65.1	98.4	65.0
F. Liu et al. (2021)*	84.7	33.0	88.8	89.5	56.4	50.1	94.1	66.5	32.3	79.6	94.2	29.6	19.9	63.4	40.3	61.6	62.4	96.9	63.5
J.-C. Li et al. (2021)*	89.1	44.0	91.6	89.4	75.2	62.3	98.6	77.3	41.2	81.5	91.6	54.5	31.8	65.0	46.0	78.5	69.6	97.6	71.4
X. Sun et al. (2021)*	92.5	56.2	**98.0**	85.1	**91.7**	**79.0**	97.7	86.2	40.1	**96.6**	90.8	**70.2**	**50.9**	84.3	49.9	**91.2**	**89.4**	**99.4**	80.5
Huang et al. (2022)*	91.0	49.1	95.6	91.0	76.3	54.1	97.1	73.7	27.3	92.8	95.3	30.2	7.8	73.9	56.8	71.4	70.8	98.7	69.6
M3T (Ours)	**97.5**	**65.7**	96.1	**96.7**	87.1	75.7	**99.9**	**90.0**	**67.2**	88.8	**97.9**	65.0	47.1	**84.6**	**80.5**	74.8	85.7	99.2	**83.3**
Unmodeled Occlusion																			
Tjaden et al. (2018)	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	73.3
Zhong et al. (2020b)	83.9	38.1	92.4	81.5	81.3	85.5	97.5	88.9	76.1	87.5	81.7	72.7	52.5	77.2	53.9	88.5	79.3	92.5	78.4
Huang et al. (2020)*	86.2	46.3	97.8	87.5	86.5	86.3	95.7	90.7	78.8	96.5	86.0	80.6	59.9	86.8	69.6	93.3	81.8	95.8	83.6
F. Liu et al. (2021)*	87.1	36.7	91.7	78.8	79.2	82.5	92.8	86.1	78.0	90.2	83.4	72.0	52.3	72.8	55.9	86.9	77.8	93.0	77.6
J.-C. Li et al. (2021)*	89.3	43.3	92.2	83.1	84.1	79.0	94.5	88.6	76.2	90.4	87.0	80.7	61.6	75.3	53.1	91.1	81.9	93.4	80.3
X. Sun et al. (2021)*	91.3	56.7	97.8	82.0	92.8	89.9	96.6	92.2	71.8	97.0	85.0	84.6	66.9	87.7	56.1	95.1	89.8	98.2	85.1
Huang et al. (2022)*	92.5	51.5	**99.2**	90.7	92.1	92.2	97.7	94.2	89.8	**98.4**	91.3	90.7	66.3	91.7	75.3	**95.9**	92.1	99.0	88.9
M3T (Ours)	**97.1**	**69.4**	**99.2**	**95.5**	**95.2**	**96.4**	**100.0**	**98.3**	**93.5**	96.8	**96.2**	**95.2**	**82.8**	**95.6**	**90.2**	95.8	**94.3**	**99.5**	**93.9**
Modeled Occlusion																			
Tjaden et al. (2018)	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	77.7
Huang et al. (2020)*	87.8	45.5	98.1	87.2	89.0	89.8	95.1	91.4	77.4	97.1	87.7	83.0	62.5	88.6	69.7	94.1	86.0	98.9	84.9
Huang et al. (2022)*	93.2	50.6	98.3	89.2	92.6	93.7	95.6	93.4	87.9	97.4	92.0	88.8	69.3	91.8	78.7	96.0	91.9	99.0	88.8
M3T (Ours)	**98.6**	**69.5**	**99.5**	**97.4**	**97.3**	**96.8**	**99.9**	**99.2**	**94.3**	**97.5**	**98.3**	**96.6**	**86.0**	**96.7**	**91.4**	**97.1**	**95.3**	**99.7**	**95.1**

Table 3.2.: Average runtimes per frame and usage of a GPU for state-of-the-art approaches. Methods that are not purely region-based are indicated by a $\star$. For the occlusion modeling scenario, which considers the tracking of two objects and the rendering of depth images, values are shown in parenthesis.

Approach	No GPU	Runtime
Tjaden et al. (2018)	✗	$15.5 \sim 21.8$ ms
Zhong et al. (2020b)	✗	41.2 ms
Huang et al. (2020)*	✗	33.1 ms
F. Liu et al. (2021)*	✗	6.9 ms
J.-C. Li et al. (2021)*	✗	32.1 ms
X. Sun et al. (2021)*	✗	$40.0 \sim 50.0$ ms
Huang et al. (2022)*	✗	$30.4 \sim 36.4$ ms
M3T (Ours)	✓(✗)	0.9 ms (5.1 ms)

to be rendered, and the reported time is for the simultaneous tracking of two objects. In comparison, except for the algorithm of F. Liu et al. (2021), for which the execution time is seven times higher, all other methods report average runtimes that are more than one order of magnitude larger. The difference is even more impressive since *M3T* only utilizes a single CPU core and does not require a GPU. In contrast, most competing methods typically use multithreading and heavily depend on a GPU. In conclusion, while different resources and computers were used, the obtained results highlight the superior efficiency of our sparse region-based method.

3.5.2. OPT Dataset

While the semi-synthetic *RBOT* dataset features a large number of objects, a difficult, highly-cluttered background, and perfect ground truth, objects are simulated with limited realism, and only very little motion blur is applied. Those shortcomings are complemented by the *OPT* dataset (Wu et al. 2017), which contains real-world recordings of 3D-printed objects on a white background with different speeds and levels of motion blur. In total, the dataset includes six objects and consists of 552 real-world sequences with various lighting conditions and defined trajectories recorded by a robot arm. An example image for each object is shown in Fig. 3.13. The sequences are classified into the following categories: translation, forward and backward, in-plane rotation, out-of-plane rotation, flashing light, moving light, and free motion.

In the experiments, the metric of Wu et al. (2017) is used. For this, we compute the average vertex error

$$e_{\mathrm{v}}(t_k) = \frac{1}{n} \sum_{i=1}^{n} \left\| \left({}_{\mathrm{M}}\widetilde{\boldsymbol{X}}_i - {}_{\mathrm{M}}\boldsymbol{T}_{\mathrm{M_{GT}}}(t_k) {}_{\mathrm{M}}\widetilde{\boldsymbol{X}}_i \right)_{3\times1} \right\|_2, \tag{3.47}$$

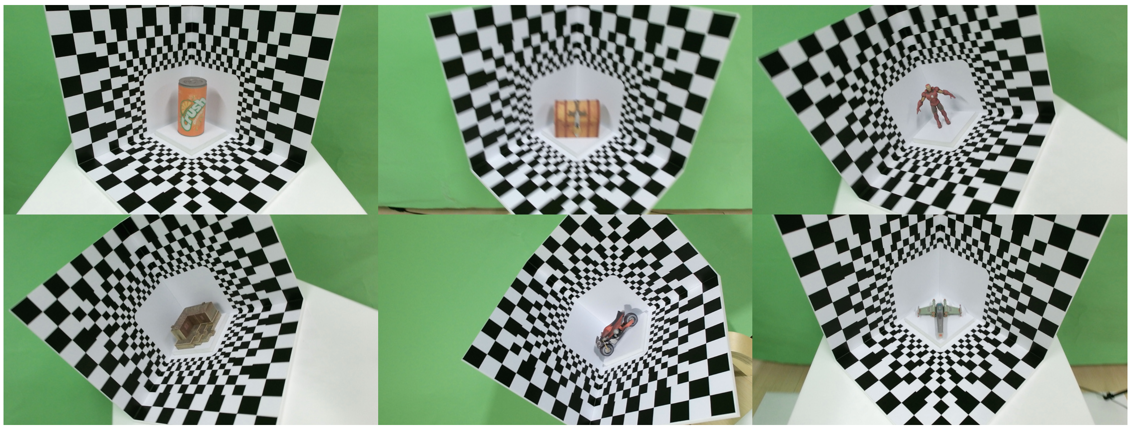


Figure 3.13.: Images from the *OPT* dataset (Wu et al. 2017), featuring the *Soda*, *Chest*, *Ironman*, *House*, *Bike*, and *Jet* objects.

with $\widetilde{\boldsymbol{X}}_i$ a vertex in the 3D mesh geometry of the object, and n the number of vertices. Tracking is considered successful if $e_\mathrm{v}(t_k) < k_\mathrm{e}d$, where d is the maximum vertex distance, and k_e is an error threshold. The tracking quality for all frames is then measured using an area-under-curve (AUC) score that integrates the percentage value of successfully tracked poses over the interval $k_\mathrm{e} \in [0, 0.2]$. This results in AUC scores between zero and twenty. For the tracker, the parameters $s = \{6, 4, 1\}$, $\sigma_\mathrm{r} = \{25, 10, 2.5\}$, $\alpha_\mathrm{h} = 0.43$, and $s_\mathrm{h} = 0.5$ are used. Also, a larger rotational regularization parameter of $\lambda_\mathrm{r} = 30000$ is adopted for the rotationally symmetric *Soda* object. The main reason is that the object geometry of the *Soda* object does not constrain the rotation around the vertical axis. In such cases, fluctuations in the gradient and Hessian can lead to drift in the object's orientation. Using more regularization allows us to mitigate this problem.

Results for our experiments on the *OPT* dataset are shown in Table 3.3. We thereby compare *M3T* to state-of-the-art region-based tracking approaches, as well as edge-based approaches from Bugaev et al. (2018), J.-C. Li et al. (2021), and Huang et al. (2022). Also, prominent methods such as *PWP3D* (Prisacariu and Reid 2012), *ElasticFusion* (Whelan et al. 2015), *UDP* (Brachmann et al. 2016), and *ORB-SLAM2* (Mur-Artal and Tardós 2017) are included. Note that not all algorithms are dedicated 3D tracking solutions. *UDP* is a global 6DoF pose estimation method, while *ElasticFusion* and *ORB-SLAM2* are visual SLAM approaches for camera pose localization that are applied to the silhouette of the object. For more information on the evaluation of those algorithms, please refer to Wu et al. (2017).

The comparison shows that our approach performs significantly better than the state of the art in region-based tracking represented by J.-C. Li et al. (2021), Zhong et al. (2020b), and Tjaden et al. (2018), with *M3T* achieving higher AUC scores for each of the six objects. Also, compared to none region-based approaches, we are able to report the highest score for four out of six objects and perform best on average. This is even

Table 3.3.: AUC scores between zero and twenty for the evaluation on the *OPT* dataset (Wu et al. 2017), comparing our approach to multiple other algorithms. Values are taken from Wu et al. (2017) and the respective publications. The best results are highlighted in bold, while the second-best values are underlined.

Approach	Soda	Chest	Ironman	House	Bike	Jet	**Avg.**
PWP3D	5.87	5.55	3.92	3.58	5.36	5.81	5.01
ElasticFusion	1.90	1.53	1.69	2.70	1.57	1.86	1.87
UDP	8.49	6.79	5.25	5.97	6.10	2.34	5.82
ORB-SLAM2	13.44	<u>15.53</u>	11.20	**17.28**	10.41	9.93	12.97
Bugaev et al. 2018	<u>14.85</u>	14.97	<u>14.71</u>	14.48	12.55	**17.17**	<u>14.79</u>
Tjaden et al. 2018	8.86	11.76	11.99	10.15	11.90	13.22	11.31
Zhong et al. 2020b	9.01	12.24	11.21	13.61	12.83	15.44	12.39
J.-C. Li et al. 2021	9.00	14.92	13.44	13.60	<u>12.85</u>	10.64	12.41
Huang et al. 2022	9.07	12.93	8.80	11.15	7.96	11.09	10.17
M3T (Ours)	**15.55**	**17.29**	**17.50**	<u>16.57</u>	**13.06**	<u>16.14</u>	**16.02**

more remarkable since *ORB-SLAM2*, which reports better results for the *House* object, uses gradient-based corner features. In contrast to *M3T*, the algorithm is thus not constrained to the contour but considers information over the entire silhouette. Also, the edge-based algorithm of Bugaev et al. (2018), which performs best for the *Jet* object, uses basin-hopping for global optimization, and, with an average reported runtime of 683 ms, is not real-time capable. In conclusion, the obtained results demonstrate that *M3T* not only performs well on simulated data but also for real-world sequences.

3.5.3. Parameter Analysis

Having evaluated the performance of our approach, we want to foster our understanding of parameter values that were introduced in Sections 3.2 and 3.3. For this, the average success rate for the *RBOT* dataset and the average AUC score for the *OPT* dataset are plotted over different parameter values. The plots are shown in Fig. 3.14. Note that the success rate and AUC score are computed over all objects and sequences. Except for the analyzed parameters, the same settings as in Sections 3.5.1 and 3.5.2 are used.

For the amplitude parameter α_h, we observe that while the parameter significantly influences the tracking success, the effect on the AUC score is much smaller. Knowing that the amplitude parameter models global uncertainty, this corresponds well with the fact that the *RBOT* dataset features highly-cluttered images while the *OPT* dataset contains a constant white background. For the slope parameter s_h, the highest tracking success is observed for $s_\text{h} \rightarrow 0$, and the best AUC score is obtained at $s_\text{h} = 0.5$. Again, this is well explained by the theoretical interpretation according to which the slope parameter models local uncertainty. Given perfect information about the object geometry,

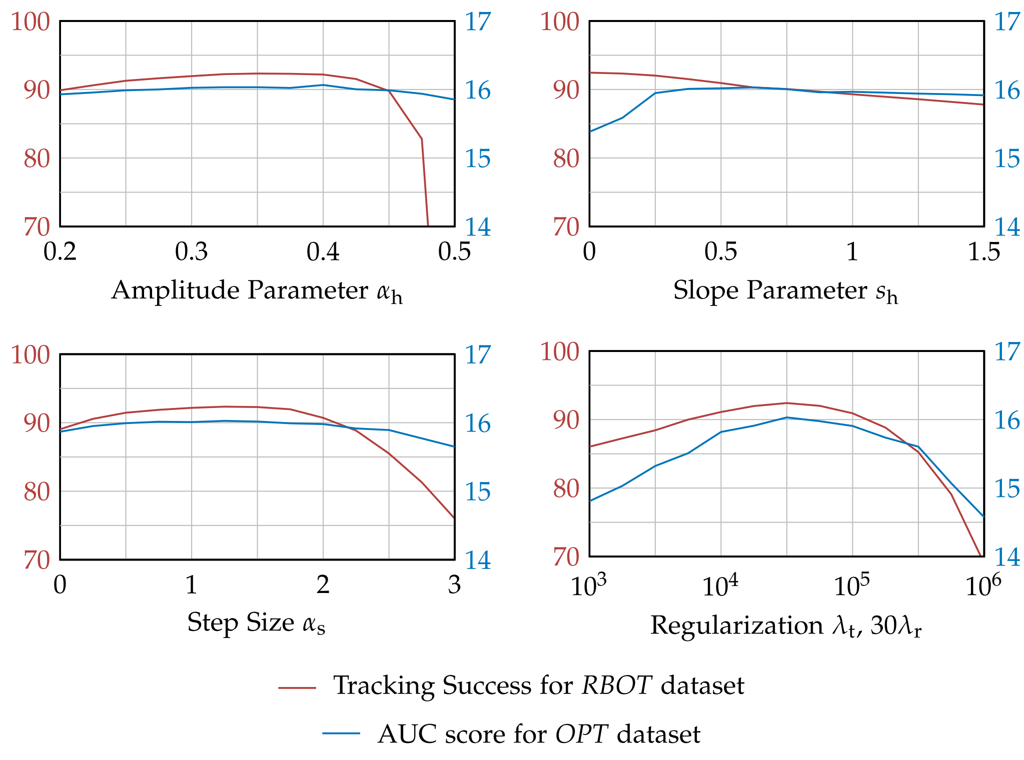


Figure 3.14.: Average tracking success for the *RBOT* dataset and average AUC score for the *OPT* dataset over different values of the amplitude parameter α_h, slope parameter s_h, step size α_s, and the rotational and translational regularization parameters λ_r and λ_t. For the evaluation of the regularization parameters, we use the constant relation $\lambda_t = 30\lambda_r$.

we do not expect any inconsistency for the semi-synthetic *RBOT* dataset. At the same time, for the *OPT* dataset, with imperfectly 3D printed objects and recorded real-world images, it is important to explicitly model local uncertainty.

Studying the plot of the step size α_s, we observe a relatively large plateau around one, with maximum values at $\alpha_s = 1.3$, both for the tracking success and the AUC score. This suggests a low dependency between the parameter and different image data. Particularly interesting are also the results for $\alpha_s = 0$. For this setting, no local optimization is conducted, showing the capability of global optimization alone. While values are slightly smaller, the good results highlight the excellent performance of the adopted global approximation.

Finally, for the evaluation of regularization, the rotational and translational parameters are modified simultaneously. To consider the different units of radians and meters, we define $\lambda_t = 30\lambda_r$. Like in previous evaluations of the *Soda* object, we increase the rotational parameter and use $\lambda_r = \lambda_t$. The resulting plots for the tracking success and the AUC score demonstrate the high importance of regularization. If values are chosen too small, the optimization is unstable for directions in which no or very little information is available. At the same time, if parameters are too large, the optimization is slowed down, and the final pose might not be reached. It is thus important to find values that lie in between. In our experience, a good approximation is to use regularization parameters that are in the same order of magnitude as the maximum rotational and translational diagonal elements of the Hessian matrix.

In conclusion, the conducted parameter analysis demonstrates that experimental results correspond well to theoretical interpretations from Sections 3.2 and 3.3. In addition to fostering our understanding, this explainability helps to guide the parameter search for new applications. Moreover, the results in Fig. 3.14 demonstrate that all parameters are well-behaved, with large plateaus around the maximum and no sudden jumps. This has the advantage that parameters are easy to tune, with a broad range of values achieving satisfying results.

3.5.4. Discussion

The conducted experiments demonstrate the excellent performance of our approach. In the following, we want to discuss design considerations that are essential in achieving those results and shed some light on remaining limitations. With respect to computational efficiency, the biggest performance gain is attributed to our novel correspondence line model and the sparse nature of the method. In addition, the sparse viewpoint model provides a highly-efficient representation, which requires only a simple search to obtain the object contour for the current pose. Also, in contrast to dense methods, for our approach, it is not necessary to compute a 2D signed distance function, but one can simply use the projected contour distance. Finally, the discrete scale-space formulation reduces the amount of computation further by combining multiple pixels into segments and supporting the use of precomputed smoothed step functions.

For the quality of the pose estimate, multiple aspects have to be considered. The first important factor is the use of smoothed step functions that provide realistic modeling of local and global uncertainty. Consequently, this leads to reliable posterior probability distributions for the contour location. Also, due to the one-dimensionality of correspondence lines and the discrete scale-space implementation, we are able to sample values over posterior probability distributions in reasonable time. This allows us to calculate the mean and the variance. Both estimates constitute the basis for fast-converging global optimization that is independent of local minima. In addition, knowledge about the uncertainty of individual correspondence lines is also used for local optimization, where numerical first-order derivatives are weighted according to the inverse variance. Finally, Tikhonov regularization is another important factor. It helps to constrain the estimate with respect to the previous pose, stabilizing the optimization for directions in which no or very little information is available.

While the described algorithm achieves remarkable results and works very well in a wide variety of applications, some challenges remain. The main limitations are thereby similar to other region-based methods. First, objects have to be rigid, and an accurate

3D model has to be known. Also, the background has to be distinguishable from the object. If large areas in the background contain colors that are present in the object, the final result might be perturbed. Another challenge comes from ambiguities where the object silhouette is very similar in the vicinity of a particular pose. Naturally, in such cases, there is not enough information, and it is impossible for the algorithm to predict the correct pose. Similarly, if large parts of the object are occluded, the visible part of the contour might not fully constrain the pose of the object, leading to erroneous estimates. Finally, like most tracking approaches, the algorithm can only be used for local optimization with a limit to the maximum pose difference from one frame to the next. Most of those limitations will be addressed in the next chapter, where we combine information from multiple modalities to further improve the robustness, accuracy, and convergence of our algorithm.

3.6. Conclusion

In this chapter, we proposed a highly-efficient sparse approach to region-based 3D object tracking that uses correspondence lines to find the pose that best explains the segmentation of the image. In addition to a thorough mathematical derivation of correspondence lines, we provided a highly-efficient scale-space implementation. Another important contribution is the introduction of smoothed step functions that allow the modeling of both local and global uncertainty. The effect of this modeling was analyzed in detail with respect to both the theoretical posterior probability and the quality of the final tracking result. For the maximization of the pose-dependent joint posterior probability, we proposed the use of an initial global optimization towards the mean of each distribution and a consecutive local optimization that considers discrete probability values. In addition, for the local first-order derivative, a novel approximation that uses inverse variance weighting on the finite difference value was developed. Furthermore, to improve results for real-world applications such as robotic manipulation, a robust occlusion handling strategy was implemented. Finally, in multiple experiments on the *RBOT* and *OPT* datasets, we demonstrated that our implementation in the *M3T* library outperforms the current state of the art in region-based tracking by a considerable margin both in terms of quality and efficiency.

4 Multi-modality Tracking

4.1. Introduction

Combining different techniques and using multiple sources of information greatly helps to improve tracking quality. In addition to region, some of the most prominent types of information are depth and texture. Depth-based methods minimize the distance between a geometric object model and measurements from a depth camera. For the consideration of texture, both keypoint features and direct optimization are used. While keypoint-based methods reduce the distance between matching characteristic points, direct optimization minimizes a pixel-wise error over the object silhouette. Region-based methods, on the other hand, differentiate between the object silhouette and the background. In addition, instead of considering the entire object as one single region, it is possible to distinguish between multiple areas on the object surface. In general, region, depth, and texture are highly complementary. For example, while region information works well to constrain in-plane rotation and translation, depth measurements are highly valuable for the prediction of the out-of-plane rotation and the camera distance. In addition, texture is able to provide information in cases of inconclusive object geometry, where techniques based on single-region and depth would fail. Also, while texture considers local object characteristics, such as text or graphics, multi-region takes into account global differences from distinct materials and colors.

To utilize this complementary information, the following chapter presents an approach that implements region, depth, and texture modalities, and allows their flexible combination. Also, it enables the incorporation of data from multiple cameras for which the relative pose is known. Because of its modularity, the method can be adapted to various object characteristics and available sensory information. In the following, we start with an introduction to the probabilistic framework that is used to combine information from different modalities and cameras. Subsequently, based on the *ICP* algorithm, a depth modality is developed. This is followed by the introduction of a keypoint-based texture modality that utilizes data from local object appearance. Finally, using the proposed multi-modality framework, we extend the region approach from Chapter 3 to multi-region tracking. This allows our method to distinguish between different regions on the object surface. An illustration of the resulting approach with all developed modalities is shown in Fig. 4.1.

For our implementation in the *M3T* library, an extensive evaluation on the *YCB-Video* (Xiang et al. 2018), *OPT* (Wu et al. 2017), and *Choi* (Choi and Christensen 2013)

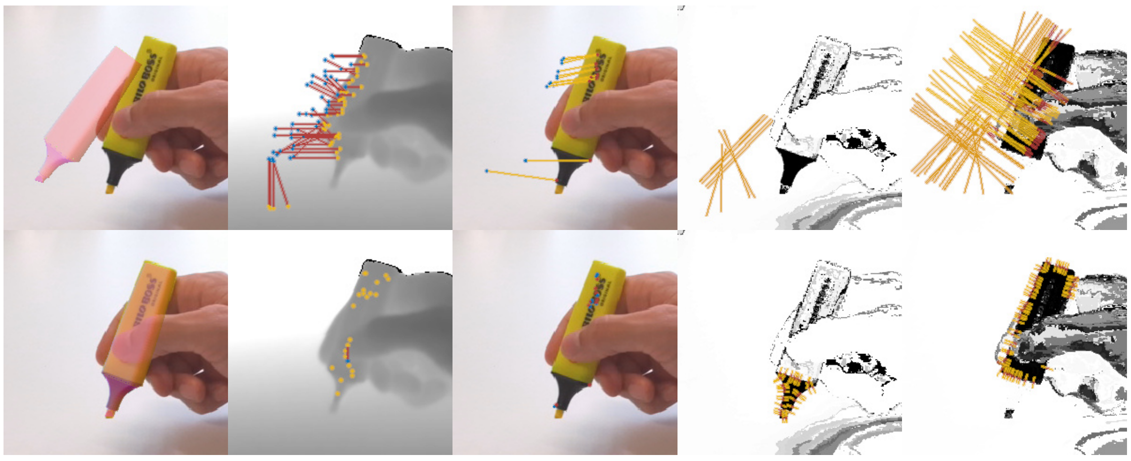


Figure 4.1.: Tracking of a marker pen using multiple modalities. In the first row, the initial state of the optimization is shown, while the second row visualizes the final state. For the left column, a rendered overlay that illustrates the object in the estimated pose is given. In the visualization of the depth modality, the correspondence between blue model points and yellow depth image points is illustrated by red lines. For the texture modality, keyframe features are given in blue, while detected points from the current frame are shown in red. Matched features are connected by yellow lines. For the two region modalities, which consider the pen's yellow body and black tip, probabilities that pixels belong to the respective object region are encoded in grayscale images. Pixels along correspondence lines are thereby illustrated in yellow with high probabilities for the location of the contour indicated in red. As before, with a decreasing scale parameter s for the optimization, correspondence lines become shorter while the resolution is increased.

datasets is conducted. We thereby differentiate between geometry-based tracking and the combination of all developed modalities. While the first uses single-region and depth, which are the only sources of information available for textureless objects, the second also considers visual appearance, combining depth, texture, and multi-region. Our experiments show that in both cases, *M3T* outperforms the state of the art with respect to tracking quality while remaining extremely efficient. Because the evaluation includes both a wide variety of conventional methods and state-of-the-art deep learning-based approaches, we also gain new insights into the current state of deep learning-based tracking. In addition, we compare our method to global 6DoF pose estimation methods and assess our approach's potential for highly-efficient pose refinement. Note that the chapter is based on *ICG* and *ICG+* and the corresponding publications (Stoiber et al. 2022b; Stoiber et al. 2023a). In addition, the texture modality draws from the master's thesis of Elsayed (2021). Also, general concepts of the multi-region approach were first presented in the bachelor's thesis of Reichert (2021).

4.2. Probabilistic Model

The following section introduces the probabilistic model for our multi-modality tracking approach. First, we start by developing a general framework that is able to combine information from multiple modalities and cameras. This is followed by the derivation of an *ICP*-based depth modality. To include texture information, a texture modality that uses keypoint features is then developed. Finally, the sparse region approach from Chapter 3 is extended to consider multiple regions on the object surface.

4.2.1. Framework

Similar to Chapter 3, we use a probabilistic formulation that expresses the probability of a particular pose given data considered by the tracking approach. The pose is then estimated by maximizing this probability density function (PDF). For our multi-modality tracker, the joint probability over all considered modalities is defined as

$$p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{D}}) \propto \prod_{i=1}^{n_\mathrm{d}} p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{D}}_{\mathrm{d}i}) \prod_{i=1}^{n_\mathrm{t}} p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{D}}_{\mathrm{t}i}) \prod_{i=1}^{n_\mathrm{r}} p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{D}}_{\mathrm{r}i}), \tag{4.1}$$

where $\boldsymbol{\mathcal{D}}_{\mathrm{d}i}$ is data from a depth modality, $\boldsymbol{\mathcal{D}}_{\mathrm{t}i}$ considers data from a texture modality, $\boldsymbol{\mathcal{D}}_{\mathrm{r}i}$ denotes data from a region modality, and $\boldsymbol{\theta}^\top = \begin{bmatrix}\boldsymbol{\theta}_\mathrm{r}^\top & \boldsymbol{\theta}_\mathrm{t}^\top\end{bmatrix}$ is the pose variation. Note that while, for texture and depth, only one modality per camera is used, for multi-region, multiple modalities can be configured per camera. The parameter n_r thereby describes the number of region modalities summed over all cameras.

Like in Section 3.3.4, Newton optimization with Tikhonov regularization is used to estimate the pose variation

$$\hat{\boldsymbol{\theta}} = \left(-\boldsymbol{H} + \begin{bmatrix} \lambda_\mathrm{r}\boldsymbol{I}_3 & \boldsymbol{0} \\ \boldsymbol{0} & \lambda_\mathrm{t}\boldsymbol{I}_3 \end{bmatrix} \right)^{-1} \boldsymbol{g}, \tag{4.2}$$

where λ_r and λ_t are rotational and translational regularization parameters. The gradient vector $\boldsymbol{g}$ and the Hessian matrix $\boldsymbol{H}$ are the first- and second-order derivatives of the logarithm of the joint PDF in Eq. (4.1). They are defined as

$$\boldsymbol{g}^\top = \frac{\partial}{\partial \boldsymbol{\theta}} \ln\big(p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{D}})\big)\Big|_{\boldsymbol{\theta}=\boldsymbol{0}}, \tag{4.3}$$

$$\boldsymbol{H} = \frac{\partial^2}{\partial \boldsymbol{\theta}^2} \ln\big(p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{D}})\big)\Big|_{\boldsymbol{\theta}=\boldsymbol{0}}. \tag{4.4}$$

Based on the gradient vectors and Hessian matrices of individual modalities, the full gradient vector and Hessian matrix can be assembled as follows

$$\boldsymbol{g} = \sum_{i=1}^{n_\mathrm{d}} \boldsymbol{g}_{\mathrm{d}i} + \sum_{i=1}^{n_\mathrm{t}} \boldsymbol{g}_{\mathrm{t}i} + \sum_{i=1}^{n_\mathrm{r}} \boldsymbol{g}_{\mathrm{r}i}, \tag{4.5}$$

$$\boldsymbol{H} = \sum_{i=1}^{n_\mathrm{d}} \boldsymbol{H}_{\mathrm{d}i} + \sum_{i=1}^{n_\mathrm{t}} \boldsymbol{H}_{\mathrm{t}i} + \sum_{i=1}^{n_\mathrm{r}} \boldsymbol{H}_{\mathrm{r}i}. \tag{4.6}$$

For the depth and texture modality, the gradient vectors $\boldsymbol{g}_{\mathrm{d}i}$ and $\boldsymbol{g}_{\mathrm{t}i}$ and the Hessian matrices $\boldsymbol{H}_{\mathrm{d}i}$ and $\boldsymbol{H}_{\mathrm{t}i}$ will be defined in Sections 4.2.2 and 4.2.3. For multi-region tracking, only the sparse viewpoint model that considers the object's geometry has to be adapted. The gradient vector $\boldsymbol{g}_{\mathrm{r}i}$ and the Hessian matrix $\boldsymbol{H}_{\mathrm{r}i}$ remain the same as in the previous chapter. They are defined in Section 3.3.5.

Finally, given the calculated variation vector $\hat{\boldsymbol{\theta}}$, pose estimates are updated as follows

$$_\mathrm{C}\boldsymbol{T}_\mathrm{M}^{+} = {}_\mathrm{C}\boldsymbol{T}_\mathrm{M} \begin{bmatrix} \exp([\hat{\boldsymbol{\theta}}_\mathrm{r}]_\times) & \hat{\boldsymbol{\theta}}_\mathrm{t} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{4.7}$$

where C denotes the coordinate frame of an arbitrary depth or color camera. Typically, only the pose estimate with respect to one camera is calculated using Eq. (4.7). For all other cameras, the fixed relative transformation is used. By iteratively repeating the process, one is able to find the object pose that maximizes the joint PDF, considering depth, texture, and multi-region information from multiple cameras.

4.2.2. Depth Modality

In the following section, we develop a depth modality based on the *ICP* approach of Besl and McKay (1992). While signed distance functions (Fitzgibbon 2003) could also be used, the sparse nature of *ICP* fits better to the sparse region approach developed in Chapter 3. As defined by the *ICP* process, we start with a search of depth image points that correspond to model points. For this, similar to projective data association (Blais and Levine 1995; Dorai et al. 1998), a 3D surface point $\boldsymbol{X}_i$ from the sparse viewpoint model is first projected into the depth image. Given a user-defined radius and stride, multiple 3D points within a quadratic area are then reconstructed. The final correspondence point $\boldsymbol{P}_i \in \mathbb{R}^3$ is then selected as the point closest to the point $\boldsymbol{X}_i$ in euclidean space. Correspondences with a distance bigger than a threshold are rejected. Note that techniques such as normal shooting (Chen and Medioni 1992), as well as rejection strategies based on the median distance (Diebel et al. 2004), the best percentage (Pulli 1999), and the compatibility of normal vectors (Pulli 1999) were also tested. However, in the end, this relatively simple procedure worked best.

For the probabilistic model, we formulate a normal distribution that adopts the point-to-plane error metric of Chen and Medioni (1992). The distance between the 3D surface point $\boldsymbol{X}_i$ and the correspondence point $\boldsymbol{P}_i$ is thereby calculated along the associated normal vector $\boldsymbol{N}_i$. Given data $\mathcal{D}_\mathrm{d}$ from a single depth modality, the probability of the pose variation vector $\boldsymbol{\theta}$ is expressed by the following normal distribution

$$p(\boldsymbol{\theta} \mid \mathcal{D}_\mathrm{d}) \propto \prod_{i=1}^{n} \exp\left(-\frac{1}{2{d_{\mathrm{Z}i}}^2{\sigma_\mathrm{d}}^2}\left({}_\mathrm{M}\boldsymbol{N}_i^\top\left({}_\mathrm{M}\boldsymbol{X}_i - {}_\mathrm{M}\boldsymbol{P}_i(\boldsymbol{\theta})\right)\right)^2\right), \tag{4.8}$$

with

$${}_\mathrm{M}\widetilde{\boldsymbol{P}}_i(\boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{I} - [\boldsymbol{\theta}_\mathrm{r}]_\times & -\boldsymbol{\theta}_\mathrm{t} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{bmatrix} {}_\mathrm{M}\boldsymbol{R}_\mathrm{C} & {}_\mathrm{M}\boldsymbol{t}_\mathrm{C} \\ \boldsymbol{0} & 1 \end{bmatrix} {}_\mathrm{C}\widetilde{\boldsymbol{P}}_i, \tag{4.9}$$

where the correspondence point $\boldsymbol{P}_i$ is variated instead of the model point $\boldsymbol{X}_i$. This has the advantage that the normal vector remains fixed, and only one vector has to be variated. For the linearized inverse variation, we simply use the negative variation vector $-\boldsymbol{\theta}$. Also, note that the user-defined standard deviation σ_d is scaled by the correspondence point's depth value $d_{\mathrm{Z}i}$. The scaling takes into account that the number and quality of depth measurements decrease with the distance to the camera. In addition, it also ensures compatibility with the region modality's uncertainty, which is defined in units of pixels and therefore increases with the camera distance.

Finally, based on the designed PDF, the following gradient vector and Hessian matrix for a single depth modality can be derived

$$\boldsymbol{g}_\mathrm{d} = -\sum_{i=1}^{n} \frac{1}{{d_{\mathrm{Z}i}}^2{\sigma_\mathrm{d}}^2} {}_\mathrm{M}\boldsymbol{N}_i^\top\left({}_\mathrm{M}\boldsymbol{X}_i - {}_\mathrm{M}\boldsymbol{P}_i\right) \begin{bmatrix} {}_\mathrm{M}\boldsymbol{P}_i \times {}_\mathrm{M}\boldsymbol{N}_i \\ {}_\mathrm{M}\boldsymbol{N}_i \end{bmatrix}, \tag{4.10}$$

$$\boldsymbol{H}_\mathrm{d} = -\sum_{i=1}^{n} \frac{1}{{d_{\mathrm{Z}i}}^2{\sigma_\mathrm{d}}^2} \begin{bmatrix} {}_\mathrm{M}\boldsymbol{P}_i \times {}_\mathrm{M}\boldsymbol{N}_i \\ {}_\mathrm{M}\boldsymbol{N}_i \end{bmatrix} \begin{bmatrix} {}_\mathrm{M}\boldsymbol{P}_i \times {}_\mathrm{M}\boldsymbol{N}_i \\ {}_\mathrm{M}\boldsymbol{N}_i \end{bmatrix}^\top. \tag{4.11}$$

4.2.3. Texture Modality

To consider local object appearance from text, graphics, or patterns, we design a PDF that uses keypoint features. Note that while direct optimization could also be used, we believe that the large basin of convergence and high robustness to illumination changes of keypoint features outweigh the requirement for strong texture. For our method, keypoints are detected on each new frame within a rectangular region close to the previous pose estimate. Descriptors from those points are then matched to features from keyframes. In general, a frame is considered a keyframe if the orientational difference to existing keyframes exceeds a certain threshold. Note that the use of keyframes helps to reduce drift and makes tracking more robust. Finally, if a frame is considered a keyframe, a depth rendering is generated, and keypoints that fall on the object silhouette are reconstructed using Eq. (3.22). Together with their respective descriptors, unoccluded 3D points are then stored for the keyframe.

Given detected 2D points $\boldsymbol{x}_i'$ from the current frame and matching 3D model points ${}_\mathrm{M}\boldsymbol{X}_i$ from keyframes, a PDF can be formulated. Similar to the method of Vacchetti et al. (2004), we assume a normal distribution for the reprojection error and write the following function for independent point pairs

$$p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{D}}_\mathrm{t}) \propto \prod_{i=1}^{n} \exp\bigg(-\frac{1}{2{\sigma_\mathrm{t}}^2} \rho_\mathrm{tuk}\Big(\big\lVert \boldsymbol{x}_i' - \boldsymbol{x}_i(\boldsymbol{\theta}) \big\rVert_2 \Big) \bigg), \tag{4.12}$$

with variated model points that are projected into the image space

$$\boldsymbol{x}_i(\boldsymbol{\theta}) = \boldsymbol{\pi}\Bigg(\begin{bmatrix} {}_\mathrm{C}\boldsymbol{R}_\mathrm{M} & {}_\mathrm{C}\boldsymbol{t}_\mathrm{M} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{I} + [\boldsymbol{\theta}_\mathrm{r}]_\times & \boldsymbol{\theta}_\mathrm{t} \\ \boldsymbol{0} & 1 \end{bmatrix} {}_\mathrm{M}\widetilde{\boldsymbol{X}}_i \Bigg). \tag{4.13}$$

The user-defined standard deviation σ_t takes into account the uncertainty of the texture modality compared to other modalities. Like the region modality's standard deviation σ_r, it is defined in units of pixels. The term ρ_tuk denotes the Tukey norm. It minimizes the effect of outliers from wrong matches and is defined as

$$\rho_\mathrm{tuk}(x) = \begin{cases} \frac{c^2}{6}\left(1 - \left(1 - \left(\frac{x}{c}\right)^2\right)^3\right) & \text{if } |x| \leq c \\ \frac{c^2}{6} & \text{otherwise} \end{cases}, \tag{4.14}$$

where c is a constant value that is provided by the user. It is typically set to the maximum residual error expected for inliers.

Based on the PDF in Eq. (4.12), the gradient vector and the Hessian matrix required for Newton optimization can be derived. Like in other approaches (Lepetit and Fua 2005), we thereby first approximate the Tukey norm as a re-weighted quadratic expression

$$\rho_\mathrm{tuk}\big(\lVert \boldsymbol{x}_i' - \boldsymbol{x}_i(\boldsymbol{\theta}) \rVert_2 \big) \approx w_i \lVert \boldsymbol{x}_i' - \boldsymbol{x}_i(\boldsymbol{\theta}) \rVert_2^2, \tag{4.15}$$

with

$$w_i = \frac{\rho_\mathrm{tuk}(r_i)}{r_i}, \quad r_i = \lVert \boldsymbol{x}_i' - \boldsymbol{x}_i \rVert_2^2, \tag{4.16}$$

where weights w_i are considered to be constant and are recalculated in each iteration of the optimization. For their calculation, residuals r_i are evaluated at $\boldsymbol{\theta} = \boldsymbol{0}$. Subsequently, given those simplifications, the gradient vector and the Hessian matrix can be derived by applying the chain rule to the logarithm of the PDF as follows

$$\boldsymbol{g}_\mathrm{t} = \sum_{i=0}^{n} -\frac{w_i}{2{\sigma_\mathrm{t}}^2} \frac{\partial r_i}{\partial \boldsymbol{x}_i} \frac{\partial \boldsymbol{x}_i}{\partial {}_\mathrm{C}\boldsymbol{X}_i} \frac{\partial {}_\mathrm{C}\boldsymbol{X}_i}{\partial \boldsymbol{\theta}} \bigg|_{\boldsymbol{\theta}=\boldsymbol{0}}, \tag{4.17}$$

$$\boldsymbol{H}_\mathrm{t} \approx \sum_{i=0}^{n} -\frac{w_i}{2{\sigma_\mathrm{t}}^2} \left(\frac{\partial \boldsymbol{x}_i}{\partial {}_\mathrm{C}\boldsymbol{X}_i} \frac{\partial {}_\mathrm{C}\boldsymbol{X}_i}{\partial \boldsymbol{\theta}} \right)^\top \frac{\partial^2 r_i}{\partial {\boldsymbol{x}_i}^2} \left(\frac{\partial \boldsymbol{x}_i}{\partial {}_\mathrm{C}\boldsymbol{X}_i} \frac{\partial {}_\mathrm{C}\boldsymbol{X}_i}{\partial \boldsymbol{\theta}} \right) \bigg|_{\boldsymbol{\theta}=\boldsymbol{0}}, \tag{4.18}$$

where second-order partial derivatives with respect to $\boldsymbol{x}_i$ are neglected. Starting from Eq. (4.13) and using the pinhole camera model from Eq. (3.21), we calculate the required partial derivatives as

$$\frac{\partial \boldsymbol{x}_i}{\partial {}_\mathrm{C}\boldsymbol{X}_i} = \frac{1}{{}_\mathrm{C}{Z_i}^2} \begin{bmatrix} {}_\mathrm{C}Z_i f_x & 0 & -{}_\mathrm{C}X_i f_x \\ 0 & {}_\mathrm{C}Z_i f_y & -{}_\mathrm{C}Y_i f_y \end{bmatrix}, \tag{4.19}$$

$$\frac{\partial {}_\mathrm{C}\boldsymbol{X}_i}{\partial \boldsymbol{\theta}} = {}_\mathrm{C}\boldsymbol{R}_\mathrm{M} \begin{bmatrix} -[{}_\mathrm{M}\boldsymbol{X}_i]_\times & \boldsymbol{I}_3 \end{bmatrix}. \tag{4.20}$$

Also, for the residual r_i, the following first- and second-order derivatives can be defined

$$\frac{\partial r_i}{\partial \boldsymbol{x}_i} = \begin{bmatrix} 2\,(x_i - x_i') & 2\,(y_i - y_i') \end{bmatrix}, \tag{4.21}$$

$$\frac{\partial^2 r_i}{\partial {\boldsymbol{x}_i}^2} = 2\,\boldsymbol{I}_2. \tag{4.22}$$

Finally, with the derived gradient vector and Hessian matrix, our multi-modality tracking approach is able to consider texture information.

4.2.4. Multi-region Tracking

In the following, we build on the sparse region approach presented in Chapter 3 and extend it to multi-region tracking. This not only has the advantage that additional information is considered but also facilitates better statistical modeling of individual regions. In our approach, we consider each region on the object's surface independently. This means that only the respective object surface is regarded as foreground while everything else is defined as background. The technique is very similar to conventional region-based tracking, which only differentiates between the object silhouette and the background. Consequently, we can directly adopt the probabilistic formulation presented in Chapter 3 and simply deploy one modality per region.

To represent the geometry of individual regions, we again use sparse viewpoint models. As before, they store precomputed 3D contour points and normal vectors for multiple views all around the object. In addition, the sparse viewpoint model, again, contains the distances along the normal vector for which the foreground and background are not interrupted by each other. This information is required during tracking to ensure that only one transition along each correspondence line exists. In summary, the sparse viewpoint model thus contains the same information as before. However, since multiple connecting geometries are considered for multi-region tracking, the generation process has to be adapted.

In contrast to single-region tracking, sparse viewpoint models for multi-region tracking need to consider individual surface patches instead of the entire object. An example of this is shown in Fig. 4.2. The geometry of surface patches is defined in separate meshes, which are created by the user. During model generation, those meshes are rendered from multiple views all around the object to produce silhouette and depth images. For a particular surface patch that models a region, the silhouette image shows a distinct pixel value. This allows a simple extraction of the required contour. For the extracted contour, it is important to validate points and reject view-dependent segments caused by shadows from neighboring geometries. An example of this is shown in Fig. 4.3. For the validation, one simply compares the depth value of each contour point with that of neighboring pixels from other regions. If the depth of those neighboring pixels is smaller than what can be explained by a typical surface gradient, the contour point is rejected.

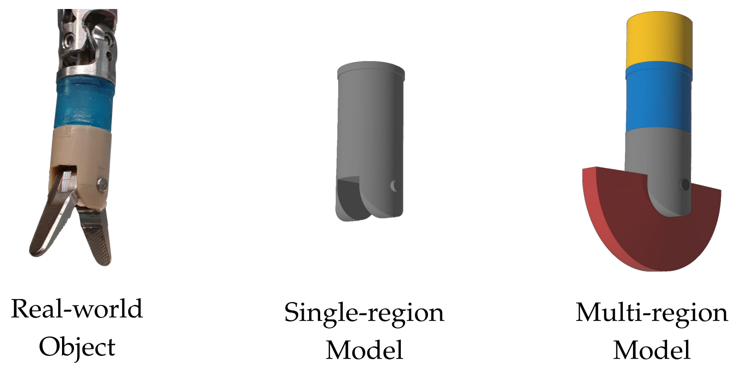


Figure 4.2.: Region models of a robotic end effector. The image in the middle shows the 3D model that is used for single-region tracking. In the image on the right, the geometry is separated into blue and gray surface patches that are employed for multi-region tracking. In addition, a fixed yellow cylinder for the gimbal joint and a red shell geometry for the movable pliers are used to explicitly consider neighboring structures.

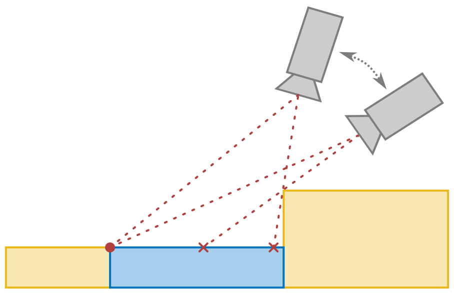


Figure 4.3.: Contour validation for a blue region neighbored by two yellow regions. The left contour point, which is shown as a circle, is invariant to small camera changes and is therefore accepted. The point on the right, which is marked by a cross, emerges from an elevated edge of a neighboring geometry. It is highly view-dependent and is therefore rejected.

In addition to *fixed* neighboring regions, we also consider *movable* external bodies for which the pose relative to the tracked surface can change. Also, we take into account *same-region* geometries that model color statistics similar to the main region. Both cases can be useful for the tracking of kinematic structures, such as robots that consist of multiple bodies. For *movable* geometries, one simply needs to create a shell mesh that covers the entire volume that can be occupied. An example is shown in Fig. 4.2, where a red mesh considers movable pliers. The geometry is used during the validation to reject areas of the contour that could become occluded. Also, in the case of *same-region* surfaces, it does not make sense to deploy correspondence lines if regions with similar color statistics neighbor the main region. Affected contour areas are thus also rejected.

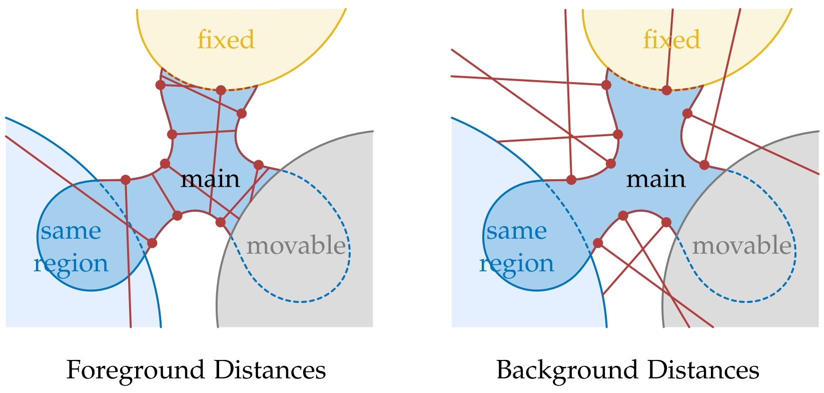


Figure 4.4.: Example for the validation of contour points and the computation of foreground and background distances. The valid contour is colored in red. Contour segments that are occluded by a *movable* body or that neighbor a *same-region* body are invalid. Segments on the transition to a *fixed* body are only valid if they are invariant to small view changes. For the computation of foreground distances, only the *main* and *same-region* areas can be traversed. Background distances are stopped by any surface that considers the same region as the main geometry.

Examples for both cases are shown in Fig. 4.4.

While for the validated contour, points and normal vectors can be sampled as usual, it is essential to correctly calculate foreground and background distances. This ensures that all available information is taken into account and only one transition exists along correspondence lines that are considered. Background distances, which are oriented outwards, are thereby limited either by the main region itself or by other *same-region* areas. Inwards-oriented foreground distances, on the other hand, can only traverse the main region or *same-region* bodies that are fixed. Examples for both cases are again shown in Fig. 4.4. Finally, given the extended sparse viewpoint model that was developed in this section, we are able to consider multiple object regions while, at the same time, taking into account neighboring geometries.

4.3. Implementation

The following section provides details for the implementation of the proposed modalities in the *M3T* algorithm and defines the parameters used in the evaluation. If not defined otherwise, the same values are employed in the region modality as in Section 3.4. For the amplitude and slope parameters, we adopt $\alpha_\mathrm{h} = 0.43$ and $s_\mathrm{h} = 0.5$, which were previously used for the evaluation on the *OPT* dataset. The only main changes for the region modality consider the discretization of color histograms and the number of conducted iterations. For color histograms, we decrease the number of bins from 32 to 16, which results in a total of 4096 equidistant bins. Also, we only use 4 instead of 7 iterations for the optimization. Both changes help to further improve efficiency while, at the same time, they do not significantly affect tracking quality.

For the depth modality, we use a sparse viewpoint model similar to the definition in Section 3.4.1. The main difference is that we sample points on the object's surface instead of the contour. Also, we do not require foreground and background distances. To explicitly consider occlusions known during model generation, we include the corresponding geometry in the rendering of each viewpoint. Points are then only sampled on the main object's silhouette, which is not occluded. As for the region modality, the model allows for a highly-efficient representation of the object geometry and directly provides the required surface points $\boldsymbol{X}_i$ and normal vectors $\boldsymbol{N}_i$.

To find correspondence points $\boldsymbol{P}_i$ for the depth modality, image values on a quadratic grid with a stride of 5 mm and a radius equal to the correspondence threshold r_d are considered potential candidates. Both parameter values are projected from meters into pixels based on the depth of 3D points $\boldsymbol{X}_i$. Correspondence points with a distance bigger than the threshold r_d are rejected. Note that both the standard deviation σ_d and the threshold r_d are adjusted for each iteration. This allows us to define our confidence in the data and the range in which depth information is considered. Because characteristics such as depth image quality and frame-to-frame pose differences depend on the sequence, we adjust the parameters for every dataset and provide them in the evaluation section. Finally, like for the region modality, individual model points are subject to occlusion. We, therefore, adopt the same strategy as in Section 3.4.4 and use information from depth images and renderings to reject occluded points.

For the texture modality, we integrate the *BRISK* (Leutenegger et al. 2011), *ORB* (Rublee et al. 2011), *FREAK* (Alahi et al. 2012), *SIFT* (Lowe 2004), and *DAISY* (Tola et al. 2010) descriptors from the *OpenCV* library. Note that *FREAK* and *DAISY* are used in combination with the highly-efficient *ORB* detector. All other algorithms provide their own detection method. In general, we found that while *SIFT* achieves a slightly higher score, *ORB* provides the best trade-off between quality and efficiency. A detailed comparison is given in Section 4.4.4. For *ORB*, a maximum of 300 feature points are retrieved, and 3 scale levels that differ by a factor of 1.2 are used. Also, *SIFT* is run with 3 octave layers, a contrast threshold of 0.04, an edge threshold of 10, and a sigma value of 0.7. For parameter values of other descriptors, please refer to the source code.

During matching, we use the Hamming distance for binary descriptors and the euclidean norm for *SIFT* and *DAISY*. A match is valid if the distance ratio to the second-best match is smaller than 0.7. As input to the detector, we use a rectangular image crop based on the previous pose estimate. If the full object is visible in the image, the crop is scaled to a defined size of $200 \times 200\,\text{px}$. A frame is considered a keyframe if the orientation difference to the previous keyframe is bigger than $10°$. In the evaluation, a Tukey norm constant of $c = 20\,\text{px}$ is used. The standard deviation σ_t depends on the dataset and will be defined later. Finally, to check if points from a keyframe are occluded, we again use the strategy from Section 3.4.4.

For multi-region tracking, our implementation does not require any new equations

Table 4.1.: Geometries featured in renderings for the contour generation, the validation of contour points, as well as the computation of foreground and background distances. The + and × symbols indicate the rendering of different pixel values. For the − symbol, the background value is rendered.

Rendering	Main	Fixed	Movable	Fixed Same-region	Movable Same-region
Contour Generation	+	×			
Occlusion Check	−	−	+		
Same-region Check	−	−		+	+
Foreground Distance	+	−	−	+	
Background Distance	+	−		+	+

or parameters. Instead, only the generation of the sparse viewpoint model is extended, with independent modalities taking into account each region. The proposed modifications of the sparse viewpoint model consider the validation of contour points and the computation of foreground and background distances. Multiple renderings are thereby created for each view. An overview of the considered geometries and the rendered pixel values are provided in Table 4.1. First, the *contour generation* rendering is used both to extract the contour of the *main* geometry and to check the depth values of neighboring *fixed* geometries. For the *occlusion check*, one discards contour points that lie on the corresponding rendered silhouette. In contrast, for the *same-region check*, points with neighboring pixels that are not the background are rejected. Finally, for the computation of *foreground distances* and *background distances*, one proceeds along the respective line directions until pixel values change or the image ends.

4.4. Evaluation

In this section, we present an extensive evaluation of our approach on the *YCB-Video* (Xiang et al. 2018), *OPT* (Wu et al. 2017), and *Choi* (Choi and Christensen 2013) datasets. This allows us to assess our implementation's robustness, accuracy, and efficiency compared to the state of the art in 3D object tracking. We thereby not only evaluate the full configuration of our *M3T* tracker with all modalities but, in addition, consider geometry-based tracking with a single region and depth modality. This allows us to assess the performance that can be expected for the tracking of textureless objects. In multiple ablation studies, we compare different feature descriptors, demonstrate the importance of individual components of our method, and evaluate the algorithm's convergence. Finally, we also compare our approach to global 6DoF pose estimation methods and assess results for highly-efficient pose refinement. Videos which show the

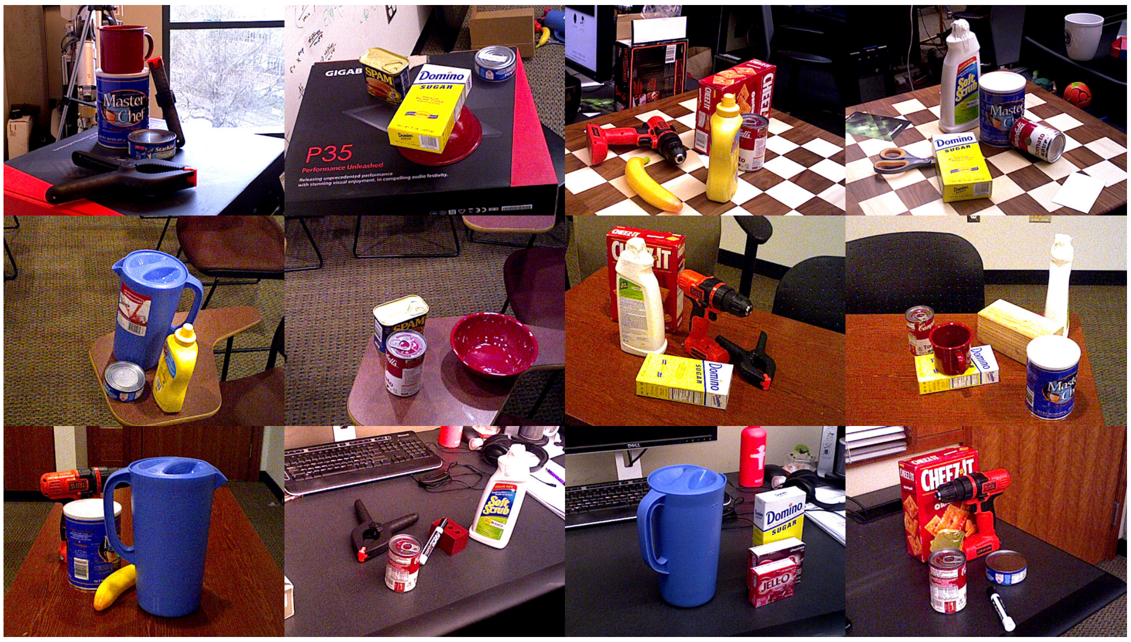


Figure 4.5.: Images from all 12 sequences of the *YCB-Video* dataset (Xiang et al. 2018) that are used for the evaluation.

performance of current and previous versions of our approach are available online. [1,2]

4.4.1. YCB-Video Dataset

The *YCB-Video* dataset (Xiang et al. 2018) uses 21 *Yale-CMU-Berkeley (YCB)* objects (Calli et al. 2015) and considers 12 sequences with a total of 2949 keyframes for evaluation. It contains both color and depth images. Because it includes additional training sequences, it is very popular with deep learning-based methods. Example images for each of the 12 sequences are shown in Fig. 4.5. For each frame, the dataset contains ground-truth poses that were obtained using global refinement on the depth data. Based on those poses, the average-distance (ADD) error e_{ADD} and the average-shortest-distance (ADD-S) error $e_{\text{ADD-S}}$ are used for evaluation. The two error metrics were proposed by Hinterstoisser et al. (2013) and are defined as follows

$$e_{\text{ADD}} = \frac{1}{n}\sum_{i=1}^{n} \left\| \left({}_{\text{M}}\tilde{\boldsymbol{X}}_i - {}_{\text{M}}\boldsymbol{T}_{\text{M}_{\text{GT}}}\, {}_{\text{M}}\tilde{\boldsymbol{X}}_i \right)_{3\times 1} \right\|_2, \tag{4.23}$$

$$e_{\text{ADD-S}} = \frac{1}{n}\sum_{i=1}^{n} \min_{j\in[n]} \left\| \left({}_{\text{M}}\tilde{\boldsymbol{X}}_i - {}_{\text{M}}\boldsymbol{T}_{\text{M}_{\text{GT}}}\, {}_{\text{M}}\tilde{\boldsymbol{X}}_j \right)_{3\times 1} \right\|_2, \tag{4.24}$$

[1]https://www.youtube.com/watch?v=NfNzxXupX54
[2]https://www.youtube.com/watch?v=qMr1RHCsnDk

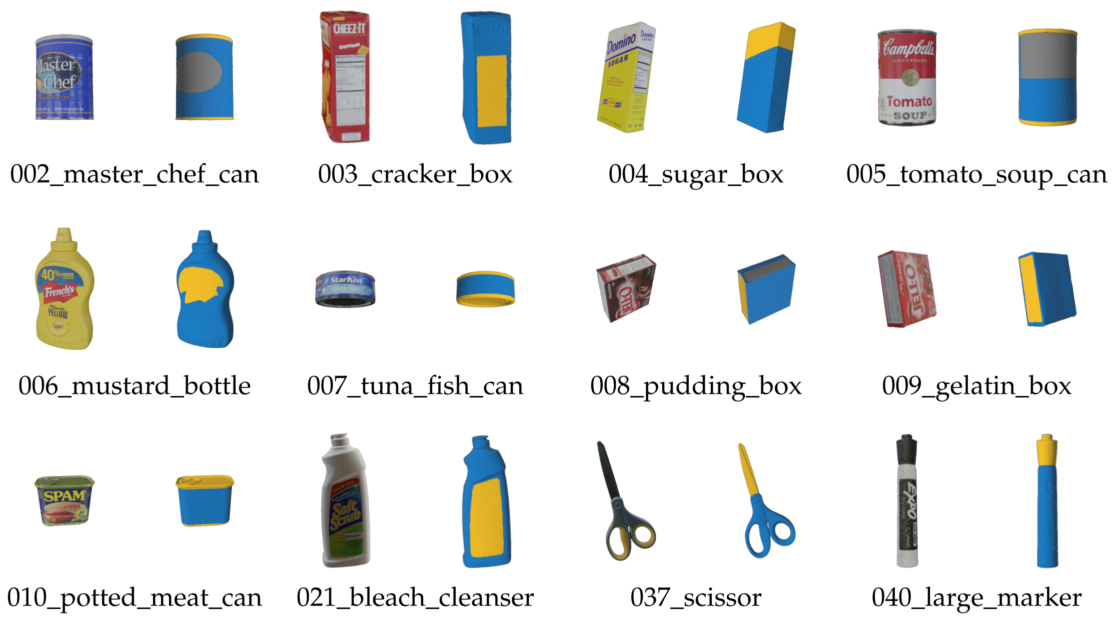


Figure 4.6.: *YCB* objects that are suitable for multi-region tracking. The right image for each object illustrates the modeled regions in different colors.

where ${}_{\mathrm{M}}\boldsymbol{T}_{\mathrm{M_{GT}}}$ is the transformation from the ground-truth (GT) to the estimated model frame, $\boldsymbol{X}_i$ is a 3D vertex of the mesh model, n is the number of vertices, and $()_{3\times1}$ denotes the first three elements of a vector. For the evaluation, ADD and ADD-S errors are used in an area-under-curve score that integrates the percentage of successful frames over a varying threshold. Based on m frames and a constant maximum threshold of $e_\mathrm{t} = 0.1\,\mathrm{m}$, scores can be calculated using the following simple equation

$$s = \frac{1}{m}\sum_{i=1}^{m} \max\left(1 - \frac{e_i}{e_\mathrm{t}}, 0\right). \tag{4.25}$$

The error e_i is thereby either the ADD error e_{ADD} or the ADD-S error $e_{\mathrm{ADD\text{-}S}}$ for frame i, depending on which area-under-curve score is calculated.

Results of the evaluation are shown in Table 4.2. For our algorithm, the parameter values $\sigma_\mathrm{r} = \{25, 15, 10\}$, $\sigma_\mathrm{d} = \{50, 30, 20\}$, $\sigma_\mathrm{t} = \{10, 10, 3\}$, $s = \{7, 4, 2\}$, and $r_\mathrm{t} = \{70, 50, 40\}$ are used, where values are given in units of pixels and millimeters. The given sets define parameters for each iteration, with the last value used for all remaining iterations. Occlusions are considered using depth images and the strategy presented in Section 3.4.4. For multi-region tracking, 12 suitable objects with distinctive regions were identified. An overview of those objects and the modeled regions is shown in Fig. 4.6. In the evaluation, our method is compared to *DeepIM* (Y. Li et al. 2018), *se(3)-TrackNet* (Wen et al. 2020), and *PoseRBPF* (Deng et al. 2021), which are all state-of-the-art deep learning-based methods. In addition, results from the particle-filter-based approach of Wüthrich et al. (2013) are also provided.

Table 4.2.: Results on the *YCB-Video* dataset (Xiang et al. 2018) with ADD and ADD-S area-under-curve scores in percent. Except for *PoseRBPF* (Deng et al. 2021), results are taken from Wen et al. (2020). For *DeepIM* (Y. Li et al. 2018), the score over all frames was adjusted to be consistent with the evaluation of other methods. The best values are in bold, while the second-best are underlined. Objects with no conclusive geometry are indicated by a *, while objects with no or very little texture are marked by a $^\diamond$.

Approach	Wüthrich		DeepIM		se(3)-TrackNet		PoseRBPF + SDF		M3T Geom. (Ours)		M3T ORB (Ours)		M3T SIFT (Ours)	
Initial Pose	GT		GT		GT		PoseCNN		GT		GT		GT	
Re-initialization	No		Yes (290)		No		Yes (2)		No		No		No	
Objects	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S
002_master_chef_can*	55.6	90.7	89.0	93.8	93.9	96.3	89.3	96.7	64.3	89.8	94.0	97.9	**94.7**	**98.0**
003_cracker_box	96.4	97.2	88.5	93.0	96.5	97.2	96.0	97.1	82.4	90.4	96.3	98.2	**96.8**	**98.4**
004_sugar_box	97.1	97.9	94.3	96.3	**97.6**	98.1	94.0	96.4	94.3	97.5	96.0	98.3	96.6	**98.5**
005_tomato_soup_can*	64.7	89.5	89.1	93.2	95.0	97.2	87.2	95.2	77.5	97.9	94.8	**98.1**	**95.2**	**98.1**
006_mustard_bottle	97.1	98.0	92.0	95.1	95.8	97.4	**98.3**	98.5	96.2	98.4	96.2	98.4	97.1	**98.7**
007_tuna_fish_can*	69.1	93.3	92.0	96.4	86.5	91.1	86.8	93.6	77.0	95.9	**94.5**	**97.2**	94.1	97.1
008_pudding_box	96.8	97.9	80.1	88.3	**97.9**	**98.4**	60.9	87.1	73.7	88.8	80.6	90.9	80.4	90.5
009_gelatin_box	97.5	98.4	92.0	94.4	97.8	98.4	**98.2**	98.6	97.2	98.8	96.9	98.8	96.8	**99.1**
010_potted_meat_can	83.7	86.7	78.0	88.9	77.8	84.2	76.4	83.5	93.0	97.2	94.8	97.9	**95.4**	**98.1**
011_banana$^\diamond$	86.3	96.1	81.0	90.5	94.9	97.2	92.8	97.7	**95.5**	**98.4**	94.1	98.2	92.8	98.2
019_pitcher_base$^\diamond$	97.3	97.7	90.4	94.7	96.8	97.5	**97.7**	98.1	97.0	98.8	97.0	98.8	97.2	**98.9**
021_bleach_cleanser	95.2	97.2	81.7	90.5	**95.9**	97.2	**95.9**	97.0	92.4	**97.4**	90.1	96.5	91.6	97.0
024_bowl$^{\star\diamond}$	30.4	97.2	38.8	90.6	80.9	94.5	34.0	93.0	70.3	**98.1**	**85.9**	97.9	84.1	97.9
025_mug$^\diamond$	83.2	93.3	83.2	92.0	91.5	96.9	86.9	96.7	95.5	**98.5**	94.1	98.3	**95.6**	98.4
035_power_drill	97.1	97.8	85.4	92.3	96.4	97.4	**97.8**	98.2	96.6	98.5	96.5	98.4	96.8	**98.7**
036_wood_block	**95.5**	96.9	44.3	75.4	95.2	96.7	37.8	93.6	93.7	97.3	94.2	**97.5**	91.7	96.7
037_scissors$^\diamond$	4.2	16.2	70.3	84.5	**95.7**	97.5	72.7	85.5	75.6	92.4	93.9	97.4	94.9	**97.6**
040_large_marker*	35.6	53.0	80.4	91.2	92.2	96.0	89.2	97.3	88.7	97.8	88.2	97.3	**94.0**	**98.1**
051_large_clamp$^\diamond$	61.2	72.3	73.9	84.1	**94.7**	96.9	90.1	95.5	93.7	97.6	94.1	97.7	93.8	**97.8**
052_extra_large_clamp$^\diamond$	**93.7**	96.6	49.3	90.3	91.7	95.8	84.4	94.1	84.4	93.7	88.0	95.2	91.8	**97.0**
061_foam_brick$^\diamond$	**96.8**	98.1	91.6	95.5	93.7	96.7	96.1	98.3	96.1	98.5	96.2	**98.6**	94.0	97.9
All Frames	78.0	90.2	82.3	91.9	93.0	95.7	87.5	95.2	86.5	96.3	93.7	97.7	**94.3**	**97.9**

For geometry-based tracking, which uses single-region and depth, *M3T* achieves state-of-the-art results with respect to the ADD-S metric, outperforming all previous algorithms. At the same time, results for the ADD score are slightly less competitive. The main reason for this is that purely geometry-based tracking cannot fully constrain the pose if the geometry is not conclusive. For example, it is impossible to predict the correct pose for a rotationally symmetric object. However, even with that handicap, geometry-based *M3T* surpasses *DeepIM* and comes very close to the results of *PoseRBPF* for the ADD metric. Finally, for the full configuration of *M3T*, which combines depth, texture, and multi-region information, results are further improved. Both with *ORB* and

Table 4.3.: Average frames per second (FPS) in hertz and hardware requirements for the CPU and GPU. Except for *PoseRBPF* (Deng et al. 2021), results are taken from Wen et al. (2020). Note that *PoseRBPF* was evaluated without pose refinement based on signed distance functions (SDFs).

Approach	Single Core	No GPU	FPS
Wüthrich	✓	✓	12.9 Hz
DeepIM		✗	12.0 Hz
se(3)-TrackNet		✗	90.9 Hz
PoseRBPF		✗	7.6 Hz
M3T Geometry (Ours)	✓	✓	1035.7 Hz
M3T ORB (Ours)	✓	✗	312.4 Hz
M3T SIFT (Ours)	✓	✗	111.7 Hz

SIFT, *M3T* outperforms the competition by a notable margin. Also, as expected, for the ADD metric, a considerable difference compared to purely geometry-based tracking without multi-region and texture can be observed. In conclusion, to the best of our knowledge, *M3T* currently reports the highest scores on the *YCB-Video* dataset.

As discussed in Section 1.2, speed and efficiency are essential in real-world robotic applications. We thus report the speed and required hardware for all algorithms in Table 4.3. As in Section 3.5, the evaluation of *M3T* was again conducted on a computer with an *Intel Core i9-11900K* CPU and a *NVIDIA RTX A5000* GPU. In comparison, other approaches were evaluated by Wen et al. (2020) using an *Intel Xeon E5-1660 v3* CPU and a *NVIDIA Tesla K40c* GPU. The obtained results highlight the outstanding efficiency of our approach. For example, while geometry-based *M3T* runs only on a single CPU core, it is more than one order of magnitude faster than the second-best algorithm *se(3)-TrackNet*, which requires a high-performance GPU. Also, although additional computational resources are required to run the full configuration of the tracker, the algorithm is still highly competitive. Utilizing a single CPU core, the tracker achieves an average framerate of 312.4 Hz for the combination with *ORB* and 111.7 Hz for *SIFT*. Also, compared to deep learning-based methods, which heavily depend on parallel computation, in our approach, the GPU is only required for the occasional reconstruction of 3D model points in keyframes. As a consequence, it remains mostly idle.

4.4.2. OPT Dataset

While the *YCB-Video* dataset features challenging sequences in real-world environments and a large number of objects, ground truth has limited accuracy, and, with a large threshold of $e_\mathrm{t} = 0.1\,\mathrm{m}$, the dataset mostly evaluates robustness. Also, images do not contain motion blur, favoring texture-based methods. The *OPT* dataset (Wu et al. 2017),

Table 4.4.: AUC scores between zero and twenty for the evaluation on the *OPT* dataset (Wu et al. 2017), comparing our approach to multiple other algorithms. Values are taken from Wu et al. (2017) and the respective publications. The best results are highlighted in bold, while the second-best values are underlined.

Approach	Soda	Chest	Ironman	House	Bike	Jet	**Avg.**
PWP3D	5.87	5.55	3.92	3.58	5.36	5.81	5.01
ElasticFusion	1.90	1.53	1.69	2.70	1.57	1.86	1.87
UDP	8.49	6.79	5.25	5.97	6.10	2.34	5.82
ORB-SLAM2	13.44	15.53	11.20	17.28	10.41	9.93	12.97
Bugaev et al. 2018	14.85	14.97	14.71	14.48	12.55	17.17	14.79
Tjaden et al. 2018	8.86	11.76	11.99	10.15	11.90	13.22	11.31
Zhong et al. 2020b	9.01	12.24	11.21	13.61	12.83	15.44	12.39
J.-C. Li et al. 2021	9.00	14.92	13.44	13.60	12.85	10.64	12.41
Huang et al. 2022	9.07	12.93	8.80	11.15	7.96	11.09	10.17
M3T Geometry (Ours)	6.28	15.84	<u>18.00</u>	17.78	16.36	16.02	15.05
M3T ORB (Ours)	<u>16.51</u>	<u>16.97</u>	**18.29**	<u>18.59</u>	**17.13**	<u>17.91</u>	**17.57**
M3T SIFT (Ours)	**16.72**	**17.37**	17.93	**18.64**	<u>16.92</u>	**17.82**	**17.57**

which was also used in Section 3.5.2, nicely complements those properties. It includes a large number of real-world sequences with color and depth images that contain significant motion blur. Accurate ground truth was obtained using a calibration board. Example images are shown in Fig. 3.13. For the evaluation, the AUC score, which was defined in Section 3.5.2, is used. Similar to the ADD and ADD-S metrics, it integrates the percentage of successfully tracked frames with respect to a variated error threshold. However, in contrast to those metrics, it also takes into account the object's size, using a threshold based on the largest distance between model vertices.

Evaluation results are reported in Table 4.4. Note that since most objects do not contain distinct regions, multi-region tracking is not considered. Also, since objects are not occluded, we do not model them explicitly. For our algorithm, the parameters $\sigma_\mathrm{r} = \{15, 5, 1.5\}$, $\sigma_\mathrm{d} = \{35, 35, 25\}$, $\sigma_\mathrm{t} = \{5, 1, 0.5\}$, $s = \{6, 4, 1\}$, and $r_\mathrm{t} = \{50, 20, 10\}$ are used. Also, in contrast to Section 3.5.2, we do not constrain the rotationally symmetric *Soda* object with a higher regularization parameter λ_r. Instead, the same parameters are used for all objects. This makes the evaluation more challenging for geometry-based tracking. In the evaluation, we compare *M3T* to state-of-the-art classical methods that use different sources of information, including region, edge, texture, and depth. The results show that our approach performs best for all six objects and improves significantly on the previously best-performing method of Bugaev et al. (2018). With respect to geometry-based tracking, we observe that, for most objects, the tracker comes very close to the full configuration of *M3T*, which includes texture. However, for the *Soda* object, results are a lot worse. The reason for this is that geometry alone is not able to constrain

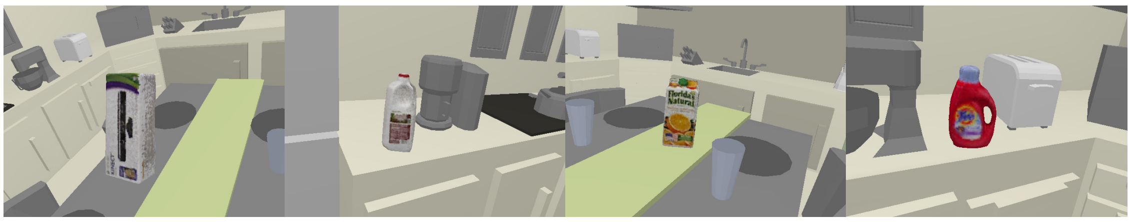


Figure 4.7.: Images from all 4 sequences of the *Choi* dataset (Choi and Christensen 2013), featuring the *Kinect Box*, *Milk*, *Orange Juice*, and *Tide* objects.

the rotationally symmetric *Soda* object. It is, therefore, not possible to predict the correct pose. This, again, highlights the simple fact that, while geometry-based tracking works well for many objects, for some cases and scenarios, it is important to consider as much information as possible.

4.4.3. Choi Dataset

To further evaluate the accuracy of our method, we use the *Choi* dataset (Choi and Christensen 2013). It features four synthetic sequences with color and depth images that provide perfect ground truth. Example images for each sequence and object are shown in Fig. 4.7. To evaluate the accuracy, root-mean-square (RMS) errors for translation and rotation parameters are calculated. One thereby uses x, y, and z differences between the translation vectors ${}_{\mathrm{C}}\boldsymbol{t}_{\mathrm{M}}$ and ${}_{\mathrm{C}}\boldsymbol{t}_{\mathrm{M_{GT}}}$, and differences between the roll, pitch, and yaw values of the rotation matrices ${}_{\mathrm{C}}\boldsymbol{R}_{\mathrm{M}}$ and ${}_{\mathrm{C}}\boldsymbol{R}_{\mathrm{M_{GT}}}$.

Results of the evaluation are shown in Table 4.5. In the evaluation, we only consider geometry-based *M3T*. The main reason is that texture is of low quality for the rendered objects, showing significant blur and no sharp edges. Also, because frame-to-frame motion is small and geometries have no local ambiguities, one can expect that the performance of the full configuration of *M3T* is very similar. Since no significant occlusions occur, we do not consider occlusion handling. For our algorithm, the parameters $\sigma_{\mathrm{r}} = \{5\}$, $\sigma_{\mathrm{d}} = \{10, 1\}$, $s = \{2, 1\}$, and $r_{\mathrm{t}} = \{10\}$ are used. In the evaluation, we compare to the particle-based algorithms of Choi and Christensen (2013) and Krull et al. (2015). Also, we include the learning-based method of Tan et al. (2017), the region- and depth-based approach of Kehl et al. (2017), and the deep learning-based method of Manhardt et al. (2018). Our algorithm achieves the highest accuracy for almost all parameters and performs best on average. Note, however, that since the dataset features perfect depth and uncluttered color images, results have to be considered as an upper bound. Nevertheless, the experiments demonstrate that, with good enough data, our approach is highly accurate.

Table 4.5.: Mean RMS errors for translation and rotation parameters on the *Choi* dataset (Choi and Christensen 2013), given in units of millimeters and degrees. Results are from the respective papers. The best values are highlighted in bold, while the second-best results are underlined.

Approach		Choi	Krull	Tan	Kehl	Manhardt	M3T (Ours)
Kinect Box	X	1.84	0.83	<u>0.15</u>	0.76	1.46	**0.05**
	Y	2.23	1.67	<u>0.19</u>	1.09	2.28	**0.14**
	Z	1.36	0.79	<u>0.09</u>	0.38	10.61	**0.02**
	Roll	6.41	1.11	<u>0.09</u>	0.17	1.84	**0.02**
	Pitch	0.76	0.55	<u>0.06</u>	0.18	2.09	**0.02**
	Yaw	6.32	1.04	<u>0.04</u>	0.20	1.23	**0.02**
Milk	X	0.93	0.51	<u>0.09</u>	0.64	3.89	**0.04**
	Y	1.94	1.27	<u>0.11</u>	0.59	4.25	**0.07**
	Z	1.09	0.62	<u>0.08</u>	0.24	57.68	**0.03**
	Roll	3.83	2.19	**0.07**	0.41	38.74	<u>0.12</u>
	Pitch	1.41	1.44	<u>0.09</u>	0.29	27.62	**0.04**
	Yaw	3.26	1.90	**0.06**	0.42	42.68	<u>0.11</u>
Orange Juice	X	0.96	0.52	<u>0.11</u>	0.50	0.65	**0.03**
	Y	1.44	0.74	<u>0.09</u>	0.69	0.69	**0.03**
	Z	1.17	0.63	<u>0.09</u>	0.17	6.49	**0.01**
	Roll	1.32	1.28	<u>0.08</u>	0.12	1.50	**0.04**
	Pitch	0.75	1.08	<u>0.08</u>	0.20	0.68	**0.03**
	Yaw	1.39	1.20	<u>0.08</u>	0.19	0.39	**0.05**
Tide	X	0.83	0.69	<u>0.08</u>	0.34	1.74	**0.02**
	Y	1.37	0.81	<u>0.09</u>	0.49	0.74	**0.03**
	Z	1.20	0.81	<u>0.07</u>	0.18	10.71	**0.01**
	Roll	1.78	2.10	<u>0.05</u>	0.15	1.78	**0.03**
	Pitch	1.09	1.38	<u>0.12</u>	0.39	1.64	**0.04**
	Yaw	1.13	1.27	<u>0.05</u>	0.37	0.80	**0.03**
Mean Translation		1.36	0.82	<u>0.10</u>	0.51	8.43	**0.04**
Mean Rotation		2.45	1.38	<u>0.07</u>	0.26	10.08	**0.04**

4.4.4. Ablation Studies

In the following, we compare different feature descriptors, demonstrate the importance of individual components of our method, and assess the algorithm's convergence. For feature descriptors and detectors, many approaches exist. To study their effect on our algorithm's tracking quality and efficiency, we conduct a comparison of prominent methods on the *YCB-Video* (Xiang et al. 2018) and *OPT* (Wu et al. 2017) datasets. Results of the evaluation are reported in Table 4.6. The experiments demonstrate that while

Table 4.6.: Comparison of *M3T* with different feature descriptors on the *YCB-Video* (Xiang et al. 2018) and *OPT* (Wu et al. 2017) datasets. AUC scores are between 0 and 20. ADD and ADD-S area-under-curves scores are given in percent.

Dataset	OPT	YCB-Video		
Features	AUC	ADD	ADD-S	FPS
M3T BRISK	17.16	<u>94.2</u>	<u>97.8</u>	121.2
M3T ORB	**17.57**	93.7	97.7	<u>312.4</u>
M3T FREAK	15.84	91.7	96.9	**400.5**
M3T SIFT	**17.57**	**94.3**	**97.9**	111.7
M3T DAISY	16.78	93.6	97.6	204.5

SIFT (Lowe 2004) achieves the highest scores, *BRISK* (Leutenegger et al. 2011) and *ORB* (Rublee et al. 2011) also deliver excellent results. At the same time, *FREAK* (Alahi et al. 2012) achieves the best runtime. However, unfortunately, tracking quality is significantly worse. Given the small difference for AUC, ADD, and ADD-S scores and the large difference in runtime between *ORB* and *SIFT*, we believe that, in most cases, *ORB* provides the best trade-off between quality and efficiency. In the following chapter, we will therefore use *ORB* as our default descriptor and detector.

In addition to the effect of different descriptors, we want to assess the importance of developed modalities, as well as regularization and occlusion handling strategies. For this, we conduct an ablation study that disables individual components. Results of the evaluation are shown in Table 4.7. For the region and depth modality, our experiments show that, while the effect differs between the datasets, each of the two modalities significantly contributes to the final scores. For texture and multi-region on the *YCB-Video* dataset, this is somewhat different. In contrast to the ablation of region and depth, the obtained scores remain relatively high. The main reason is that both texture and multi-region help to overcome ambiguities in the object geometry. As a result, if one is disabled, the other can mostly compensate for the missing information. Nevertheless, for real-world applications, both techniques have their advantages and use cases. While, in our experiments, the texture modality achieves better tracking results, multi-region tracking is robust to motion blur, significantly faster, and does not require a GPU. Also, the use of each of them highly depends on object characteristics. While the texture modality performs well in the presence of local features from text or graphics, multi-region is able to incorporate information from global differences, such as distinct materials and colors. Finally, the ablation study also demonstrates the importance of occlusion handling and regularization. Especially for occlusion handling, we observe a significant difference. For regularization, values on the *Choi* and *OPT* datasets are mostly the same. However, for the more challenging sequences of the *YCB-Video* dataset, it remains essential.

Table 4.7.: Ablation study evaluating the importance of individual components of *M3T* for results on the *Choi* (Choi and Christensen 2013), *YCB-Video* (Xiang et al. 2018), and *OPT* (Wu et al. 2017) datasets. Translational and rotational errors are provided in millimeters and degrees. AUC scores are between 0 and 20. ADD and ADD-S area-under-curve scores are given in percent.

Dataset	Choi		OPT	YCB-Video		
Experiment	Trans.	Rot.	AUC	ADD	ADD-S	FPS
M3T Geom.	0.04	0.04	15.05	86.5	96.3	1035.7
M3T (ORB)	0.04	0.04	17.57	93.7	97.7	312.4
W/o Region	0.07	0.06	15.90	22.4	48.6	444.3
W/o Depth	14.75	16.05	17.40	47.5	60.9	426.1
W/o Texture	-	-	15.05	90.2	96.6	742.2
W/o Multi-region	-	-	-	92.3	96.9	354.8
W/o Occlusion Hand.	-	-	-	85.4	93.0	309.9
W/o Regularization	0.04	0.05	17.40	88.6	94.1	314.6

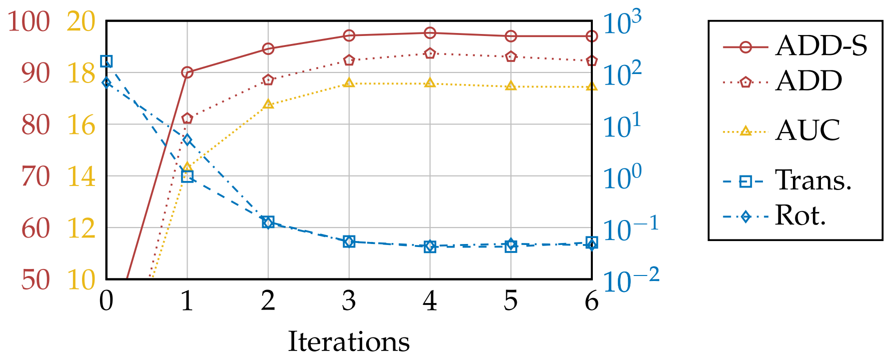


Figure 4.8.: Convergence plots for *M3T ORB* on the *YCB-Video*, *OPT*, and *Choi* datasets in red, yellow, and blue, respectively. Translational and rotational errors are provided in millimeters and degrees. AUC scores are between 0 and 20. ADD and ADD-S area-under-curve scores are given in percent.

Finally, we also want to assess the performance of our algorithm with respect to the number of iterations conducted by the optimization. Note that we thereby consider the main iterations as depicted in Algorithm 3.1, which each performs a correspondence search and two Newton steps. Convergence plots that provide average results for the considered datasets are provided in Fig. 4.8. The plots demonstrate that with only three iterations, accurate results are obtained on all three datasets. In conclusion, independent of accuracy and robustness, the fast convergence of our algorithm ensures that a maximum of only four iterations is sufficient to obtain excellent results.

Table 4.8.: Comparison with global 6DoF pose estimation methods on the *YCB-Video* dataset (Xiang et al. 2018). ADD(S) and ADD-S area-under-curve scores are given in percent. Results for *AAEs* (Sundermeyer et al. 2018) and *CosyPose* (Labbé et al. 2020) were computed by us.[3,4] Scores for all other methods are from the respective publications. Symmetric objects for which the ADD(S) metric reports the ADD-S instead of the ADD error are indicated by a $\star$.

Approach	PoseCNN		AAEs		Dense Fusion		Cosy Pose		PVN3D		FFB6D		M3T ORB (Ours)	
(Training) Data	Real RGB		Textured 3D Model		Real RGB-D		Real RGB		Real RGB-D		Real RGB-D		Untextured 3D Model	
Objects	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S
002_master_chef_can	50.9	84.0	27.1	50.6	-	96.4	37.3	90.6	80.5	96.0	80.6	96.3	**94.0**	**97.9**
003_cracker_box	51.7	76.9	32.2	64.5	-	95.5	76.8	94.9	94.8	96.1	94.6	96.3	**96.3**	**98.2**
004_sugar_box	68.6	84.3	73.6	88.6	-	97.5	95.2	97.6	96.3	97.4	**96.6**	97.6	96.0	**98.3**
005_tomato_soup_can	66.0	80.9	72.3	84.4	-	94.6	90.5	94.6	88.5	96.2	89.6	95.6	**94.8**	**98.1**
006_mustard_bottle	79.9	90.2	77.5	90.9	-	97.2	92.7	96.5	96.2	97.5	**97.0**	97.8	96.2	**98.4**
007_tuna_fish_can	70.4	87.9	71.2	92.2	-	96.6	93.9	**97.5**	89.3	96.0	88.9	96.8	**94.5**	97.2
008_pudding_box	62.9	79.0	47.9	67.7	-	96.5	93.5	96.2	**95.7**	**97.1**	94.6	**97.1**	80.6	90.9
009_gelatin_box	75.2	87.1	74.8	82.9	-	98.1	94.1	96.1	96.1	97.7	**96.9**	98.1	**96.9**	**98.8**
010_potted_meat_can	59.6	78.5	53.6	63.3	-	91.3	75.9	84.0	88.6	93.3	88.1	94.7	**94.8**	**97.9**
011_banana	72.3	85.9	13.1	51.6	-	96.6	90.0	95.6	93.7	96.6	**94.9**	97.2	94.1	**98.2**
019_pitcher_base	52.5	76.8	77.6	91.7	-	97.1	94.0	97.3	96.5	97.4	96.9	97.6	**97.0**	**98.8**
021_bleach_cleanser	50.5	71.9	42.0	62.6	-	95.8	82.1	92.7	93.2	96.0	**94.8**	**96.8**	90.1	96.5
024_bowl*	69.7	69.7	79.1	79.1	-	88.2	87.8	87.8	90.2	90.2	96.3	96.3	**97.9**	**97.9**
025_mug	57.7	78.0	58.0	80.9	-	97.1	87.8	94.9	**95.4**	97.6	94.2	97.3	94.1	**98.3**
035_power_drill	55.1	72.8	61.2	77.9	-	96.0	89.7	95.1	95.1	96.7	95.9	97.2	**96.5**	**98.4**
036_wood_block*	65.8	65.8	55.2	55.2	-	89.7	80.5	80.5	90.4	90.4	92.6	92.6	**97.5**	**97.5**
037_scissors	35.8	56.2	0.8	7.0	-	95.2	67.6	81.5	92.7	96.7	**95.7**	**97.7**	93.9	97.4
040_large_marker	58.0	71.4	55.6	67.6	-	**97.5**	84.3	93.1	**91.8**	96.7	89.1	96.6	88.2	97.3
051_large_clamp*	49.9	49.9	72.2	72.2	-	72.9	91.3	91.3	93.6	93.6	96.8	96.8	**97.7**	**97.7**
052_extra_large_clamp*	47.0	47.0	59.5	59.5	-	69.8	75.7	75.7	88.4	88.4	**96.0**	**96.0**	95.2	95.2
061_foam_brick*	87.8	87.8	56.2	56.2	-	92.5	94.7	94.7	96.8	96.8	97.3	97.3	**98.6**	**98.6**
All Frames	60.0	75.9	57.5	72.8	-	93.1	83.8	92.6	91.8	95.5	92.7	96.6	**94.7**	**97.7**

4.4.5. Global Pose Estimation

Given the strong results of global detection and pose estimation algorithms, the question arises whether 3D object tracking is even necessary. To answer this question, we conduct a comparison with state-of-the-art methods on the *YCB-Video* dataset (Xiang et al. 2018). Results are provided in Table 4.8. The evaluation compares our approach to *PoseCNN* (Xiang et al. 2018), *Augmented Autoencoders (AAEs)* (Sundermeyer et al. 2018), *DenseFusion* (C. Wang et al. 2019), *CosyPose* (Labbé et al. 2020), *PVN3D* (He et al. 2020), and *FFB6D* (He et al. 2021). Both *AAEs* and *CosyPose* are evaluated by us using source code that is publicly available.[3,4] To ensure compatibility with reported results from *PVN3D* and *FFB6D*, we adopt the ADD(S) score. It is a combined metric that uses ADD-S values for symmetric objects and ADD errors in all other cases.

The comparison demonstrates that *M3T* is able to outperform all global 6DoF pose estimation methods by a considerable margin, achieving better results than the currently best-performing algorithm *FFB6D*. This is especially remarkable since *FFB6D* trains on a large amount of pose-annotated real-world data that originates from a similar pose distribution. For many applications, such data is not available. In contrast, *M3T* only requires textureless 3D models and no training data. One reason for the superior performance is that 3D object tracking considers the pose on a frame-to-frame basis. Starting from a nearby pose and taking advantage of temporal information simplifies the task significantly compared to global estimation. In addition to quality, efficiency and speed are other advantages of 3D object tracking. While *FFB6D* depends on a high-end GPU and reports a runtime of 75 ms (He et al. 2021), *M3T* hardly uses the GPU and requires only 3.2 ms per frame. This is more than one order of magnitude faster. In conclusion, the results show that while tracking by detection is possible, for many real-world applications, it is not the most sensible solution. Given the high efficiency and good performance of *M3T*, in our opinion, it is best to rely on continuous 3D object tracking for local pose updates while using global 6DoF pose estimation for initialization and to ensure long-term consistency.

4.4.6. Pose Refinement

Given that *M3T* is a local optimization method, it can also be used for pose refinement. In the following, we assess the performance of *M3T* for this application. We thereby try to improve the predictions of *PoseCNN* (Xiang et al. 2018), *AAEs* (Sundermeyer et al. 2018), and *CosyPose* (Labbé et al. 2020) and compare results on the *YCB-Video* dataset (Xiang et al. 2018). For the algorithm, we only use multi-region and depth. The reason for this is that our texture modality only considers the relative pose between frames and does not provide any absolute constraints on the pose. Also, since temporal information is not available for pose refinement, color histograms of region modalities are initialized at each iteration before correspondences are established. While the resulting histograms do not show the same quality as for tracking, they still include useful information that helps the algorithm to converge.

Depending on the global 6DoF pose estimation method, errors along the principal axis can be relatively large. To cope with those bigger errors, we use the parameters $\sigma_\mathrm{r} = \{20, 10, 10\}$, $\sigma_\mathrm{d} = \{300, 100, 25\}$, $s = \{5, 5, 3\}$, $r_\mathrm{t} = \{300, 300, 100\}$, and $\lambda_\mathrm{t} = 1000$, and conduct 7 instead of 4 iterations. For efficiency, strides are increased from 5 mm to 10 mm. Also, we do not use occlusion handling. Note that while more iterations are performed and a bigger area is considered, omitting the texture modality helps to save

[3] `https://github.com/DLR-RM/AugmentedAutoencoder`
[4] `https://github.com/ylabbe/cosypose`

Table 4.9.: Refined and unrefined results on the *YCB-Video* dataset (Xiang et al. 2018). ADD and ADD-S area-under-curve scores are given in percent. For *PoseCNN* (Xiang et al. 2018) with *MH ICP*, results are from the corresponding publication. To evaluate the refinement, predictions for *PoseCNN* are taken from the *YCB_Video_toolbox*[5] while poses for *AAE* (Sundermeyer et al. 2018) and *CosyPose* (Labbé et al. 2020) are computed using available source code[3,4].

Approach	PoseCNN						AAEs				CosyPose			
Refinement	MH ICP		-		M3T (Ours)		-		M3T (Ours)		IM		IM + M3T (Ours)	
Objects	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S
002_master_chef_can	69.0	95.8	50.0	84.6	**70.8**	96.7	27.1	50.6	47.6	88.4	37.3	90.6	41.1	**97.6**
003_cracker_box	**80.7**	91.8	53.0	77.5	75.8	89.9	32.2	64.5	72.0	90.9	76.8	94.9	78.6	**96.4**
004_sugar_box	**97.2**	**98.2**	68.3	84.5	91.6	96.1	73.6	88.6	89.0	96.7	95.2	97.6	92.7	96.7
005_tomato_soup_can	81.6	94.5	66.1	81.4	86.4	93.6	72.3	84.4	85.7	92.3	90.5	94.6	**92.4**	**95.8**
006_mustard_bottle	**97.0**	**98.4**	80.8	91.1	94.8	97.9	77.5	90.9	89.1	97.9	92.7	96.5	95.4	98.0
007_tuna_fish_can	83.1	97.1	70.5	88.4	84.8	96.3	71.2	92.2	80.4	97.0	**93.9**	**97.5**	93.2	96.9
008_pudding_box	**96.6**	**97.9**	62.2	79.3	83.9	91.9	47.9	67.7	69.8	90.3	93.5	96.2	82.3	91.3
009_gelatin_box	**98.2**	**98.8**	74.9	87.7	96.9	98.6	74.8	82.9	96.5	98.5	94.1	96.1	96.0	98.3
010_potted_meat_can	**83.8**	**92.8**	59.3	78.8	77.4	88.8	53.6	63.3	68.7	75.7	75.9	84.0	76.6	84.2
011_banana	91.6	96.9	72.3	86.3	87.9	95.6	13.1	51.6	28.9	68.5	90.0	95.6	**94.4**	**98.0**
019_pitcher_base	**96.7**	97.8	52.9	77.6	95.0	98.1	77.6	91.7	91.8	97.8	94.0	97.3	96.1	**98.3**
021_bleach_cleanser	**92.3**	96.8	50.2	71.7	87.1	95.3	42.0	62.6	76.6	92.0	82.1	92.7	88.9	**97.3**
024_bowl	17.5	78.3	3.0	69.6	8.4	82.9	17.3	79.1	23.2	82.4	**34.5**	87.8	33.9	**88.5**
025_mug	81.4	95.1	58.4	78.8	88.5	96.6	58.0	80.9	89.5	97.6	87.8	94.9	**95.2**	**98.3**
035_power_drill	**96.9**	**98.0**	55.2	73.2	93.7	97.2	61.2	77.9	92.7	97.0	89.7	95.1	95.3	97.9
036_wood_block	**79.2**	90.5	26.4	64.3	48.4	77.3	1.6	55.2	7.4	85.1	24.8	80.5	33.2	**91.7**
037_scissors	78.4	**92.2**	34.8	55.9	56.5	77.6	0.8	7.0	1.6	8.5	67.6	81.5	**82.5**	90.7
040_large_marker	85.4	97.2	58.2	71.9	84.8	95.8	55.6	67.6	87.4	94.5	84.3	93.1	**91.1**	**97.7**
051_large_clamp	**52.6**	75.4	24.6	50.1	52.2	76.4	32.8	72.2	43.7	85.2	40.1	91.3	41.3	**95.1**
052_extra_large_clamp	28.7	65.3	16.3	44.5	29.1	70.8	26.9	59.5	35.0	65.5	**40.2**	**75.7**	40.0	75.1
061_foam_brick	48.3	97.1	40.4	88.2	46.2	96.7	19.4	56.2	39.2	79.2	51.7	94.7	**52.6**	**97.5**
All Frames	**79.3**	93.0	53.7	76.3	77.2	92.1	50.5	72.8	70.1	88.8	76.1	92.6	78.5	**94.8**

computation. As a consequence, we obtain a runtime of 3.2 ms per frame, which is the same as for 3D object tracking with the full *M3T* algorithm.

Results of the evaluation are shown in Table 4.9. For the considered global 6DoF pose estimation methods, both refined and unrefined values are reported. Unrefined poses are obtained using source code[3,4] and predictions[5] that are publicly available. In addition, results from Xiang et al. (2018) are provided, which were obtained using an extensive *multi-hypothesis (MH) ICP* refinement that requires more than 10 s per pose (C. Wang et al. 2019). The evaluation shows that, even for the excellent results of *CosyPose*, which were already refined using an iterative matching (IM) approach, *M3T* is able to

[5]`https://github.com/yuxng/YCB_Video_toolbox`

Table 4.10.: Ablation study comparing unrefined scores with refined results of different versions of *M3T*. Values show ADD and ADD-S area-under-curve scores on the *YCB-Video* dataset (Xiang et al. 2018) in percent.

Approach	PoseCNN		AAEs		CosyPose	
Refinement	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S
M3T	77.2	92.1	70.1	88.8	78.5	94.8
W/o Multi-region	76.8	92.0	69.4	88.3	78.3	94.7
W/o Region	68.7	86.9	62.8	82.9	75.0	92.7
Unrefined	53.7	76.3	50.5	72.8	76.1	92.6

improve predictions for almost all objects. Also, while extensive *MH ICP* refinement is slightly better for the ADD score, *M3T* performs best for the ADD-S score. This is especially impressive given the huge difference in runtime for the two algorithms.

Finally, we want to ensure that *M3T* uses both depth and region information for refinement and that improvements come not only from the *ICP*-based depth modality. For this, a short ablation study is conducted, for which results are shown in Table 4.10. The obtained scores demonstrate that *M3T* is not just an inflated *ICP* approach, but that region information significantly helps to improve performance. Also, while the difference is not big, multi-region consistently helps to improve results even further. Given the excellent pose predictions and computational efficiency, we are confident that, in addition to 3D object tracking, *M3T* has many applications in pose refinement.

4.5. Conclusion

In this chapter, we extended the previously developed region approach to highly-efficient multi-modality tracking. The method fuses depth, texture, and multi-region information in a probabilistic formulation that is able to consider multiple cameras. While depth and region provide absolute information, texture modalities estimate pose changes relative to keyframes. Thanks to this relative formulation, no model texture is required, and an untextured mesh of the object is sufficient for all three modalities. This is highly advantageous for usability. It significantly reduces the effort of 3D model creation and, for example, allows us to use CAD models or geometric scans. In summary, the resulting approach is highly modular and facilitates a flexible combination of different modalities, depending on the tracked object and available sensory information.

In experiments on the *YCB-Video*, *OPT*, and *Choi* datasets, we showed that *M3T* outperforms all existing methods by a considerable margin, both with respect to robustness and accuracy. In addition, even for purely geometry-based tracking, which considers information available for textureless objects, our approach achieves state-of-the-art performance. At the same time, *M3T* shows outstanding efficiency, running at more than

1 kHz on a single CPU core for geometry-based tracking and more than 300 Hz for the full configuration with all modalities. In a detailed ablation study, we also demonstrated that the developed modalities are highly complementary and that each component contributes to the final results. In addition, we analyzed the effect of different feature descriptors and proved the algorithm's fast convergence.

Given the performance of our approach, the obtained results suggest that deep learning-based 3D object tracking techniques do not yet surpass classical methods. This is especially surprising given that such algorithms require a textured 3D model and, at the expense of high computational cost and limited real-time capability, can directly consider vast amounts of information from training data. Also, with respect to global 6DoF pose estimation, we showed that *M3T* is able to achieve similar or better results at a fraction of the computational cost. Consequently, to provide continuous feedback for real-world applications, we believe that a two-stage process that uses global pose estimation for initialization and 3D object tracking for highly-efficient updates is most sensible. In addition, experiments demonstrate that global 6DoF pose estimation results can be further improved using *M3T* for highly-efficient pose refinement.

5 Multi-body Tracking

5.1. Introduction

Kinematic structures that consist of multiple connected bodies are very common in the real world. They range from simple articulated objects to complex multi-body systems. Examples range from basic tools such as staplers, scissors, or punchers over furniture and appliances with moving doors, drawers, or flaps to mechanisms and machines typically found in industry. Most prominent in the case of robotic manipulation are robots themselves, which, in general, consist of multiple rigid bodies that are connected by joints. However, despite their relevance, most 3D object tracking methods do not consider multi-body systems. While, in theory, algorithms could track bodies independently, in most cases, the task becomes unfeasible without kinematic information. As a consequence, many tracking algorithms are not usable in applications like robotic manipulation that feature kinematic structures.

To overcome this limitation, we propose a flexible framework that allows the extension of rigid object tracking methods to kinematic structures. Our approach focuses on Newton-like optimization, which is widely used in various tracking techniques. The resulting framework allows a flexible configuration with various joints and constraints. To the best of our knowledge, it is the first that is able to model closed kinematic chains. In order to project equations from individual rigid bodies to a multi-body system, Jacobians are used. For closed kinematic chains, a novel formulation that features Lagrange multipliers is developed. Based on this unified framework, we derive the required equations for joints and constraints using the axis-angle parameterization. The developed constraint equations thereby cover the entire space of possible rotations. Also, in a detailed mathematical proof, we show that our constraint formulation converges in a single iteration and leads to an exact kinematic solution.

Based on the developed framework, we extend the approach from Chapter 4 to multi-body tracking. This results in a multi-body, multi-modality, and multi-camera tracking algorithm that we call *M3T*. It was already used in Chapters 3 and 4, where it demonstrated excellent efficiency and tracking quality for rigid objects. Example images that show *M3T* for the tracking of kinematic structures are given in Fig. 5.1. For the evaluation of our algorithm, we introduce the *Robot Tracking Benchmark (RTB)*, a highly-realistic synthetic dataset that features various robots and a large number of sequences. Based on this dataset, we conduct a wide variety of experiments that

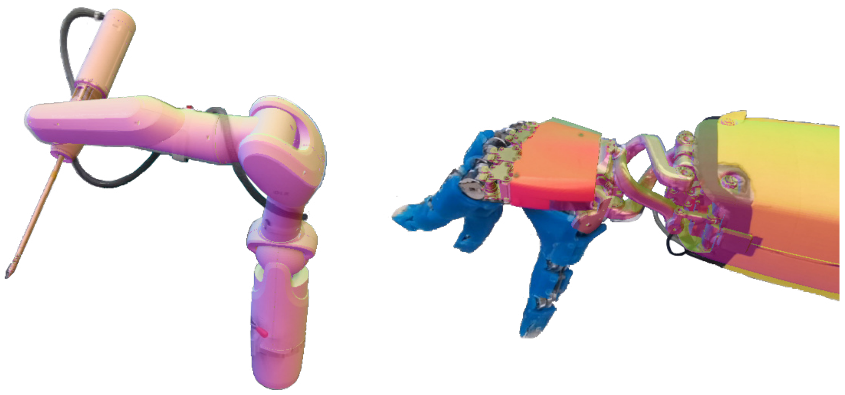


Figure 5.1.: Real-world examples that show the tracking of kinematic structures. To illustrate the predicted pose and configuration, the object model is rendered as an overlay. The image on the left shows the tracking of the medical robot *MIRO*, which consists of a chain structure with 7 degrees of freedom. In the right image, the palm, wrist, and forearm of the humanoid robot *David* are tracked. The kinematic structure includes multiple closed chains and consists of 12 individual bodies.

assess the advantages and disadvantages of different kinematic configurations. Also, the convergence of the derived constraint equations is analyzed for arbitrary cases. Finally, the evaluation demonstrates that the excellent performance of the previously developed approach for rigid objects directly transfers to kinematic structures. Note that the chapter is based on *Mb-ICG* and the corresponding publication (Stoiber et al. 2023b). The *RTB* dataset was created together with Martin Sundermeyer and Wout Boerdijk.

5.2. Framework

In the following, we derive a flexible framework that allows extending tracking algorithms from rigid objects to kinematic structures. For this, the general architecture of methods compatible with our approach is discussed. This is followed by an extension to tree-like structures and, finally, to closed kinematic structures. The provided formulation is very general and allows different pose, joint, and constraint parameterizations.

5.2.1. Rigid Objects

For the task of 3D object tracking, a wide variety of algorithms that use different sources of information have been developed. Examples of existing techniques can be found in Chapter 2. Based on the considered data, most methods derive PDFs $p(\boldsymbol{\theta})$ or energy functions $E(\boldsymbol{\theta})$ that depend on the pose variation vector $\boldsymbol{\theta}$. Given such a function, the pose that best explains the considered data is found by maximizing the probability or minimizing the energy. While different techniques have been employed for this optimization task, the following framework focuses on Newton-like methods.

As discussed in Chapters 3 and 4, Newton-like optimization iteratively estimates the variation vector and updates the object pose. The estimated pose variation $\hat{\boldsymbol{\theta}}$ is thereby calculated as follows

$$\hat{\boldsymbol{\theta}} = -\boldsymbol{H}^{-1}\boldsymbol{g}, \tag{5.1}$$

where the gradient vector $\boldsymbol{g}$ and the Hessian matrix $\boldsymbol{H}$ are the first- and second-order derivatives of the negative logarithmic probability $-\ln(p(\boldsymbol{\theta}))$ or the energy function $E(\boldsymbol{\theta})$. Because of the shorter notation, we will derive our approach for the energy function $E(\boldsymbol{\theta})$. Note, however, that with the relation $E(\boldsymbol{\theta}) = -\ln(p(\boldsymbol{\theta}))$, exactly the same equations can be used for PDFs. Finally, Eq. (5.1) is not only limited to classical Newton optimization with an exact Hessian matrix but also allows the use of approximations from Gauss-Newton or quasi-Newton methods. As a consequence, many popular algorithms such as *RAPID* (Harris and Stennett 1990), *PWP3D* (Prisacariu and Reid 2012), or the object tracker employed in *Slam++* (Salas-Moreno et al. 2013) are compatible with the developed framework.

5.2.2. Tree-like Structures

In general, tree-like kinematic structures consist of individual bodies that are connected by joints. All bodies, except for the root, have a single parent. Based on those connections, the 6DoF pose variation $\boldsymbol{\theta}$ of each body is described by joint variations $\boldsymbol{\theta}_{\mathrm{j}} \in \mathbb{R}^{n_{\mathrm{j}}}$ along the kinematic chain from the root. The number $n_{\mathrm{j}} \in [0..6]$ describes how many degrees of freedom a particular joint allows for a body relative to its parent. Finally, the combined variation of a kinematic structure with n bodies is given by all joint variations $\boldsymbol{\theta}_{\mathrm{k}}^{\top} = \left[\boldsymbol{\theta}_{\mathrm{j}0}^{\top} \dots \boldsymbol{\theta}_{\mathrm{j}n}^{\top}\right]$. Note that, in this case, the joint variation $\boldsymbol{\theta}_{\mathrm{j}0}$ directly describes the pose variation $\boldsymbol{\theta}_0$ of the root body. An example that shows the relation between individual parameters in a kinematic structure is given in Fig. 5.2.

Starting from the energy functions $E_i(\boldsymbol{\theta}_i)$ of individual bodies, the combined energy of the kinematic structure can be calculated as the sum of the energy of all bodies

$$E_{\mathrm{k}}(\boldsymbol{\theta}_{\mathrm{k}}) = \sum_{i=1}^{n} E_i(\boldsymbol{\theta}_i), \tag{5.2}$$

where variation vectors $\boldsymbol{\theta}_i$ of individual bodies are functions of the combined variation $\boldsymbol{\theta}_{\mathrm{k}}$. Given the definitions of gradient vectors and Hessian matrices, we can now write

$$\boldsymbol{g}_{\mathrm{k}}^{\top} = \left.\frac{\partial E_{\mathrm{k}}}{\partial \boldsymbol{\theta}_{\mathrm{k}}}\right|_{\boldsymbol{\theta}_{\mathrm{k}}=\boldsymbol{0}} = \sum_{i=1}^{n} \left.\frac{\partial E_i}{\partial \boldsymbol{\theta}_i}\frac{\partial \boldsymbol{\theta}_i}{\partial \boldsymbol{\theta}_{\mathrm{k}}}\right|_{\boldsymbol{\theta}_{\mathrm{k}}=\boldsymbol{0}} = \sum_{i=1}^{n} \boldsymbol{g}_i^{\top}\boldsymbol{J}_i, \tag{5.3}$$

$$\boldsymbol{H}_{\mathrm{k}} = \left.\frac{\partial^2 E_{\mathrm{k}}}{\partial {\boldsymbol{\theta}_{\mathrm{k}}}^2}\right|_{\boldsymbol{\theta}_{\mathrm{k}}=\boldsymbol{0}} \approx \sum_{i=1}^{n} \left.\frac{\partial \boldsymbol{\theta}_i}{\partial \boldsymbol{\theta}_{\mathrm{k}}}^{\top}\frac{\partial^2 E_i}{\partial {\boldsymbol{\theta}_i}^2}\frac{\partial \boldsymbol{\theta}_i}{\partial \boldsymbol{\theta}_{\mathrm{k}}}\right|_{\boldsymbol{\theta}_{\mathrm{k}}=\boldsymbol{0}} \tag{5.4}$$

$$\approx \sum_{i=1}^{n} \boldsymbol{J}_i^{\top}\boldsymbol{H}_i\boldsymbol{J}_i, \tag{5.5}$$

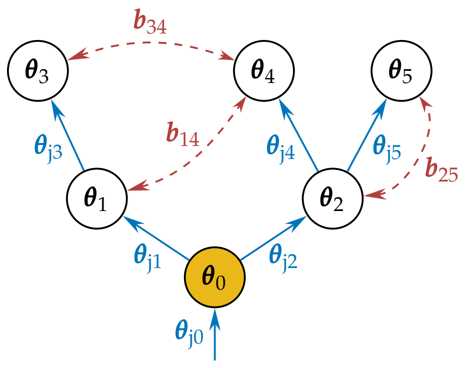


Figure 5.2.: Relation between variation parameters in a kinematic structure. The 6DoF pose variations $\boldsymbol{\theta}_0$ to $\boldsymbol{\theta}_5$ of individual bodies are illustrated as vertices in a graph. For the root body, the vertex is colored yellow. Connections to parent bodies and the associated joint variations $\boldsymbol{\theta}_{j1}$ to $\boldsymbol{\theta}_{j5}$ are indicated by blue arrows. For the root body, which does not have a parent, the joint variation $\boldsymbol{\theta}_{j0}$ directly describes the pose variation $\boldsymbol{\theta}_0$. Finally, dashed red lines indicate the constraints $\boldsymbol{b}_{14}$, $\boldsymbol{b}_{34}$, and $\boldsymbol{b}_{25}$ between bodies.

with $\boldsymbol{g}_i$ and $\boldsymbol{H}_i$ the gradient vector and the Hessian matrix of the rigid body i. Note that second-order derivatives of pose variations $\boldsymbol{\theta}_i$ with respect to $\boldsymbol{\theta}_\text{k}$ were neglected. The Jacobian matrices $\boldsymbol{J}_i$ describe how the variation of the entire kinematic structure affects the pose of individual bodies. A detailed derivation of the required body Jacobians is given in Section 5.3.2. One major difference of our formulation in comparison to the approach of Lowe (1991) is that we do not calculate the change of each measurement with respect to the combined variation $\boldsymbol{\theta}_\text{k}$. Instead, the change of a body's measurements is first computed for the minimal 6DoF pose variation $\boldsymbol{\theta}_i$ and then projected to the full kinematic structure. This has the advantage that the same gradient vectors and Hessian matrices as for single object tracking can be used. In addition, for kinematic structures with large numbers of bodies and measurements, it is more efficient.

5.2.3. Closed Kinematic Structures

To model closed kinematic structures, we start from the previously defined tree-like structures and include additional constraints. A constraint describes any relation between bodies that can be expressed by a constraint equation $\boldsymbol{b}(\boldsymbol{\theta}_\text{k}) = \boldsymbol{0}$. A simple example is a rotational joint where translational differences normal to the rotation axis have to be zero. An illustration of the topology of a tree-like kinematic structure with constraints between individual bodies is shown in Fig. 5.2. Given n constraints, the full constraint equation for the kinematic structure is written as $\boldsymbol{b}_\text{k}^\top = \begin{bmatrix}\boldsymbol{b}_0^\top \dots \boldsymbol{b}_n^\top\end{bmatrix}$. To integrate those constraints into the combined energy function $E_\text{k}(\boldsymbol{\theta}_\text{k})$, we write the following Lagrangian function

$$\mathcal{L}(\boldsymbol{\theta}_\text{k}, \boldsymbol{\lambda}) = E_\text{k}(\boldsymbol{\theta}_\text{k}) + \boldsymbol{b}_\text{k}(\boldsymbol{\theta}_\text{k})^\top \boldsymbol{\lambda}, \tag{5.6}$$

where $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers. To find a solution for $\boldsymbol{\theta}_\mathrm{k}$ that satisfies the imposed constraints and minimizes the energy function, the following equation has to be solved

$$\nabla\mathcal{L}(\boldsymbol{\theta}_\mathrm{k},\boldsymbol{\lambda}) = \begin{bmatrix} \boldsymbol{g}_\mathrm{k}(\boldsymbol{\theta}_\mathrm{k}) + \boldsymbol{B}_\mathrm{k}(\boldsymbol{\theta}_\mathrm{k})^\top\boldsymbol{\lambda} \\ \boldsymbol{b}_\mathrm{k}(\boldsymbol{\theta}_\mathrm{k}) \end{bmatrix} \overset{!}{=} \mathbf{0}, \tag{5.7}$$

where the constraint Jacobian $\boldsymbol{B}_\mathrm{k}$ is the first-order derivative of $\boldsymbol{b}_\mathrm{k}$ with respect to $\boldsymbol{\theta}_\mathrm{k}$. Detailed derivations of the constraint function and Jacobian are given in Section 5.3.3.

Analogous to the minimization of the energy function $E(\boldsymbol{\theta})$ of a single object, we want to linearize Eq. (5.7) around $\boldsymbol{\theta}_\mathrm{k} = \mathbf{0}$ and iteratively find a solution. For this, we use the first-order approximations $\boldsymbol{g}_\mathrm{k}(\boldsymbol{\theta}_\mathrm{k}) \approx \boldsymbol{g}_\mathrm{k} + \boldsymbol{H}_\mathrm{k}\boldsymbol{\theta}_\mathrm{k}$ and $\boldsymbol{b}_\mathrm{k}(\boldsymbol{\theta}_\mathrm{k}) \approx \boldsymbol{b}_\mathrm{k} + \boldsymbol{B}_\mathrm{k}\boldsymbol{\theta}_\mathrm{k}$ to derive the following linear equation

$$\begin{bmatrix} \hat{\boldsymbol{\theta}}_\mathrm{k} \\ \hat{\boldsymbol{\lambda}} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{H}_\mathrm{k} & \boldsymbol{B}_\mathrm{k}^\top \\ \boldsymbol{B}_\mathrm{k} & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{g}_\mathrm{k} \\ \boldsymbol{b}_\mathrm{k} \end{bmatrix}. \tag{5.8}$$

Similar to Eq. (5.1), the derived equation is used to estimate $\hat{\boldsymbol{\theta}}_\mathrm{k}$ and to iteratively minimize $E_\mathrm{k}(\boldsymbol{\theta}_\mathrm{k})$ while, at the same time, considering the constraints $\boldsymbol{b}_\mathrm{k}(\boldsymbol{\theta}_\mathrm{k}) = \mathbf{0}$. Note that, in general, given a positive definite Hessian $\boldsymbol{H}_\mathrm{k}$ and unique and non-contradicting constraint equations that lead to non-zero and linearly independent row vectors in the constraint Jacobian $\boldsymbol{B}_\mathrm{k}$, the matrix in Eq. (5.8) is invertible and a unique solution exists for the estimated variation vector $\hat{\boldsymbol{\theta}}_\mathrm{k}$.

Finally, in contrast to the *postimposed* method of Drummond and Cipolla (2002), which first predicts unconstrained poses and applies constraints later, our formulation directly estimates a variation vector that is compatible with constraints. In addition, the approach is not limited to velocity constraints. Together with our formulation for tree-like kinematic structures, the developed multi-body tracking framework supports an efficient minimal parameterization while simultaneously providing the flexibility to add constraints for closed chains and temporary limitations. It, therefore, combines the best of both worlds.

5.3. Parameterization

In this section, we derive all equations required to define a kinematic structure. For the pose parameterization, we focus on the axis-angle representation. Note that while tracking algorithms based on Euler angles, twist coordinates, or quaternions have also been developed, in our experience, this parameterization is the most popular. After an introduction to general mathematical concepts, Jacobian matrices and constraint equations are derived. In both cases, we allow to lock or free the rotation and translation along individual coordinate axes, facilitating various joints and constraints. Finally, we describe how to update the pose of individual bodies.

5.3.1. Preliminaries

Like in previous chapters, we use the following transformation matrix ${}_{\mathrm{A}}\boldsymbol{T}_{\mathrm{B}} \in \mathbb{SE}(3)$ to define the pose between two reference frames A and B

$$ {}_{\mathrm{A}}\boldsymbol{T}_{\mathrm{B}} = \begin{bmatrix} {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}} & {}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{B}} \\ \boldsymbol{0} & 1 \end{bmatrix}, \tag{5.9} $$

where ${}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}} \in \mathbb{SO}(3)$ is a rotation matrix and ${}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{B}} \in \mathbb{R}^3$ is a translation vector. They together describe the transformation from B to A. For the variation of poses with a minimal set of parameters, we again use the axis-angle representation for the rotation and write

$$ \boldsymbol{T}(\boldsymbol{\theta}) = \begin{bmatrix} \exp([\boldsymbol{\theta}_{\mathrm{r}}]_\times) & \boldsymbol{\theta}_{\mathrm{t}} \\ \boldsymbol{0} & 1 \end{bmatrix}, \tag{5.10} $$

where $\boldsymbol{\theta}_{\mathrm{r}} \in \mathbb{R}^3$ and $\boldsymbol{\theta}_{\mathrm{t}} \in \mathbb{R}^3$ are the rotational and translational components of the variation vector $\boldsymbol{\theta}^\top = \begin{bmatrix}\boldsymbol{\theta}_{\mathrm{r}}^\top & \boldsymbol{\theta}_{\mathrm{t}}^\top\end{bmatrix}$. The matrix $[\boldsymbol{\theta}_{\mathrm{r}}]_\times$ is the skew-symmetric cross-product matrix of $\boldsymbol{\theta}_{\mathrm{r}}$. Finally, to project the variation vector $\boldsymbol{\theta}$ from reference frame B to A, the following adjoint representation is used

$$ \mathrm{Ad}({}_{\mathrm{A}}\boldsymbol{T}_{\mathrm{B}}) = \begin{bmatrix} {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}} & \boldsymbol{0} \\ [{}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{B}}]_\times \, {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}} & {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}} \end{bmatrix}. \tag{5.11} $$

Based on those general mathematical concepts, we are able to describe kinematic structures and derive the required body Jacobians and constraint equations.

5.3.2. Body Jacobians

Each body has a defined location relative to its joint and corresponding parent. In the following, we introduce a general class of joints that allow motion along a user-defined set of rotational and translational axes. Based on the joint reference frame J, motion along these directions is modeled by the variation vector $\boldsymbol{\theta}_{\mathrm{j}}$. The full 6DoF pose variation $\boldsymbol{\theta}$ of a body's model frame M given the variation of its parent and joint is then defined as

$$ \boldsymbol{T}(\boldsymbol{\theta}) = {}_{\mathrm{M}}\boldsymbol{T}_{\mathrm{P}}\, \boldsymbol{T}(\boldsymbol{\theta}_{\mathrm{p}})\, {}_{\mathrm{P}}\boldsymbol{T}_{\mathrm{J}}\, \boldsymbol{T}(\bar{\boldsymbol{\theta}}_{\mathrm{j}})\, {}_{\mathrm{J}}\boldsymbol{T}_{\mathrm{M}}, \tag{5.12} $$

where P is the reference frame of the parent and $\boldsymbol{\theta}_{\mathrm{p}}$ the corresponding 6DoF variation vector. An illustration of the transformations is shown in Fig. 5.3. The bar above the joint variation vector $\bar{\boldsymbol{\theta}}_{\mathrm{j}}$ indicates the extension of $\boldsymbol{\theta}_{\mathrm{j}}$, which has n_{j} dimensions, to a 6DoF vector. For a typical joint that only allows movement along specified axes of the joint frame, we simply set fixed directions to zero.

To project equations of individual bodies to the multi-body system, Jacobians are used. Given a body's joint and parent, we are able to formulate a recursive strategy to calculate the Jacobian. Based on the Jacobian of the parent body $\boldsymbol{J}_{\mathrm{p}}$, we calculate

$$ \boldsymbol{J} = \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\theta}_{\mathrm{k}}}\bigg|_{\boldsymbol{\theta}_{\mathrm{k}}=\boldsymbol{0}} = \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\theta}_{\mathrm{p}}} \boldsymbol{J}_{\mathrm{p}} + \begin{bmatrix} \boldsymbol{0} & \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\theta}_{\mathrm{j}}} & \boldsymbol{0} \end{bmatrix}\bigg|_{\boldsymbol{\theta}_{\mathrm{k}}=\boldsymbol{0}}, \tag{5.13} $$

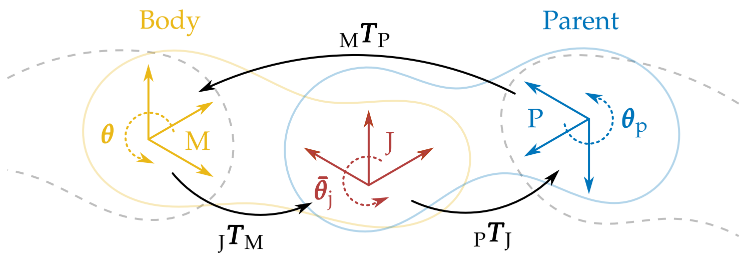

Body
Parent
$_{\mathrm{M}}\boldsymbol{T}_{\mathrm{P}}$
$\boldsymbol{\theta}$
M
J
P
$\boldsymbol{\theta}_{\mathrm{p}}$
$\bar{\boldsymbol{\theta}}_{\mathrm{j}}$
$_{\mathrm{J}}\boldsymbol{T}_{\mathrm{M}}$
$_{\mathrm{P}}\boldsymbol{T}_{\mathrm{J}}$

Figure 5.3.: Visualization of a body that is connected to its parent by a joint. Model, parent, and joint coordinate frames M, P, and J are illustrated in yellow, blue, and red, respectively. For the coordinate frames, the corresponding variation vectors $\boldsymbol{\theta}$, $\bar{\boldsymbol{\theta}}_{\mathrm{j}}$, and $\boldsymbol{\theta}_{\mathrm{p}}$, as well as the relative homogeneous transformations $_{\mathrm{J}}\boldsymbol{T}_{\mathrm{M}}$, $_{\mathrm{P}}\boldsymbol{T}_{\mathrm{J}}$, and $_{\mathrm{M}}\boldsymbol{T}_{\mathrm{P}}$, are shown.

where the partial derivative $\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\theta}_{\mathrm{j}}}$ has to be added at the correct location that corresponds to the position of $\boldsymbol{\theta}_{\mathrm{j}}$ in the vector $\boldsymbol{\theta}_{\mathrm{k}}$. The advantage of this recursive strategy is that it is not necessary to assemble the derivatives of all individual joints along the kinematic chain. Instead, we simply consider the kinematic chain using the Jacobian of the parent. For the root body, the parent Jacobian $\boldsymbol{J}_{\mathrm{p}}$ is simply zero. To calculate the Jacobian, we now only require the partial derivatives of the body variation with respect to the joint and parent variations.

Starting from Eq. (5.12) and knowing that the adjoint representation in Eq. (5.11) can be used to project variation vectors between reference frames, we write the following relations

$$\boldsymbol{\theta}|_{\boldsymbol{\theta}_{\mathrm{j}}=\mathbf{0}} = \mathrm{Ad}({}_{\mathrm{M}}\boldsymbol{T}_{\mathrm{P}})\,\boldsymbol{\theta}_{\mathrm{p}}, \tag{5.14}$$

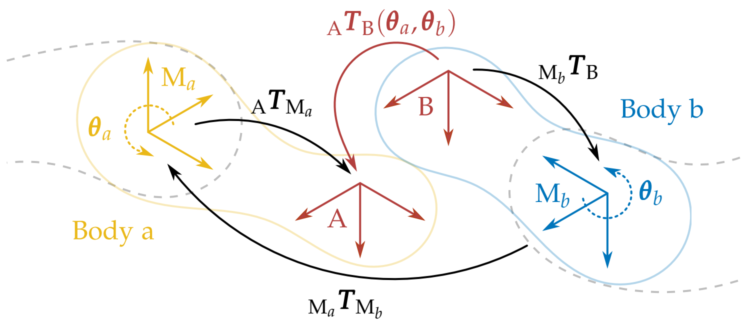
$$\boldsymbol{\theta}|_{\boldsymbol{\theta}_{\mathrm{p}}=\mathbf{0}} = \mathrm{Ad}({}_{\mathrm{M}}\boldsymbol{T}_{\mathrm{J}})\,\bar{\boldsymbol{\theta}}_{\mathrm{j}}. \tag{5.15}$$

As can be seen from those equations, the required first-order derivatives are

$$\left.\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\theta}_{\mathrm{p}}}\right|_{\boldsymbol{\theta}_{\mathrm{k}}=\mathbf{0}} = \mathrm{Ad}({}_{\mathrm{M}}\boldsymbol{T}_{\mathrm{P}}), \tag{5.16}$$

$$\left.\frac{\partial \boldsymbol{\theta}}{\partial \bar{\boldsymbol{\theta}}_{\mathrm{j}}}\right|_{\boldsymbol{\theta}_{\mathrm{k}}=\mathbf{0}} = \mathrm{Ad}({}_{\mathrm{M}}\boldsymbol{T}_{\mathrm{J}}). \tag{5.17}$$

Note that Eq. (5.17) considers the derivative with respect to the extended variation vector $\bar{\boldsymbol{\theta}}_{\mathrm{j}}$. To get the first-order derivative $\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\theta}_{\mathrm{j}}}\big|_{\boldsymbol{\theta}_{\mathrm{k}}=\mathbf{0}}$ with respect to the variation vector $\boldsymbol{\theta}_{\mathrm{j}}$, one simply assembles the columns from the partial derivative in Eq. (5.17) that are considered by $\boldsymbol{\theta}_{\mathrm{j}}$. With the derived formulation, we are able to model various joints, including revolute, prismatic, or spherical joints. For more specific joints, such as screw connections, it is typically also straightforward to derive the necessary first-order derivatives as a combination of the motion along individual axes. Instead of directly



$_{\mathrm{A}}\boldsymbol{T}_{\mathrm{B}}(\boldsymbol{\theta}_a,\boldsymbol{\theta}_b)$
M_a
$\boldsymbol{\theta}_a$
$_{\mathrm{A}}\boldsymbol{T}_{\mathrm{M}_a}$
B
$_{\mathrm{M}_b}\boldsymbol{T}_{\mathrm{B}}$
Body b
A
M_b
$\boldsymbol{\theta}_b$
Body a
$_{\mathrm{M}_a}\boldsymbol{T}_{\mathrm{M}_b}$

Figure 5.4.: Illustration of a constraint between two bodies a and b. The model coordinate frames M_a and M_b, as well as the variation vectors $\boldsymbol{\theta}_a$ and $\boldsymbol{\theta}_b$ are illustrated in yellow and blue, respectively. The constraint coordinate frames A and B associated with bodies a and b are colored in red. The relative transformation $_{\mathrm{A}}\boldsymbol{T}_{\mathrm{B}}(\boldsymbol{\theta}_a,\boldsymbol{\theta}_b)$ is calculated from $_{\mathrm{M}_b}\boldsymbol{T}_{\mathrm{B}}$, $_{\mathrm{M}_a}\boldsymbol{T}_{\mathrm{M}_b}$, and $_{\mathrm{A}}\boldsymbol{T}_{\mathrm{M}_a}$ and is considered in the constraint equation.

assembling columns from Eq. (5.17), one then simply multiplies with the joint's partial derivative $\frac{\partial\bar{\boldsymbol{\theta}}_{\mathrm{j}}}{\partial\boldsymbol{\theta}_{\mathrm{j}}}$ to get the derivative $\frac{\partial\boldsymbol{\theta}}{\partial\boldsymbol{\theta}_{\mathrm{j}}} = \frac{\partial\boldsymbol{\theta}}{\partial\bar{\boldsymbol{\theta}}_{\mathrm{j}}}\frac{\partial\bar{\boldsymbol{\theta}}_{\mathrm{j}}}{\partial\boldsymbol{\theta}_{\mathrm{j}}}$ required for the body Jacobian.

5.3.3. Constraint Equations

In the following, we derive constraints that consider rotational and translational differences between two bodies along individual axes. Differences are calculated with respect to the constraint reference frames A and B, which are defined for the corresponding bodies a and b. Constrained differences between those two frames are enforced to be zero. The relative pose between frames A and B, given the pose variation vectors $\boldsymbol{\theta}_a$ and $\boldsymbol{\theta}_b$, is thereby written as

$$_{\mathrm{A}}\boldsymbol{T}_{\mathrm{B}}(\boldsymbol{\theta}_a,\boldsymbol{\theta}_b) = {}_{\mathrm{A}}\boldsymbol{T}_{\mathrm{M}_a}\,\boldsymbol{T}(\boldsymbol{\theta}_a)^{-1}\,{}_{\mathrm{M}_a}\boldsymbol{T}_{\mathrm{M}_b}\,\boldsymbol{T}(\boldsymbol{\theta}_b)\,{}_{\mathrm{M}_b}\boldsymbol{T}_{\mathrm{B}}, \tag{5.18}$$

where M_a and M_b are the model frames of bodies a and b. An illustration of the transformation is shown in Fig. 5.4. Based on the relative pose $_{\mathrm{A}}\boldsymbol{T}_{\mathrm{B}}(\boldsymbol{\theta}_a,\boldsymbol{\theta}_b)$, we define the extended 6DoF constraint equation for all rotational and translational axes as follows

$$\bar{\boldsymbol{b}}_{ab}(\boldsymbol{\theta}_a,\boldsymbol{\theta}_b) = \begin{bmatrix} _{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}(\boldsymbol{\theta}_a,\boldsymbol{\theta}_b) \\ _{\mathrm{A}}\boldsymbol{t}_{\mathrm{B}}(\boldsymbol{\theta}_a,\boldsymbol{\theta}_b) \end{bmatrix}, \tag{5.19}$$

where $_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}$ is the rotation vector, and $_{\mathrm{A}}\boldsymbol{t}_{\mathrm{B}}$ is the translation vector of the transformation matrix $_{\mathrm{A}}\boldsymbol{T}_{\mathrm{B}}$. The rotation vector and the rotation matrix are related by the exponential map $_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}} = \exp([_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}]_\times)$. Note that, depending on the directions that should be constrained, only individual elements of the extended constraint equation $\bar{\boldsymbol{b}}_{ab}$ are used. The resulting reduced constraint equation is denoted $\boldsymbol{b}_{ab}$.

Given a constraint equation, the corresponding constraint Jacobian can be calculated using body Jacobians $\boldsymbol{J}$ that were derived in the previous section

$$\boldsymbol{B}_{ab} = \left.\frac{\partial \boldsymbol{b}_{ab}}{\partial \boldsymbol{\theta}_\mathrm{k}}\right|_{\boldsymbol{\theta}_\mathrm{k}=\boldsymbol{0}} = \left.\frac{\partial \boldsymbol{b}_{ab}}{\partial \boldsymbol{\theta}_a}\frac{\partial \boldsymbol{\theta}_a}{\partial \boldsymbol{\theta}_\mathrm{k}} + \frac{\partial \boldsymbol{b}_{ab}}{\partial \boldsymbol{\theta}_b}\frac{\partial \boldsymbol{\theta}_b}{\partial \boldsymbol{\theta}_\mathrm{k}}\right|_{\boldsymbol{\theta}_\mathrm{k}=\boldsymbol{0}} \tag{5.20}$$

$$= \left.\frac{\partial \boldsymbol{b}_{ab}}{\partial \boldsymbol{\theta}_a}\boldsymbol{J}_a + \frac{\partial \boldsymbol{b}_{ab}}{\partial \boldsymbol{\theta}_b}\boldsymbol{J}_b\right|_{\boldsymbol{\theta}_\mathrm{k}=\boldsymbol{0}}. \tag{5.21}$$

For the required partial derivatives of the extended constraint equation $\bar{\boldsymbol{b}}_{ab}$ with respect to the pose variations $\boldsymbol{\theta}_a$ and $\boldsymbol{\theta}_b$, the calculation is quite lengthy. This is especially the case for the rotational constraint, which considers the nontrivial change of the rotation vector ${}_\mathrm{A}\boldsymbol{r}_\mathrm{B}$ for the variation of the two bodies. Consequently, we provide the full derivation in Appendix E and only state the following final results

$$\left.\frac{\partial \bar{\boldsymbol{b}}_{ab}}{\partial \boldsymbol{\theta}_a}\right|_{\boldsymbol{\theta}_\mathrm{k}=\boldsymbol{0}} = \begin{bmatrix} -\boldsymbol{C}\,{}_\mathrm{A}\boldsymbol{R}_{\mathrm{M}_a} & \boldsymbol{0} \\ {}_\mathrm{A}\boldsymbol{R}_{\mathrm{M}_a}\,[{}_{\mathrm{M}_a}\boldsymbol{t}_\mathrm{B}]_\times & -{}_\mathrm{A}\boldsymbol{R}_{\mathrm{M}_a} \end{bmatrix}, \tag{5.22}$$

$$\left.\frac{\partial \bar{\boldsymbol{b}}_{ab}}{\partial \boldsymbol{\theta}_b}\right|_{\boldsymbol{\theta}_\mathrm{k}=\boldsymbol{0}} = \begin{bmatrix} \boldsymbol{C}\,{}_\mathrm{A}\boldsymbol{R}_{\mathrm{M}_b} & \boldsymbol{0} \\ -{}_\mathrm{A}\boldsymbol{R}_{\mathrm{M}_b}\,[{}_{\mathrm{M}_b}\boldsymbol{t}_\mathrm{B}]_\times & {}_\mathrm{A}\boldsymbol{R}_{\mathrm{M}_b} \end{bmatrix}. \tag{5.23}$$

To obtain the derivatives for the reduced constraint equation $\boldsymbol{b}_{ab}$, which are required in Eq. (5.21), one simply assembles the rows from Eqs. (5.22) and (5.23) that correspond to the constrained axes of coordinate frame A. Finally, given the Jacobians $\boldsymbol{B}_{ab}$ of individual constraints, the full constraint Jacobian $\boldsymbol{B}_\mathrm{k}$ can be assembled by row-wise concatenation of those individual matrices.

The matrix $\boldsymbol{C}$, which is used in the calculated first-order derivatives and which we call variation matrix, is defined as follows

$$\boldsymbol{C} = \frac{\alpha}{2}\cot\left(\frac{\alpha}{2}\right)\boldsymbol{I} - \frac{\alpha}{2}[\boldsymbol{e}]_\times + \left(1 - \frac{\alpha}{2}\cot\left(\frac{\alpha}{2}\right)\right)\boldsymbol{e}\boldsymbol{e}^\top, \tag{5.24}$$

where $\alpha \in \mathbb{R}$ is the rotation angle and $\boldsymbol{e} \in \mathbb{R}^3$ with $\|\boldsymbol{e}\|_2 = 1$ is the rotation axis of the rotation vector ${}_\mathrm{A}\boldsymbol{r}_\mathrm{B} = \alpha\boldsymbol{e}$ evaluated at $\boldsymbol{\theta}_\mathrm{k} = \boldsymbol{0}$. The matrix describes how the rotation vector ${}_\mathrm{A}\boldsymbol{r}_\mathrm{B}$ changes with the variation of a subsequent infinitesimal rotation. It is derived in Appendix E as part of the first-order derivatives. In Appendix F, we show that the variation matrix $\boldsymbol{C}$ is a normal, non-symmetric, and non-orthogonal matrix that is directly related to the rotation matrix by

$${}_\mathrm{A}\boldsymbol{R}_\mathrm{B} = \boldsymbol{C}^\top\boldsymbol{C}^{-1} = \boldsymbol{C}^{-1}\boldsymbol{C}^\top. \tag{5.25}$$

Also, for the rotation vector ${}_\mathrm{A}\boldsymbol{r}_\mathrm{B}$, we are able to prove that

$${}_\mathrm{A}\boldsymbol{r}_\mathrm{B} = \boldsymbol{C}\,{}_\mathrm{A}\boldsymbol{r}_\mathrm{B} = \boldsymbol{C}^\top{}_\mathrm{A}\boldsymbol{r}_\mathrm{B}. \tag{5.26}$$

This shows that the rotation vector ${}_\mathrm{A}\boldsymbol{r}_\mathrm{B}$ is an eigenvector of both the original and transposed variation matrix $\boldsymbol{C}$.

For the approach in Section 5.2.2, which uses body Jacobians to project equations to a minimal parameterization, only exact solutions are possible. At the same time, for the constraint approach in Section 5.2.3 and the equations proposed in this section, kinematic compliance is not apparent. Especially for the rotation, which is highly non-linear in Euclidean space, one could expect that multiple iterations of the Newton optimization are required to converge to a kinematically accurate result. However, in Appendix G, we are able to prove mathematically that this is not the case. In our proof, the developed constraint equations and Jacobians are introduced into the linear equations of the Newton optimization. Also, the previously developed property that ${}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}} = \boldsymbol{C}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}$ is used. Based on this, we calculate the relative pose change per iteration. The results demonstrate that both rotational and translational constraints converge to a kinematically compliant result in a single iteration. While the proof in Appendix G assumes that constraint coordinate frames are equal to model frames, experiments in Section 5.5.4 demonstrate that results also hold for more general cases.

If constraint equations are fulfilled and $\bar{\boldsymbol{b}}_{ab} = \boldsymbol{0}$, the coordinate frames A and B are equal, and the variation matrix $\boldsymbol{C}$ reduces to the identity matrix. As a consequence, the derivatives in Eqs. (5.22) and (5.23) turn into the adjoint representations $-\operatorname{Ad}({}_{\mathrm{A}}\boldsymbol{T}_{\mathrm{M}_a})$ and $\operatorname{Ad}({}_{\mathrm{A}}\boldsymbol{T}_{\mathrm{M}_b})$. A short proof of this is given in Appendix H. The derived expressions directly correspond to the formulation of Drummond and Cipolla (2002). This equivalence demonstrates that our approach is an extension of their method. However, the big advantage of our generalization is that we directly operate on pose differences instead of velocities. This ensures that even if the pose constraint is not fulfilled initially, our method will automatically converge to a consistent result. As a consequence, it is possible to model closed chains or add temporary constraints that are not currently fulfilled. Both cases are not viable using velocity constraints. In addition, we would like to highlight the benefits of directly using the rotation vector in the constraint equations. Compared to other options, such as using orthogonal axes, our formulation has the advantage that it provides consistent results over the entire space of possible rotations. We are, therefore, able to recover even from maximum rotational differences of $\alpha = \pi$ in a single iteration.

5.3.4. Pose Update

Given the derived Jacobians and constraint equations, we are able to estimate the variation vector of the kinematic structure $\boldsymbol{\theta}_{\mathrm{k}}^{\top} = \left[\boldsymbol{\theta}_{\mathrm{j}0}^{\top} \dots \boldsymbol{\theta}_{\mathrm{j}n}^{\top}\right]$ using the linear relation of the Newton optimization in Eq. (5.8). Similar to the calculation of body Jacobians, we formulate a recursive strategy for the pose update. With the estimated joint variation $\hat{\boldsymbol{\theta}}_{\mathrm{j}}$ and the pose of the parent, the pose of the body is updated as follows

$$ {}_{\mathrm{A}}\boldsymbol{T}_{\mathrm{M}}^{+} = {}_{\mathrm{A}}\boldsymbol{T}_{\mathrm{P}}\,{}_{\mathrm{P}}\boldsymbol{T}_{\mathrm{J}}\,\boldsymbol{T}(\hat{\boldsymbol{\theta}}_{\mathrm{j}})\,{}_{\mathrm{J}}\boldsymbol{T}_{\mathrm{M}}, \tag{5.27} $$

where A is some arbitrary frame of reference such as the camera frame. Like in Eq. (5.12), the full 6DoF variation vector $\hat{\boldsymbol{\theta}}_{\mathrm{j}}$ has to be used for the calculation of the transformation matrix. Again, it is simply created by adding zeros to the fixed axes of our joint variation. Note that for the pose of the joint coordinate frame J, either the transformation with respect to the body ${}_{\mathrm{J}}\boldsymbol{T}_{\mathrm{M}}$ or the parent ${}_{\mathrm{P}}\boldsymbol{T}_{\mathrm{J}}$ is fixed. The other is inferred from the previous pose estimates of the body and parent, as well as the fixed transformation. While most of the time, both options are valid, for some joints, which allow, for example, rotation around two axes, it makes a difference if ${}_{\mathrm{J}}\boldsymbol{T}_{\mathrm{M}}$ or ${}_{\mathrm{P}}\boldsymbol{T}_{\mathrm{J}}$ is fixed. Finally, to update the root body, which does not have a parent, the following strategy is adopted

$$ {}_{\mathrm{A}}\boldsymbol{T}^{+}_{\mathrm{M}_0} = {}_{\mathrm{A}}\boldsymbol{T}_{\mathrm{M}_0}\, {}_{\mathrm{M}_0}\boldsymbol{T}_{\mathrm{J}_0}\, \boldsymbol{T}(\hat{\boldsymbol{\theta}}_{\mathrm{j}0})\, {}_{\mathrm{J}_0}\boldsymbol{T}_{\mathrm{M}_0}. \tag{5.28} $$

It considers the variation with respect to the previous pose estimate in the joint's frame of reference. In summary, the developed approach allows us to estimate all body poses in the kinematic structure, ensuring that variation vectors comply with constraints and poses are only updated along free joint directions.

5.4. Implementation

The following section uses the developed framework to extend the multi-modality tracking approach from Chapter 4 to multi-body objects. In addition, for the evaluation, parameter values that differ from definitions in Sections 3.4 and 4.3 are specified. For kinematic structures, it is very common that 3D models of bodies intersect or that they are very close together. Also, different bodies often belong to the same region. To ensure this does not lead to problems, we have to validate contour and surface points for region and depth modalities. For the texture modality, no additional checks are required. The main reason is that, during the computation of keyframes, it is already evaluated if detected points lie on the visible object silhouette.

To validate surface points from depth modalities, a silhouette image is rendered that contains all bodies. Based on this image, surface points of a body are considered valid if they fall on the corresponding silhouette. Note that the same rendering can be employed for multiple modalities that use the same camera. For contour points from region modalities, we also render a silhouette image. However, instead of using a unique identifier (ID) for each body, a region ID is assigned to each body, which is rendered to the respective silhouette. Bodies with the same region ID belong to the same region and consider similar color statistics. As described in Section 3.4.1, it is essential that, for each correspondence line, the body region on the inside and the non-body region on the outside of the contour are not interrupted for a minimum continuous distance along the line, which was defined as 3 segments. Consequently, for each contour point, we check if rendered pixels along the normal vector fulfill this rule. If it is violated, the contour point is rejected. An illustration of the described validation strategies is shown in Fig. 5.5. In addition to contour point validation, the silhouette image is also used to check whether pixel values that should be assigned to color histograms are on the correct silhouette. If they are on the wrong silhouette, they are ignored. Both the rendering and validation are repeated for each correspondence search.

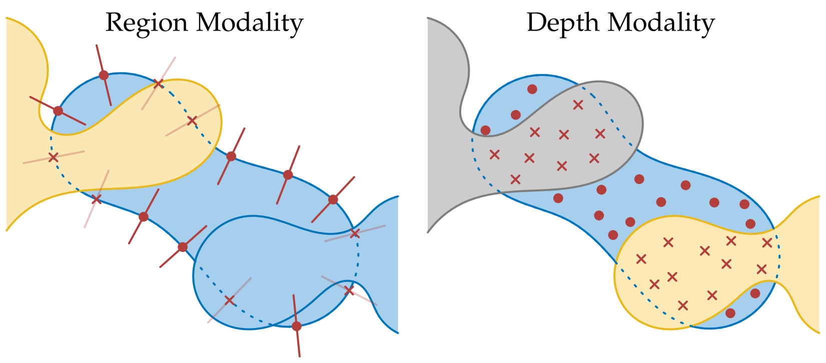


Figure 5.5.: Validation strategies for the region and depth modality. Valid points are visualized with a circle, while invalid points are marked by a cross. For the region modality, the body in the center and lower right corner are rendered with the same region ID. To check if a contour point is valid, pixels on a line along the normal vector are evaluated. If the distance to the inside is interrupted by another silhouette or the distance to the outside is interrupted by the same silhouette, the corresponding contour point is rejected. For the depth modality, all bodies are rendered with unique values. Any surface point that is not on the body's own silhouette is invalid.

In addition to validation strategies, we allow individual region modalities to share color histograms. This is especially beneficial if individual bodies are occluded while other bodies that model the same region are still visible. Also, color histograms consider more information and, therefore, perform slightly better. Another modification required for multi-body tracking considers the number of correspondence lines and points each modality deploys. Since the size of bodies varies widely, the number has to be dynamically adapted to uniformly cover the contour and surface of all bodies. We, therefore, scale the number of lines and points according to the surface area and contour length of a body's current silhouette relative to a reference area and length. Note that surface areas and contour lengths can be precomputed and stored for all views of sparse viewpoint models. In the evaluation in Section 5.5, we use the maximum contour length and surface area of all views and bodies within an object as reference. Also, except for the *Medical Robot*, which is shown in Fig. 5.6, a maximum of 300 correspondence lines and points is deployed per body. For the *Medical Robot*, which consists of equally sized bodies, we only use 100 lines and points to improve computational efficiency.

Like in previous chapters, we use Newton optimization with Tikhonov regularization for the maximization of the joint probability. The linear relation in Eq. (5.8) is thereby extended to include the diagonal regularization matrix $\boldsymbol{\Sigma}$. It is directly added to the Hessian matrix of the full kinematic structure $\boldsymbol{H}_\mathrm{k}$. The diagonal elements of $\boldsymbol{\Sigma}$ consist of the rotational and translational parameters $\lambda_\mathrm{r} = 100$ and $\lambda_\mathrm{t} = 1000$. They are applied to the respective rotational and translational components of the full variation vector $\boldsymbol{\theta}_\mathrm{k}$. Finally, the constraint Jacobian $\boldsymbol{B}_\mathrm{k}$ and the Hessian matrix $\boldsymbol{H}_\mathrm{k}$ have very different orders of magnitude. To ensure stability when solving the linear equation of the optimization, we use a robust Cholesky decomposition with pivoting.

Since, in the evaluation, our tracker deals with large frame-to-frame pose differences, we conduct 6 iterations in which correspondences are established. For the scale parameter of the region modality, we use $s = \{9, 7, 5, 2\}$. Also, the correspondence point threshold of the depth modality, which is given in millimeters, is defined as $r_\mathrm{t} = \{100, 80, 50\}$, and is sampled with a stride of 8 mm. Note that values for the thresholds r_t and the stride are scaled according to the depth of model points and are defined at a reference distance of 1 m. In addition to those parameters, we use the standard deviations $\sigma_\mathrm{r} = \{25, 15, 10\}$ and $\sigma_\mathrm{d} = \{50, 30, 20\}$, which are given in pixels and millimeters, respectively. The provided sets define parameters for each iteration, with the last value employed in all remaining iterations. Finally, in the evaluation, we only use geometry-based tracking with a single region and depth modality per body. The main reason is that objects in the dataset are mostly untextured, and most bodies are modeled by a single homogeneous material and color. Finally, with the developed framework and the implementation details described in this section, we are able to use *M3T* for a wide variety of open and closed kinematic structures.

5.5. Evaluation

The following section provides a detailed evaluation of the developed framework and its implementation in the *M3T* tracker. For this, we introduce the *Robot Tracking Benchmark (RTB)*, a highly-realistic synthetic dataset that considers multiple robots and a large number of sequences. Subsequently, the used evaluation metrics are defined. Based on the dataset, different configurations of kinematic structures are assessed with respect to quality and efficiency. After this, the convergence of constraints is analyzed. Finally, a comparison to another state-of-the-art algorithm is provided, and remaining limitations are discussed. All experiments were again conducted on a computer with an *Intel Core i9-11900K* CPU and a *NVIDIA RTX A5000* GPU. A video that shows the performance of our tracker for real-world sequences and that illustrates the *RTB* dataset is publicly available.[1]

5.5.1. Robot Tracking Benchmark

In the past, evaluations of multi-body tracking and pose estimation methods considered only a very limited number of sequences. Consequently, experiments were typically of a more qualitative nature (Krainin et al. 2011; Klingensmith et al. 2013; Bohg et al. 2014; Michel et al. 2015; Pauwels and Kragic 2015; Schmidt et al. 2015a). The main reason is that diverse real-world data with moving cameras and high-quality pose annotations is hard to obtain. However, to ensure reliable evaluation results, a sufficient number of realistic sequences with accurate ground truth is essential.

[1]`https://www.youtube.com/watch?v=0ORZvDDbDjA`

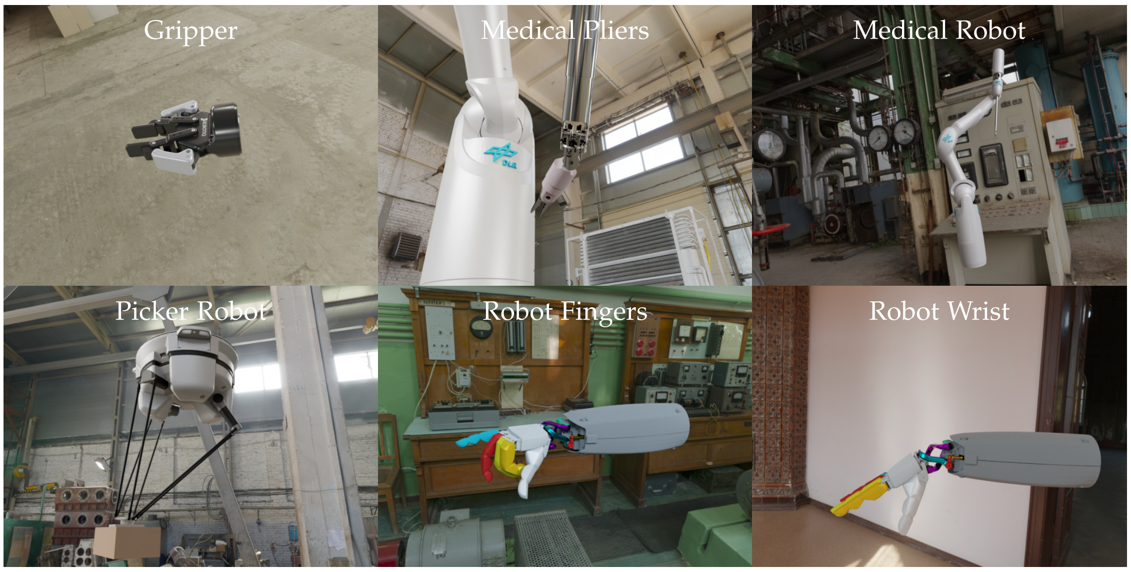


Figure 5.6.: Multi-body objects included in the *RTB* dataset, featuring the *Gripper*, *Medical Pliers*, *Medical Robot*, *Picker Robot*, *Robot Fingers*, and *Robot Wrist*. The *Gripper*, *Picker Robot*, and *Robot Wrist* contain closed kinematic chains. Note that for the *Robot Fingers*, the wrist is locked, and only finger joints are moving, while the *Robot Wrist* features locked fingers and a movable wrist.

We, therefore, introduce the *Robot Tracking Benchmark (RTB)*. It is a novel synthetic dataset that was developed using the procedural rendering pipeline *BlenderProc*[2] and which is publicly available[3]. Example images of the dataset are shown in Fig. 5.6. To facilitate the creation of the dataset, *BlenderProc* was extended to load robot models from *URDF* files. Also, for the generation of articulated motions, forward and backward kinematics were integrated. The open-source pipeline produces photo-realistic sequences with *HDRi* lighting[4] and physically-based materials. Perfect ground-truth annotations for camera and robot trajectories are provided in the *BOP* format (Hodaň et al. 2020). Many physical effects, such as motion blur, rolling shutter, and camera shaking, are accurately modeled to reflect real-world conditions. For each frame, the following four depth qualities are available: *Ground Truth*, *Azure Kinect*, *Active Stereo*, and *Stereo*. They simulate sensors with different characteristics. While the first quality provides perfect ground truth, the second considers measurements with the distance-dependent noise characteristics of the *Azure Kinect* time-of-flight sensor (Tölgyessy et al. 2021). Smoothed depth is thereby modeled using random Gaussian shifting, and, at very dark surfaces, missing measurements are considered. Finally, for the *Active Stereo* and *Stereo* qualities, two stereo RGB images with and without a pattern from a simulated dot projector were rendered. Depth images were then reconstructed using the *SGM* algorithm of

[2]`https://github.com/DLR-RM/BlenderProc`
[3]`https://zenodo.org/record/7548537`
[4]`https://polyhaven.com/hdris`

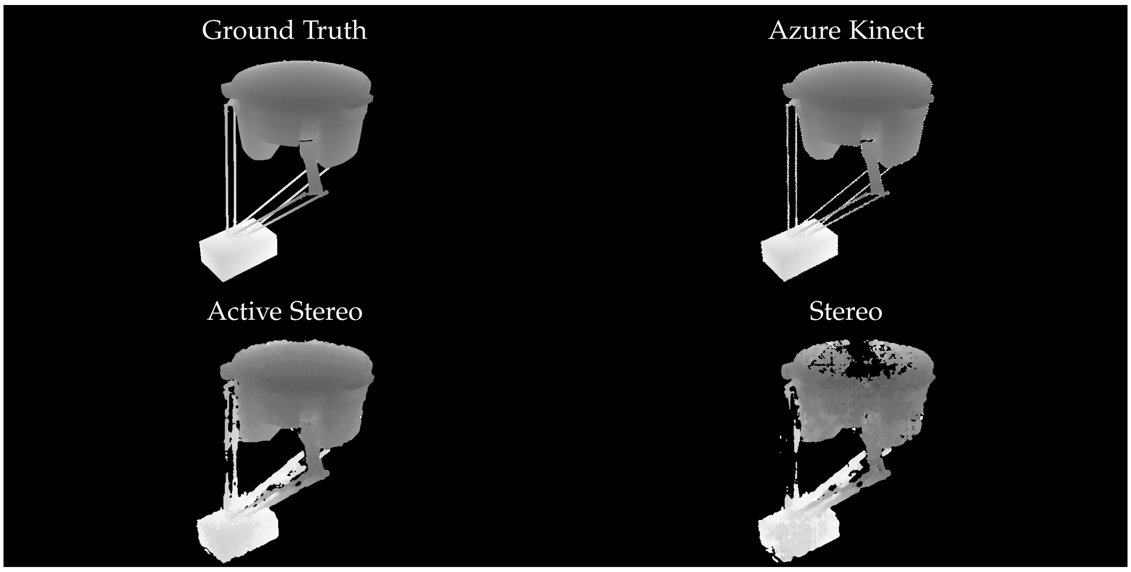


Figure 5.7.: Depth image qualities provided in the *RTB* dataset, featuring *Ground Truth* with perfect measurements, noise characteristics from an *Azure Kinect* camera, and stereo reconstructions from *SGM*. For *Active Stereo*, the pattern from a dot projector is simulated to add additional texture information, while *Stereo* operates without a pattern.

Table 5.1.: Characteristic numbers and thresholds for objects included in the *RTB* dataset.

	Gripper	Medical Pliers	Medical Robot	Picker Robot	Robot Fingers	Robot Wrist
Number of Bodies	9	5	8	11	16	12
Degrees of Freedom	2	4	7	3	15	2
Closed Chains	4	0	0	6	0	4
Error Threshold e_t [m]	0.01	0.01	0.1	0.1	0.05	0.05

Hirschmüller (2005). Examples of the four qualities are shown in Fig. 5.7.

The benchmark features 6 robotic systems with different kinematics, ranging from simple chain and tree topologies to structures with complex closed kinematics. Example images of all multi-body objects included in the dataset are shown in Fig. 5.6. An overview of their kinematic structure is given in Table 5.1. For each robotic system, we provide three difficulty levels: *Easy*, *Medium*, and *Hard*. In all sequences, the kinematic system is in motion. However, while for *Easy*, the camera is mostly static with respect to the robot, *Medium* and *Hard* feature faster and shakier motions for both the robot and camera. Consequently, motion blur increases, which also reduces the quality of stereo matching. Finally, for each object, difficulty level, and depth image quality, 10 sequences with 150 frames are available. In total, this results in 108.000 frames that feature different kinematic structures, motion patterns, depth measurements, scenes, and lighting conditions. In summary, given the diverse data, our *Robot Tracking Benchmark* allows to extensively measure, compare, and ablate the performance of multi-body tracking algorithms, which is essential for further progress in the field.

5.5.2. Metrics

For the evaluation on the *RTB* dataset, we again adopt ADD and ADD-S area-under-curve scores (Hinterstoisser et al. 2013) and modify them for multi-body structures. The average-distance (ADD) error e_{ADD} and average-shortest-distance (ADD-S) error $e_{\text{ADD-S}}$, which were also used in Section 4.4.1, are thereby computed for each body and frame as follows

$$e_{\text{ADD}} = \frac{1}{n_{\text{v}}} \sum_{i=1}^{n_{\text{v}}} \left\| \left({}_{\text{M}}\widetilde{\boldsymbol{X}}_i - {}_{\text{M}}\boldsymbol{T}_{\text{M}_{\text{GT}}} \, {}_{\text{M}}\widetilde{\boldsymbol{X}}_i \right)_{3\times 1} \right\|_2, \tag{5.29}$$

$$e_{\text{ADD-S}} = \frac{1}{n_{\text{v}}} \sum_{i=1}^{n_{\text{v}}} \min_{j\in[n_{\text{v}}]} \left\| \left({}_{\text{M}}\widetilde{\boldsymbol{X}}_i - {}_{\text{M}}\boldsymbol{T}_{\text{M}_{\text{GT}}} \, {}_{\text{M}}\widetilde{\boldsymbol{X}}_j \right)_{3\times 1} \right\|_2, \tag{5.30}$$

where ${}_{\text{M}}\boldsymbol{T}_{\text{M}_{\text{GT}}}$ is the transformation between the ground-truth (GT) and estimated model pose, $\widetilde{\boldsymbol{X}}_i$ is a vertex from the 3D mesh of the body written in homogeneous coordinates, n_{v} is the number of vertices, and $()_{3\times 1}$ denotes the first three elements of a vector. Note that, in the case that a rigid body is composed of multiple 3D meshes, one simply averages the conventional and shortest distance error over all meshes. Based on the errors of individual bodies and frames e_{ij}, we compute the area-under-curve score for an entire kinematic structure and sequence as follows

$$s = \frac{1}{n_{\text{b}} n_{\text{f}}} \sum_{i=1}^{n_{\text{b}}} \sum_{j=1}^{n_{\text{f}}} \max\left(1 - \frac{e_{ij}}{e_{\text{t}}}, 0\right), \tag{5.31}$$

with n_{b} the number of bodies, n_{f} the number of frames, and e_{t} an error threshold. For the computation of ADD and ADD-S area-under-curve scores, we utilize the error metrics e_{ADD} and $e_{\text{ADD-S}}$ for the error e_{ij}, respectively. Error thresholds e_{t} that are used for the considered multi-body objects are given in Table 5.1. They are defined to be approximately ten percent of each object's diameter. The resulting score considers each body equally important, takes into account the size of the kinematic structure, and rewards accuracy while limiting the impact of bad predictions.

5.5.3. Kinematic Configuration

Using the *RTB* dataset, we thoroughly evaluate the proposed multi-body tracking framework. For this, we compare four kinematic configurations per object. We thereby differentiate between (i) *independently* tracked bodies, (ii) *projection* to a minimal parameterization with Jacobians, (iii) *constraints* using Lagrange multipliers, and (iv) a

Table 5.2.: Comparison of different kinematic configurations on the *RTB* dataset, showing ADD-S area-under-curve scores in percent and average runtimes in milliseconds. Objects with tree-like kinematic structures are indicated by a $\star$.

Configurations	Gripper	Medical Pliers*	Medical Robot*	Picker Robot	Robot Fingers*	Robot Wrist	**Average**	**Runtime [ms]**
Independent	21.6	34.6	52.1	21.0	28.5	28.3	31.0	13.2
Projected	71.1	81.7	95.1	54.0	90.0	86.1	79.7	13.5
Constrained	87.4	79.9	92.9	93.7	84.3	96.8	89.2	16.2
Combined	87.9	81.7	95.1	94.7	90.0	97.4	91.1	13.8

combination of projection and constraints. In the *combined* case, the main configuration is equal to the *projected* scenario. However, all chains are closed using additional constraints from the *constrained* version.

Results of the evaluation are shown in Table 5.2. The experiments clearly demonstrate that *independent* tracking of bodies is not an option for kinematic structures. Also, while the *projection* to a minimal parameterization works very well for tree-like structures, for objects with closed kinematic chains, such as the *Gripper*, *Picker Robot*, and *Robot Wrist*, it is a significant disadvantage to not use the full kinematic information. Finally, what is most interesting is the comparison of the *constrained* and *combined* configurations. Even though both include the full kinematic information, the *combined* version performs better than the *constrained*. The main reason for this is a difference in regularization. For the *constrained* configuration, regularization considers the full pose variation of each body independently. On the other hand, for the *combined* case, relative joint variations between bodies are regularized. Since bodies in a kinematic structure move relative to each other and the entire structure moves with respect to the camera, the relative regularization of the *combined* configuration is closer to reality. Consequently, using projection and combining it with constraints to model closed chains works best.

In addition to tracking quality, we also compare computational efficiency. Results in Table 5.2 show that runtimes for the *independent*, *projected*, and *combined* configurations are almost equal. For the *constrained* configuration, it is, however, noticeably larger. To analyze this difference in more detail, we create an additional experiment that compares the *projected* and *constrained* configurations. We thereby consider a kinematic chain with one rotational link per body and a variable number of bodies. For the comparison, we measure the runtime required for a single iteration of the optimization and the pose update. Results over the number of bodies are given in Fig. 5.8. The plots show a significant difference between the two configurations. The main reason is that, while for the *constrained* version, 11 unknowns are added per additional body, with 6 parameters for the body pose and 5 for the constraint, only 1 unknown is required in the *projected*

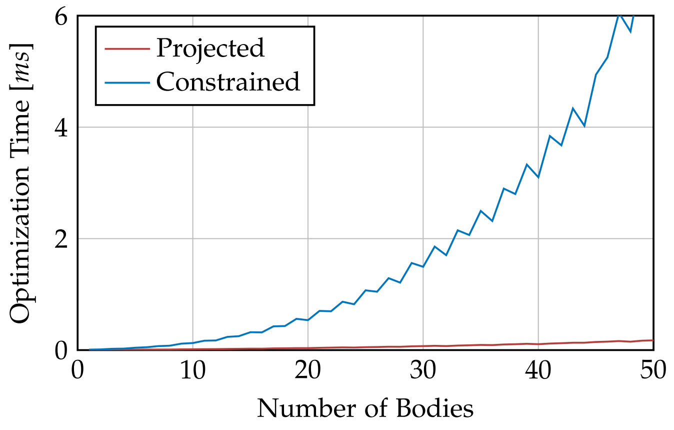


Figure 5.8.: Optimization time per iteration for the *projected* and *constrained* configuration over the number of bodies in a kinematic chain.

scenario. Together with the Cholesky decomposition's computational complexity of $\mathcal{O}(n^3)$, which is used to solve the linear system of equations, this explains the obtained results. Also, while the difference between the two formulations decreases for joints with more than one degree of freedom, the *constrained* configuration is never more efficient than the *projected*. In conclusion, experiments in this section demonstrate that using projection where possible and combining it with constraints where necessary works best both for quality and efficiency.

5.5.4. Constraint Convergence

In Appendix G, we proved mathematically that constraints converge in a single iteration for the special case of equal model and constraint coordinate frames. In the following, we analyze the more general case of unequal coordinate frames. For this, random transformations ${}_{\mathrm{A}}\boldsymbol{T}_{\mathrm{M}_a}$ and ${}_{\mathrm{B}}\boldsymbol{T}_{\mathrm{M}_b}$ are defined. Also, we start with an initial random difference ${}_{\mathrm{A}}\boldsymbol{T}_{\mathrm{B}}$ between constraint reference frames A and B. Transformations are thereby generated using normalized 3D rotation and translation vectors with a random orientation from a uniform distribution. The length of those rotation and translation vectors is then sampled from uniform distributions on the intervals $[-1, 1]$ and $[-\pi, \pi]$, with translations in meters and rotations in radians. Given an initial pose difference ${}_{\mathrm{A}}\boldsymbol{T}_{\mathrm{B}}$, four Newton iterations are conducted. For each iteration, absolute rotation and translation errors $||{}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}||_2$ and $||{}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{B}}||_2$ are computed. In total, we evaluate 100.000 random cases and report error percentiles. Obtained results are visualized in Fig. 5.9. The plots demonstrate that, like in our proof, both rotational and translational constraints converge in a single iteration. Also, while we do not report them here, similar results are obtained if individual components of the rotation and translation are constrained.

To provide additional context, we compare our results to convergence plots that use orthogonality constraints for the rotation. One thereby enforces orthogonality between

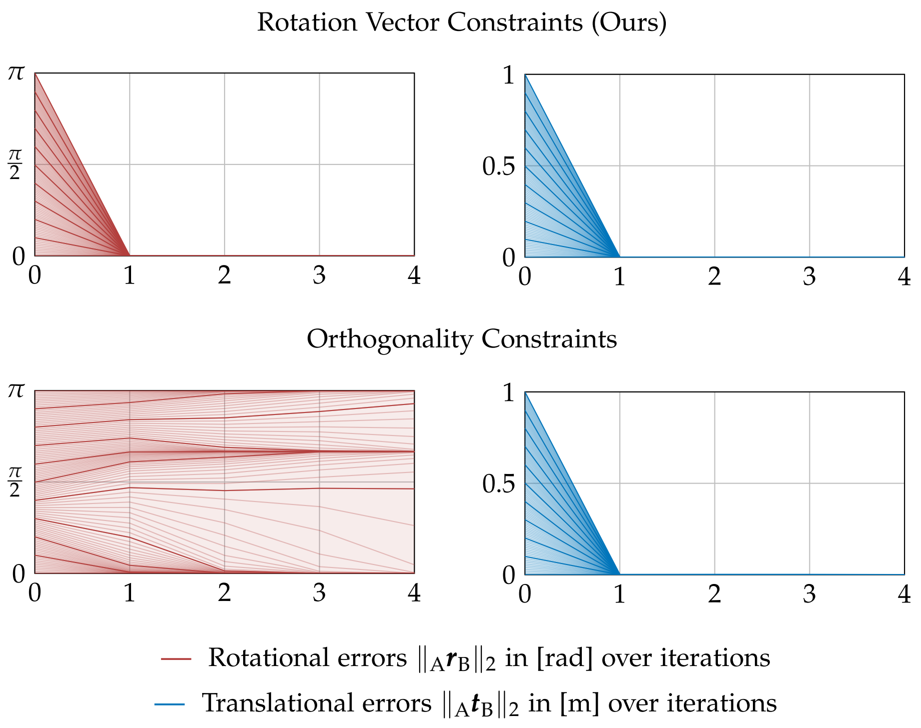


Figure 5.9.: Convergence plots showing rotational and translational errors over iterations of the Newton method. Initial errors are uniformly distributed with a maximum of $\|_A\boldsymbol{r}_B\|_2 = \pi$ and $\|_A\boldsymbol{t}_B\|_2 = 1\,\text{m}$. For the translation, the constraint ${}_A\boldsymbol{t}_B = \boldsymbol{0}$ is used. For the rotation, we differentiate between our constraints, which restrict the rotation vector using ${}_A\boldsymbol{r}_B = \boldsymbol{0}$, and orthogonality constraints that ensure that axes of reference frame A are orthogonal to axes of reference frame B. Dark lines visualize error deciles while bright lines show error percentiles.

the axes of the constraint coordinate frames A and B as follows

$$b_{\text{r}ijab}(\boldsymbol{\theta}_a, \boldsymbol{\theta}_b) = {}_A\boldsymbol{e}_i^\top \, {}_A\boldsymbol{R}_B(\boldsymbol{\theta}_a, \boldsymbol{\theta}_b) \, {}_B\boldsymbol{e}_j, \tag{5.32}$$

with the tuple $(i, j) \in \{(\text{x}, \text{y}), (\text{y}, \text{z}), (\text{z}, \text{x})\}$. Derivatives that are required for the implementation are given in Appendix I. Convergence plots of the experiment are again shown in Fig. 5.9. The results demonstrate that multiple iterations are required to converge to a kinematically compliant result. However, what is even worse is that the orthogonality requirement is not unique over the space of possible rotations. In total, 7 additional pose configurations exist that fulfill the defined orthogonality constraints. In Fig. 5.9, those cases are visible in the convergence towards a rotational error of π and $\frac{2}{3}\pi$. In conclusion, our experiments demonstrate that, in contrast to other approaches, the developed constraints converge to a kinematically exact solution over the entire space of possible rotations while requiring only a single iteration.

Table 5.3.: Comparison of tracking algorithms on the *RTB* dataset, showing ADD and ADD-S area-under-curve scores in percent and average runtimes in milliseconds.[6] Objects with tree-like kinematic structures are indicated by a $\star$.

Object		Gripper		Medical Pliers*		Medical Robot*		Picker Robot		Robot Fingers*		Robot Wrist		**Average**	
App.	Level	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S
DART	Easy	33.9	57.2	29.7	44.9	75.1	86.3	27.9	33.8	54.3	64.8	51.6	63.3	45.4	58.4
	Medium	1.4	3.1	11.0	25.5	12.1	20.7	3.2	5.4	2.9	4.7	4.9	8.3	5.9	11.3
	Hard	1.4	3.1	8.1	20.1	3.6	7.8	1.0	2.2	1.2	2.1	1.4	2.6	2.8	6.3
	All	12.2	21.2	16.2	30.1	30.2	38.3	10.7	13.8	19.5	23.9	19.3	24.7	18.0	25.3
	Runtime	4.1 ms		3.7 ms		5.8 ms		9.6 ms		12.8 ms		11.4 ms		7.9 ms	
M3T (Ours)	Easy	92.1	94.1	78.3	86.8	93.1	97.0	96.0	97.3	95.6	96.8	97.9	98.5	92.2	95.1
	Medium	81.5	86.7	64.6	79.0	86.5	94.2	90.8	93.6	86.9	91.0	96.6	97.8	84.4	90.4
	Hard	78.6	83.0	62.9	79.2	81.8	94.1	90.7	93.0	76.5	82.2	94.0	95.9	80.7	87.9
	All	84.1	87.9	68.6	81.7	87.1	95.1	92.5	94.7	86.3	90.0	96.2	97.4	85.8	91.1
	Runtime	9.7 ms		6.8 ms		20.5 ms		11.4 ms		16.7 ms		17.5 ms		13.8 ms	

5.5.5. Comparison

In addition to the framework's evaluation, we compare our *M3T* tracker to another popular method. For this, we decided to use *DART* (Schmidt et al. 2015b; Schmidt et al. 2015a). It is one of the few available general-purpose algorithms for the tracking of kinematic structures. *DART* uses signed distance functions to consider depth information and allows the configuration of kinematic structures with both rotational and prismatic joints. For our evaluation, we use source code that is publicly available[5] and leave all parameters at their default values. Except for the *Picker Robot*, all experiments are conducted on the same computer.[6] Results of the evaluation are shown in Table 5.3. The comparison emphasizes the performance of our approach. While for *Easy* sequences, *DART* is still able to track some objects well, for *Medium* and *Hard*, most sequences are impossible. The main reason is that, due to the large pose differences between frames, the algorithm quickly loses most objects. This suggests that using depth alone and not being able to model closed kinematic structures significantly limits tracking performance. In comparison, *M3T* is able to successfully track almost all objects and achieves relatively high scores even for sequences in the category *Hard*.

With respect to runtime, both *DART* and *M3T* are able to operate well above 30 Hz, which is the frequency of most consumer RGB-D cameras. For *DART*, the largest runtime is observed for the *Robot Fingers*, which has the maximum number of bodies. For our

[5] `https://github.com/tschmidt23/dart`

[6] Due to memory requirements of the *Picker Robot*, the object's evaluation with *DART* had to be conducted on a computer with an *Intel Xeon Gold 6254* CPU and a *NVIDIA RTX A6000* GPU.

Table 5.4.: Tracking results on the *RTB* dataset for different depth image qualities with ADD and ADD-S area-under-curve scores in percent.

Approach	DART		M3T (Ours)	
Depth Quality	ADD	ADD-S	ADD	ADD-S
Ground Truth	21.1	29.0	88.7	93.0
Azure Kinect	16.2	22.4	87.9	92.5
Active Stereo	18.0	25.3	86.2	91.5
Stereo	16.9	24.5	80.4	87.6

approach, most resources are required for the tracking of the *Medical Robot*, which uses a large number of correspondence lines and points. With its highly-optimized *CUDA* implementation and the used *NVIDIA RTX A5000* GPU, *DART* is even faster than our approach. At the same time, for our method, the GPU is only required to validate contour and surface points. Consequently, it remains mostly idle. Also, the algorithm runs on a single CPU core. In summary, while differences with respect to runtime and required computational resources exist, both *DART* and *M3T* are highly efficient and provide real-time performance.

Finally, in addition to difficulty levels, we evaluate both approaches with respect to different depth image qualities provided in the *RTB* dataset. Results of the experiments are given in Table 5.4. For *DART*, the evaluation shows a notable difference between *Ground Truth* and other depth image qualities that are more realistic and imperfect. In comparison, for our approach, differences between *Ground Truth*, *Azure Kinect*, and *Active Stereo* are less severe, and results are mostly comparable. Only for *Stereo*, for which depth images often have missing measurements and provide considerably less information, the tracking performance is lower. In summary, the results demonstrate that *M3T* does not require perfect depth images. Instead, with a combination of region and depth information, one is able to achieve high-quality tracking even in challenging scenarios with imperfect depth measurements.

5.5.6. Limitations

While our framework allows to model various kinematic structures and is compatible with a wide variety of algorithms, some limitations remain. First, it focuses on approaches that employ Newton-like optimization techniques such as Gauss-Newton, Levenberg-Marquardt, or quasi-Newton methods. The framework is, therefore, not compatible with particle-based methods or existing deep learning-based techniques. Also, in this work, we only derived equations based on the axis-angle representation. Note, however, that it would be relatively easy to derive the equations required for other parameterizations. Another limitation comes from the Newton method itself. Because each iteration considers the energy function only at a single point, the algorithm can get stuck in local minima. Like for most tracking methods, this limits maximum configuration changes and pose differences between frames. Also, algorithms that should be extended with our framework have to deal with multiple bodies at the same time. Typical challenges thereby include bodies that are very close together or even intersect. As in our extension of *M3T*, it might be necessary to modify the original algorithm to successfully deal with such cases.

5.6. Conclusion

In this chapter, we developed a framework that allows the extension of existing 6DoF algorithms to multi-body object tracking. It combines projection to a minimal parameterization and the application of pose constraints into a single formulation. With the efficiency and realistic regularization of projection and the ability to model temporary connections and additional constraints, it combines the best of both worlds. The resulting formulation allows the modeling of a wide variety of kinematic structures and, to the best of our knowledge, is the first that accurately considers closed kinematic structures. In a detailed mathematical proof, as well as in experiments, we were able to show that the developed constraints enforce an exact kinematic solution, converging in a single iteration of the Newton optimization. Also, in contrast to previous work, our constraints directly operate on pose differences instead of velocities. This ensures that kinematic errors are minimized and cannot accumulate over time.

Based on the developed framework, we extended the multi-modality tracking approach from Chapter 4 to multi-body tracking. For a thorough, quantitative evaluation, we introduced the *RTB* dataset. It features highly-realistic sequences and multiple robots, both with open and closed kinematic chains. The dataset also provides numerous sequences in various settings with distinct difficulty levels and depth qualities. In a detailed comparison, we demonstrated that our *M3T* algorithm significantly outperforms *DART* (Schmidt et al. 2015b), which is a state-of-the-art articulated object tracker. With the obtained results and new possibilities for quantitative evaluation, we are confident that our tracker and dataset have many applications in robotics and computer vision. Also, given the versatility of our framework and its compatibility with a wide variety of existing and potential future methods, we hope that more approaches will move from rigid objects to kinematic structures.

6 Applications

6.1. Introduction

In real-world applications, a wide variety of robots and objects with various kinematic structures and surface characteristics exist. Also, while in some cases, it is only possible to obtain monocular RGB images, other scenarios feature extensive multi-camera setups that provide color and depth information. The previously presented mathematics, techniques, and strategies have been developed with that diversity in mind. Consequently, they support a wide variety of scenarios. While this constitutes a good foundation, it is essential that the flexibility of those theoretical capabilities is mirrored in the algorithm's implementation. In the following chapter, we present our multi-body, multi-modality, and multi-camera tracking library *M3T*. It features a highly-modular architecture that allows the configuration of different components, such as cameras, links, constraints, modalities, viewers, detectors, refiners, publishers, and subscribers. Given this flexibility, *M3T* supports all the theoretical capabilities of our approach.

While for datasets, tracking is typically started using the ground-truth pose of the first frame, for real-world applications, such information is not available. For the initialization, we therefore provide different solutions in the *M3T* library, including the integration of a deep learning-based global detection and pose estimation algorithm. Similar to experiments in Section 4.4.6, predictions can thereby be improved using our algorithm for pose refinement. Finally, most robots are not only equipped with cameras but also provide proprioceptive measurements from joint sensors. Given that this information constrains the relative location of individual links, it is highly valuable for the tracking of robots. We, therefore, discuss how to integrate such measurements into the presented algorithm. Depending on the quality of measurements, our approach can be used to improve both the robustness and accuracy of tracking results.

Finally, for the developed *M3T* library, we provide two example applications. In the first, we consider the humanoid robot *David*. The robot features an *Azure Kinect* RGB-D camera that is mounted inside the robot's head on top of an elastic neck. For the neck, only very rough forward kinematics are available. Also, the robot's arm and body are designed for efficiency and are not perfectly rigid. It is, therefore, impossible to compute the exact location of the hand relative to the camera. Instead, to interact with objects, one has to predict poses of both the hand and manipulated objects. Based on this visual feedback, one can then use relative positioning. Once the object has contact with the hand, constraints from fingers provide further information. In addition to hand tracking, we, therefore, also discuss how visual information can be combined with an existing grasp state estimation approach that considers contact information.

Finally, in the second application, we discuss the integration of the *M3T* tracker with the *MiroSurge* system for minimal-invasive robotic surgery. The setup consists of multiple *MIRO* robots that are equipped with versatile instruments for minimal-invasive surgery called *MICA*. In general, the *MICA* instrument provides two tendon-driven joints and pliers. They are connected to motors and sensors inside a drive unit at the end of a long shaft. Usually, the robot and instrument are operated by a human who uses images from an endoscope. To automate operations or place virtual fixtures that constrain the instrument, the end effector's location relative to the camera has to be known. However, given the elasticity of the mechanical structure and tendons together with the long kinematic chain, kinematic information is insufficient to predict the end effector's pose and configuration. To overcome this limitation, the *M3T* tracker is used. Similar to a human operator, the tracker uses RGB images from the endoscope to estimate the pose and configuration of the *MICA* instrument, as well as manipulated objects.

6.2. The M3T Library

In this section, we present the *M3T* library. It is implemented in C++ and is available as open-source software[1]. First, the library's architecture is discussed. We thereby provide an overview of different components and explain how to combine them. Subsequently, the tracking process, which is used to estimate the poses of all bodies, is described. This is followed by a short explanation of different detection methods that can be used for initialization. Finally, an approach that allows the incorporation of joint measurements of varying accuracy and reliability is presented.

6.2.1. Architecture

For the library's architecture, both theoretical capabilities of our approach and practical requirements from applications have to be considered. For example, the 3D object tracker presented in this thesis allows the incorporation of information from different modalities and supports the use of multiple cameras for which the relative pose is known. Geometric information for the region and depth modality are thereby stored in sparse viewpoint models that are precomputed. Also, we presented an occlusion handling strategy that uses information from depth renderings or depth camera images. To facilitate the tracking of bodies that are close to each other, contour and surface points are validated based on silhouette renderings that encode different values for individual bodies and regions. In addition, region modalities are able to share color histograms that model the foreground and background. Moreover, for multi-body objects, individual links are connected by joints to form tree-like kinematic structures.

[1] `https://github.com/DLR-RM/3DObjectTracking`

Those structures can be restricted using constraints between links. Finally, in addition to those capabilities, for real-world applications, it is important to exchange data with other processes. Also, to initialize poses and reset the tracker in cases of tracking loss, the integration of global detection and pose estimation methods is required. Furthermore, in many cases, visualizations of current predictions are needed. Based on those capabilities and requirements, we identified the following components, which provide specific functionality and data:

Camera Specifies methods to fetch images and provides those images to other components. In addition, it stores intrinsics and defines the camera's pose relative to the world coordinate frame. In general, one differentiates between *Color Cameras* and *Depth Cameras*. Depending on the used physical camera or the integration with existing image pipelines, different implementations have to be created.

Body Holds the pose of a rigid body relative to the world coordinate frame and stores the body's mesh geometry. It also specifies values for region and body IDs that are used in the rendering of silhouette images.

Renderer Geometry Stores the geometry of referenced *Bodies* on the GPU and provides everything required for rendering.

Renderer Creates a rendering based on the geometry in a referenced *Renderer Geometry* object, the poses of corresponding *Bodies*, and the view of the renderer on the scene defined by intrinsics and the renderer's pose relative to the world coordinate frame. For tracking, intrinsics and the pose are initialized from the values of a *Camera* object. Currently, three implementations exist: a *Depth Renderer* that outputs a depth image, a *Silhouette Renderer* that creates a silhouette image, which features either region or body IDs, and a *Normal Renderer* that encodes surface normal vectors. For improved efficiency, versions are provided that focus only on referenced *Bodies*, instead of rendering the entire image.

Model Precomputes a sparse viewpoint model from a referenced *Body* and stores the created data. During the generation, geometries from other *Bodies* can be considered. This can, for example, be used to consider occlusions before tracking. Two versions exist: a *Region Model* that samples contour points and a *Depth Model* that computes surface points. They are used for region and depth modalities, respectively. Modalities that consider the same geometry can use the same model.

Color Histograms This object computes and stores color histograms for the foreground and background. Typically, the object is incorporated in the *Region Modality*. However, if histograms should be shared, *Color Histograms* can also be defined separately and referenced by multiple *Region Modalities*.

Modality Considers information from a *Camera* and the pose of a *Body* to compute correspondences and calculate the gradient vector and Hessian matrix used for

Newton optimization. As previously discussed, three modalities exist: a *Region Modality* that sparsely considers region information and requires a *Color Camera*, *Body*, and *Region Model*, a *Depth Modality* that implements an *ICP*-like depth approach that considers data from a *Depth Camera*, *Body*, and *Depth Model*, and a *Texture Modality* that uses keypoint features and requires information from a *Color Camera*, *Body*, and *Silhouette Renderer*. All modalities allow referencing a *Depth Renderer* to model occlusions. Also, while *Depth Modalities* use information from the already required *Depth Camera*, both *Region* and *Texture Modalities* allow referencing an additional *Depth Camera* to consider occlusions. The physical depth camera, thereby, has to be close to the color camera. Finally, for the validation of contour points and surface points, *Region* and *Depth Modalities* use a *Silhouette Renderer*. Also, *Region Modalities* are able to reference external *Color Histograms*.

Link Defines the location of a joint reference frame relative to the reference frame of the link and its parent. For the joint, rotational and translational motion along defined coordinate axes is allowed. To build a tree-like kinematic structure, a *Link* is able to reference multiple *Links* as children. In general, *Links* contain all *Modalities* that correspond to a single referenced *Body*. If a *Body* is configured, the reference frame of the *Link* is equal to that of the *Body*. Also, it is possible to create virtual *Links* without referenced *Body* or *Modality* objects to allow the definition of kinematics that do not feature visual information.

Constraint Takes two *Links* and defines the location of constraint reference frames relative to corresponding link frames. Rotational and translational motion between the two constraint frames is restricted for user-defined coordinate axes. Currently, two constraint implementations exist: the mathematically exact *Constraint* that was discussed previously and a *Soft Constraint* that can be used to consider inaccurate proprioceptive measurements. The latter will be introduced in Section 6.2.4.

Optimizer References a single *Link* at the root of a kinematic structure and multiple corresponding *Constraints* and *Soft Constraints*. It computes Jacobian matrices for all *Links* and uses them to project gradient vectors and Hessian matrices from *Modalities* and *Soft Constraints* to the full gradient and Hessian required for Newton optimization. In addition, an *Optimizer* projects Jacobians and residuals from *Constraints* into the full system of linear equations. After a single Newton step, it updates the pose of all *Links* and *Bodies* in the kinematic structure.

Detector Depending on the implementation, a *Detector* references multiple *Optimizers*. Based on the detected pose of the root *Link*, it updates the pose of all corresponding *Link* and *Body* objects in the kinematic structure. Three different detectors are implemented: a *Static Detector* that assigns a predefined pose, a *Manual Detector* that allows a user to define four points in a color image to infer the pose, and an *Automatic Detector* that uses a deep learning-based detection and pose estimation pipeline to infer the pose of multiple objects.

Viewer Is used to visualize results. Currently, a *Normal Viewer* and an *Image Viewer* are implemented. While the first references a *Camera* and *Renderer Geometry* object to overlay normal renderings on the camera image, the second simply shows an image from a referenced *Camera*. For both viewers, variants are available that consider color or depth images.

Publisher Is used to communicate data to an external source. It features an abstract method that is called at the end of the pose optimization. Its implementation is highly dependent on the exchanged information and the receiving source. In most applications, it is used to publish predicted poses.

Subscriber Is used to read data from an external source. It provides an abstract method that is called before the optimization. As for *Publishers*, the implementation is highly dependent on the source and exchanged information. When the main method is executed, the tracker is in a defined state, without anything happening in parallel. A subscriber can, therefore, not only modify pose predictions but is able to change settings or the entire configuration of referenced objects.

Refiner References *Optimizers* to refine pose predictions of corresponding kinematic structures. It coordinates different methods provided by *Optimizers*, *Constraints*, *Soft Constraints*, *Links*, *Modalities*, *Color Histograms*, and *Renderers*. The refinement process is very similar to the *Tracking Step* of the *Tracker*, which will be described in Algorithm 6.4. The only main difference is that methods featured in the *Starting Step* in Algorithm 6.3 are executed each time before correspondences are calculated.

Tracker References *Optimizers*, *Refiners*, *Publishers*, *Subscribers*, *Viewers*, and *Detectors*. In addition to an orderly setup, it facilitates the entire tracking process. The *Tracker* thereby coordinates the execution of methods provided by different objects. To control the process, it provides interfaces to start the tracking, stop the tracking, and execute the detection of kinematic structures represented by an *Optimizer*. A detailed description of the tracking process is provided in Section 6.2.2.

With the presented components and their flexible combination, a wide variety of different applications can be supported. In our library, components are implemented as classes in C++. To connect individual objects, one simply passes shared pointers. In addition to programming, configurations can also be defined using files in the *YAML* file format. Based on those files, the entire tracker is created automatically.

An example that demonstrates how components from the *M3T* library can be configured for a simple application is shown in Fig. 6.1. It features a setup that considers information from two color cameras and one depth camera to track a kinematic structure that consists of two bodies. In the visualization, referenced objects are connected by lines in the color of the referenced object. Note that while components have no access upwards to objects from where they are referenced, connecting downwards through other objects is not a problem. Consequently, the *Tracker* has access to all components in the configuration. This capability is used to ensure that all objects are correctly set up

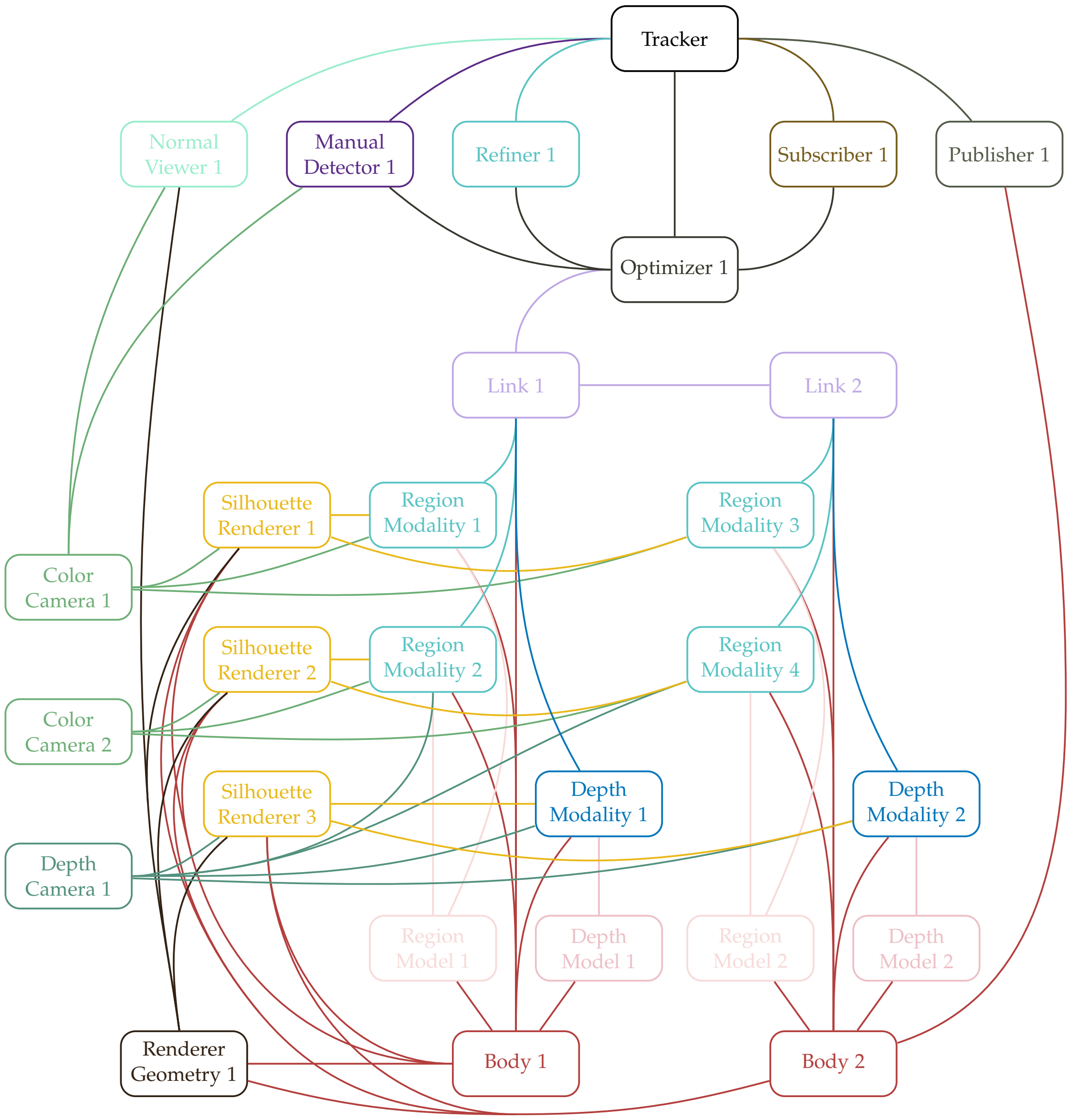


Figure 6.1.: Example configuration for the tracking of a kinematic structure with two *Bodies* using images from two *Color* and one *Depth Camera*. For each *Body* and *Color Camera*, a *Region Modality* is configured. Similarly, each *Body* connects to a *Depth Modality*. To validate points, each *Modality* references a *Silhouette Renderer* that corresponds to a *Camera*. In addition, *Region Modalities 2* and *4* consider occlusions using the *Depth Camera*. For initialization, a *Manual Detector* predicts the pose of *Link 1* and *Body 1* and uses the *Optimizer* to update the kinematic structure. Afterwards, the estimate is improved using a *Refiner*. Also, a *Subscriber* is configured that has access to all objects connected to the *Optimizer*. Finally, results are visualized using a *Normal Viewer*, and pose estimates of *Body 2* are communicated using a *Publisher*.

Algorithm 6.1 Tracking Process

```
 1: while run tracking process do
 2:     Update all Cameras
 3:     Update all Subscribers
 4:     Calculate consistent Link and Body poses
 5:     Run Detecting Step      ▷ For objects associated with the state detecting
 6:     Run Starting Step       ▷ For objects associated with the state starting
 7:     Run Tracking Step       ▷ For objects associated with the state tracking
 8:     Update all Publishers
 9:     Update all Viewers
10: end while
```

before the main process is executed. Also, during operation, the *Tracker* coordinates the execution of methods from different components in the tracking process.

6.2.2. Tracking Process

During execution, the *Tracker* runs methods of different components in the correct order. It thereby considers the state of individual kinematic structures represented by *Optimizers*. In general, they can be in the states *detecting*, *starting*, and *tracking*. Also, kinematic structures may be idle. States can be influenced by the user with interfaces to start the tracking, stop the tracking, and execute the detection. Each method allows to specify names of optimizers to affect the respective kinematic structures.

An overview of the tracking process is given in Algorithm 6.1. Each iteration starts by first updating *Cameras* and *Subscribers*. Subsequently, given that *Subscribers* are able to change the poses of individual objects, the poses of all *Links* and *Bodies* are updated to be consistent. After that, *Detecting*, *Starting*, and *Tracking Steps* are executed. Each step only considers objects associated with kinematic structures in the corresponding state. For example, in the *Starting Step*, only *Modalities* are considered that belong to a kinematic structure in the state *starting*. Finally, once individual steps have been finished, *Publishers* and *Viewers* are updated. The process then starts from the beginning.

In the following, we explain the individual steps of the tracking process in detail. For the *Detecting Step*, first, all *Detectors* associated with the state *detecting* are run. If the detection is successful, the *Detector* updates the poses of all *Links* and *Bodies* in the kinematic structure. Subsequently, for all kinematic structures that were successfully detected, associated *Refiners* improve predictions. Finally, if desired, successfully detected structures move to the state *starting*. An overview of the *Detecting Step* is shown in Algorithm 6.2.

In the *Starting Step*, first, all *Renderers* that are required by *Modalities* are executed. After that, each *Modality* is started. This operation, for example, initializes internal color histograms of *Region Modalities*. Subsequently, the *Starting Step* also performs the initialization of shared *Color Histograms*. Finally, each kinematic structure automatically moves

Algorithm 6.2 Detecting Step

```
1: Run Detectors
2: Run Refiners for successful detections
3: If desired, move to state starting for successful detections
```

Algorithm 6.3 Starting Step

```
1: Run Renderers required for the starting of Modalities
2: Start Modalities
3: Initialize Color Histograms
4: Move to state tracking
```

Algorithm 6.4 Tracking Step

```
 1: for $n$ correspondence iterations do
 2:     Run Renderers required for the calculation of correspondences
 3:     Calculate correspondences for Modalities
 4:     for $m$ update iterations do
 5:         Calculate gradients and Hessians of Modalities
 6:         for all Optimizers do
 7:             Calculate gradients and Hessians of referenced Soft Constraints
 8:             Recursively calculate Jacobians of referenced Links
 9:             Calculate Jacobians and residuals of referenced Constraints
10:             Assemble and solve linear system of equations
11:             Recursively update poses of referenced Links and Bodies
12:         end for
13:     end for
14: end for
15: Run Renderers required for the update of Modalities
16: Update Modalities
17: Update Color Histograms
```

into the state *tracking*. An overview of the individual steps is shown in Algorithm 6.3.

Finally, the main step in the tracking process considers the estimation of poses. This so-called *Tracking Step* is illustrated in Algorithm 6.4. It is a more general version of Algorithm 3.1, which was provided for region-based tracking. The *Tracking Step* performs an iterative computation of correspondences and pose updates. To calculate correspondences for *Modalities*, it first runs required *Renderers*. Typical examples are *Depth Renderers* employed for occlusion handling or *Silhouette Renderers* that are used to validate individual contour or surface points. Afterwards, correspondences are calculated. This is typically followed by two update iterations. In each iteration, gradient vectors and Hessian matrices are calculated for all *Modalities*. Afterwards, kinematic structures are updated. *Optimizers* thereby compute gradient vectors and Hessian matrices of *Soft Constraints*, recursively calculate Jacobians of *Links*, compute Jacobians and residuals of *Constraints*, and, finally, assemble everything in a single linear system of equations. *Optimizers* then solve the equations and recursively update the poses of all *Links* and *Bodies*. Subsequently, the iterations are repeated. Finally, after the last iteration, required *Renderers* are run, and *Modalities* and *Color Histograms* are updated.

In conclusion, the presented tracking process ensures that all individual components work together in an organized way. Given that each method is only executed if it is required, the process also facilitates high efficiency. Furthermore, it does not limit the flexible combination of individual components. Like the architecture, the resulting process, therefore, supports both the theoretical capabilities of our approach and practical requirements for real-world applications.

6.2.3. Initialization

To start tracking, initial poses and configurations are required for all objects and kinematic structures. As described before, setting poses and updating kinematic structures is the task of the *Detector* class. In the *M3T* library, three different implementations exist: a *Static Detector*, *Manual Detector*, and *Automatic Detector*. The *Static Detector* simply assigns a user-defined pose to the root link of a referenced *Optimizer*. It is employed for the initialization of objects that are always at the same location relative to the camera before tracking is started. A typical use case is the initialization of a robot end effector. The robot thereby first moves to a defined pose and configuration relative to the camera. Subsequently, given this defined setup, one executes the detection.

The second version is the *Manual Detector*. It was developed as part of the bachelor's thesis of Reichert (2021). The *Manual Detector* stores four characteristic 3D points that are selected from the surface of the root link's model. Typical examples of points are object corners or characteristic locations of visual texture. During detection, an image from a referenced color camera is shown to a human operator. The operator has to identify the 2D image coordinates of corresponding characteristic points by clicking into the image. The root link's pose can then be inferred using a *Perspective-n-Point (PnP)* algorithm on the obtained 3D-2D point correspondences. In our implementation, the *EPnP* algorithm of Lepetit et al. (2009) is used. *Manual Detectors* are typically employed in cases where initialization with a static pose is impossible and automatic detection is unreliable or requires too much effort to train.

Finally, the third option is the *Automatic Detector*. A basic version of this class was developed during the course of the master's thesis of Rothe (2022). The *Automatic Detector* is able to reference multiple *Optimizers* for which it infers the poses of corresponding root links. It features a two-stage process that is based on deep learning. First, the *YOLOv7* detector (C.-Y. Wang et al. 2022) is used to estimate a bounding box for each object. Based on this initial estimate, 6DoF poses are then predicted in a second step by the *AAEs* approach of Sundermeyer et al. (2018). Both steps are implemented using code

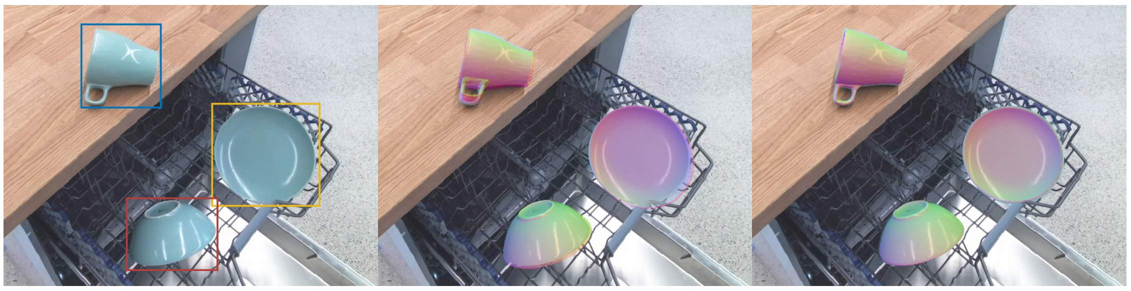


Figure 6.2.: Initialization with the *Automatic Detector* and subsequent refinement. The process is visualized by images from left to right. First, objects are detected using the *YOLOv7* detector (C.-Y. Wang et al. 2022). Afterwards, 6DoF poses are estimated by the *AAEs* algorithm (Sundermeyer et al. 2018). Finally, predictions are refined using our *M3T* approach.

that is publicly available.[2,3] To train the respective networks, significant amounts of data are required. For this, we generate highly-realistic synthetic images via *BlenderProc*[4]. While the generation requires realistic 3D models, synthetic data allows us to avoid the tedious labeling of real-world images, which, especially for 6DoF pose predictions, is unfeasible. A visualization of the pipeline is shown in Fig. 6.2.

Once initial estimates for poses and configurations of objects and kinematic structures were computed, predictions can be refined. In contrast to *Detectors*, which, for example, only consider a single color image, *Refiners* are able to take into account all information that can be used by the tracker. As explained previously, the process of refinement is very similar to the *Tracking Step* in Algorithm 6.4. The main difference compared to tracking is that the *Starting Step* from Algorithm 6.3 is executed in each iteration before correspondences are computed. The reason for this is that no temporal information is available for refinement. Consequently, it is better to reinitialize in each iteration and overwrite information from the previous step, which presumably featured a worse pose prediction. Finally, using the described detectors and our approach for refinement, we are able to obtain accurate poses to initialize the tracking.

6.2.4. Proprioception

Robotic systems typically provide joint measurements that constrain the relative pose between individual links. Examples range from highly accurate joint encoders to very rough predictions from soft or tendon-driven joints. If the error is small, measurements can improve the accuracy of pose estimation. To incorporate that information, it is common to formulate a Gaussian error model with a user-defined standard deviation.

[2]`https://github.com/DLR-RM/AugmentedAutoencoder`
[3]`https://github.com/WongKinYiu/yolov7`
[4]`https://github.com/DLR-RM/BlenderProc`

However, even for large errors, measurements can be highly valuable. In such cases, they often constrain a rough area in which predictions are feasible. By constraining the tracker to this space, one is able to significantly improve robustness. For example, if the area is smaller than typical pose differences from which the tracker can recover, it is possible to completely occlude bodies without losing tracking.

In the following, we create a soft constraint formulation that takes into account both cases. The constraint is typically applied between the tracked body and a parent for which the pose is updated using joint measurements. A detailed description of this is provided later. Like in the exact constraint formulation introduced in Section 5.3.3, soft constraints consider differences between two reference frames A and B, which are defined for the bodies a and b, respectively. Also, as before, pose differences for constraint directions are considered by the vector $\boldsymbol{b}_{ab}$. It features individual components of either the rotation vector ${}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}$ or the translation vector ${}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{B}}$. It is, therefore, a reduced version of the extended constraint equation vector $\bar{\boldsymbol{b}}_{ab}$ that was specified in Eq. (5.19). Based on the difference vector $\boldsymbol{b}_{ab}$, we define the following energy function

$$E_{\mathrm{c}}(\boldsymbol{\theta}_a, \boldsymbol{\theta}_b) = \begin{cases} \frac{1}{2{\sigma_{\mathrm{c}}}^2}\big(\|\boldsymbol{b}_{ab}\|_2 - d_{\mathrm{c}}\big)^2 & \text{if } \|\boldsymbol{b}_{ab}\|_2 > d_{\mathrm{c}} \\ 0 & \text{else} \end{cases}, \tag{6.1}$$

where σ_{c} is a standard deviation and d_{c} is a distance threshold. In the energy function, the distance threshold d_{c} defines a spherical area in which pose differences are not penalized. Outside this sphere, a quadratic error for the distance to the sphere's surface is applied. The magnitude of this error is scaled by the standard deviation σ_{c}. For the special case of $d_{\mathrm{c}} = 0$, the expression corresponds to the log-probability of a normal distribution. Thanks to this, the function is able to model both accurate measurements using a Gaussian error model and rough estimates by constraining an approximate area.

The energy of each soft constraint is considered in the combined energy function $E_{\mathrm{k}}(\boldsymbol{\theta}_{\mathrm{k}})$ in Eq. (5.2). Standard deviations thereby specify the weighting of individual soft constraints compared to other soft constraints and modalities. For the optimization, we, again, require first- and second-order derivatives with respect to the combined variation vector $\boldsymbol{\theta}_{\mathrm{k}}$. They can be computed as follows

$$\left.\frac{\partial E_{\mathrm{c}}}{\partial \boldsymbol{\theta}_{\mathrm{k}}}\right|_{\boldsymbol{\theta}_{\mathrm{k}}=\mathbf{0}} = \left.\frac{\partial E_{\mathrm{c}}}{\partial \boldsymbol{\theta}_a}\frac{\partial \boldsymbol{\theta}_a}{\partial \boldsymbol{\theta}_{\mathrm{k}}} + \frac{\partial E_{\mathrm{c}}}{\partial \boldsymbol{\theta}_b}\frac{\partial \boldsymbol{\theta}_b}{\partial \boldsymbol{\theta}_{\mathrm{k}}}\right|_{\boldsymbol{\theta}_{\mathrm{k}}=\mathbf{0}} \tag{6.2}$$

$$= \boldsymbol{g}_{\mathrm{c}a}^\top \boldsymbol{J}_a + \boldsymbol{g}_{\mathrm{c}b}^\top \boldsymbol{J}_b, \tag{6.3}$$

$$\left.\frac{\partial^2 E_{\mathrm{c}}}{\partial {\boldsymbol{\theta}_{\mathrm{k}}}^2}\right|_{\boldsymbol{\theta}_{\mathrm{k}}=\mathbf{0}} \approx \left.\frac{\partial \boldsymbol{\theta}_a}{\partial \boldsymbol{\theta}_{\mathrm{k}}}^\top \frac{\partial^2 E_{\mathrm{c}}}{\partial {\boldsymbol{\theta}_a}^2}\frac{\partial \boldsymbol{\theta}_a}{\partial \boldsymbol{\theta}_{\mathrm{k}}} + \frac{\partial \boldsymbol{\theta}_b}{\partial \boldsymbol{\theta}_{\mathrm{k}}}^\top \frac{\partial^2 E_{\mathrm{c}}}{\partial {\boldsymbol{\theta}_b}^2}\frac{\partial \boldsymbol{\theta}_b}{\partial \boldsymbol{\theta}_{\mathrm{k}}}\right|_{\boldsymbol{\theta}_{\mathrm{k}}=\mathbf{0}} \tag{6.4}$$

$$\approx \boldsymbol{J}_a^\top \boldsymbol{H}_{\mathrm{c}a} \boldsymbol{J}_a + \boldsymbol{J}_b^\top \boldsymbol{H}_{\mathrm{c}b} \boldsymbol{J}_b, \tag{6.5}$$

where second-order derivatives of the pose variations $\boldsymbol{\theta}_a$ and $\boldsymbol{\theta}_b$ with respect to the combined variation vector $\boldsymbol{\theta}_{\mathrm{k}}$ are neglected. In the derivation, the constraint gradients $\boldsymbol{g}_{\mathrm{c}a}$ and $\boldsymbol{g}_{\mathrm{c}b}$ and the constraint Hessians $\boldsymbol{H}_{\mathrm{c}a}$ and $\boldsymbol{H}_{\mathrm{c}b}$ are projected using the Jacobian matrices $\boldsymbol{J}_a$ and $\boldsymbol{J}_b$. This is very similar to the expression in Section 5.2.2, where gradient vectors and Hessian matrices from modalities are projected in the same way. Consequently, to consider soft constraints, one can simply add the constraint gradients and Hessians of individual bodies to those of corresponding modalities and consider them in the summations in Eqs. (4.5) and (4.6).

The required constraint gradient vectors and Hessian matrices of individual bodies are thereby defined as

$$\boldsymbol{g}_{\mathrm{c}i}^{\top} = \frac{\partial E_{\mathrm{c}}}{\partial \boldsymbol{\theta}_i}\bigg|_{\boldsymbol{\theta}_{\mathrm{k}}=\boldsymbol{0}} = \frac{\partial E_{\mathrm{c}}}{\partial \boldsymbol{b}_{ab}} \frac{\partial \boldsymbol{b}_{ab}}{\partial \boldsymbol{\theta}_i}\bigg|_{\boldsymbol{\theta}_{\mathrm{k}}=\boldsymbol{0}}, \tag{6.6}$$

$$\boldsymbol{H}_{\mathrm{c}i} = \frac{\partial^2 E_{\mathrm{c}}}{\partial \boldsymbol{\theta}_i^2}\bigg|_{\boldsymbol{\theta}_{\mathrm{k}}=\boldsymbol{0}} \approx \frac{\partial \boldsymbol{b}_{ab}}{\partial \boldsymbol{\theta}_i}^{\top} \frac{\partial^2 E_{\mathrm{c}}}{\partial \boldsymbol{b}_{ab}^2} \frac{\partial \boldsymbol{b}_{ab}}{\partial \boldsymbol{\theta}_i}\bigg|_{\boldsymbol{\theta}_{\mathrm{k}}=\boldsymbol{0}}, \tag{6.7}$$

where $i \in \{a, b\}$. Second-order derivatives of the difference vector $\boldsymbol{b}_{ab}$ are neglected. Note that first-order derivatives of $\boldsymbol{b}_{ab}$ with respect ot the pose variation vectors $\boldsymbol{\theta}_a$ and $\boldsymbol{\theta}_b$ were already derived in Section 5.3.3. As before, they can be assembled from rows of the matrices in Eqs. (5.22) and (5.23). In addition, first- and second-order derivatives of the energy function can be computed as follows

$$\frac{\partial E_{\mathrm{c}}}{\partial \boldsymbol{b}_{ab}}\bigg|_{\boldsymbol{\theta}_{\mathrm{k}}=\boldsymbol{0}} = \begin{cases} \frac{1}{\sigma_{\mathrm{c}}^2}\left(1 - \frac{d_{\mathrm{c}}}{\|\boldsymbol{b}_{ab}\|_2}\right)\boldsymbol{b}_{ab}^{\top} & \text{if } \|\boldsymbol{b}_{ab}\|_2 > d_{\mathrm{c}} \\ \boldsymbol{0} & \text{else} \end{cases}, \tag{6.8}$$

$$\frac{\partial^2 E_{\mathrm{c}}}{\partial \boldsymbol{b}_{ab}^2}\bigg|_{\boldsymbol{\theta}_{\mathrm{k}}=\boldsymbol{0}} = \begin{cases} \frac{1}{\sigma_{\mathrm{c}}^2}\left(\left(1 - \frac{d_{\mathrm{c}}}{\|\boldsymbol{b}_{ab}\|_2}\right)\boldsymbol{I} + \frac{d_{\mathrm{c}}}{\|\boldsymbol{b}_{ab}\|_2^3}\boldsymbol{b}_{ab}\boldsymbol{b}_{ab}^{\top}\right) & \text{if } \|\boldsymbol{b}_{ab}\|_2 > d_{\mathrm{c}} \\ \boldsymbol{0} & \text{else} \end{cases}. \tag{6.9}$$

Finally, to consider joint measurements, we typically use a two-stage configuration. First, we define a fixed virtual link at the joint's location that is updated using a subscriber and joint measurements. It represents the mean estimate from the proprioception. Starting from this virtual link, we then define the physical link that models the free joint directions. The two links are then restricted relative to each other using the developed soft constraint. An illustration of this configuration is shown in Fig. 6.3.

In theory, it would also be possible to directly update one of the soft constraint's constraint coordinate frames A or B from joint measurements. However, in that case, the tracker's optimization has to continuously adapt to a changing constraint, always starting from the last configuration. In contrast, using the described method with a virtual link, joint subscribers update the entire kinematic structure before the tracker starts its optimization. Consequently, the tracker only has to adjust to small measurement inaccuracies. Offsets that were previously predicted are thereby preserved. Thanks to this, tracking becomes more robust. In summary, with the developed soft constraints and the described configuration, it is possible to incorporate joint measurements of varying accuracy and reliability. In addition, soft constraints can also be used in other applications where the motion between links should be restricted.

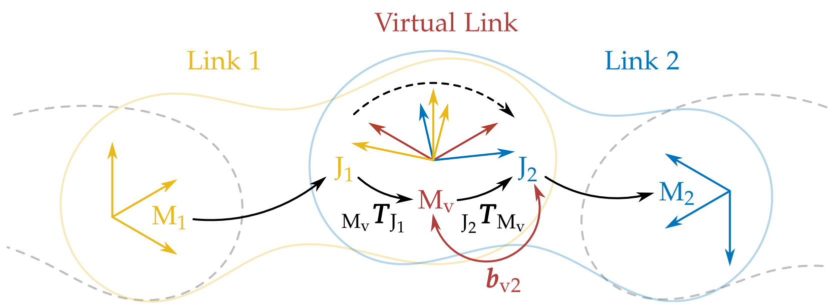


Figure 6.3.: Link configuration to consider joint measurements. The two links are configured with a virtual link in between. While the first transformation ${}_{M_v}\boldsymbol{T}_{J_1}$ is fixed and is updated by a subscriber based on joint measurements, the second transformation ${}_{J_2}\boldsymbol{T}_{M_v}$ allows motion along the joint's direction and is updated by the tracker. The tracker is thereby restricted using a soft constraint on the joint's difference vector $\boldsymbol{b}_{v2}$.

6.3. The Humanoid Robot David

In this section, we present information about the tracker's integration into the humanoid robot *David*. For this, we first provide a basic description of the robot system. This is followed by an explanation of the tracker's configuration. Subsequently, the integration with an existing grasp state estimation approach that considers contact information is discussed. Finally, the section is concluded with the presentation of example applications where the tracker is successfully used.

6.3.1. Robot System

The humanoid robot *David* (Grebenstein et al. 2011) is a fully compliant system that features elastic elements such as springs in each joint. Compared to mechanically stiff robots, this makes the system more robust to impact and safe for human-robot collaboration. The robot consists of two arms with 7 degrees of freedom mounted on top of a torso with 3 degrees of freedom (Reinecke et al. 2020). While the left arm integrates a two-finger gripper, the right arm features a fully actuated five-finger hand that matches that of a human in size. Thanks to this hand, *David* is able to use a wide variety of tools that are made for humans. Also, it can perform dexterous manipulation, where it controls the pose of an object inside the hand. Together with the two-arm setup, the robot is highly versatile and is able to perform a wide variety of tasks. Finally, to perceive its environment, *David* uses an *Azure Kinect* RGB-D camera that provides images at 30 Hz. It is mounted inside the head on top of an elastic neck (Deutschmann et al. 2022). The neck allows for a highly coupled motion in 6DoF space that facilitates the reorientation of the camera. An image of the robot is shown in Fig. 6.4.

For joints, different mechanisms are used with large differences in the quality of sensor measurements. The robot's fingers are driven by tendons for which motors and

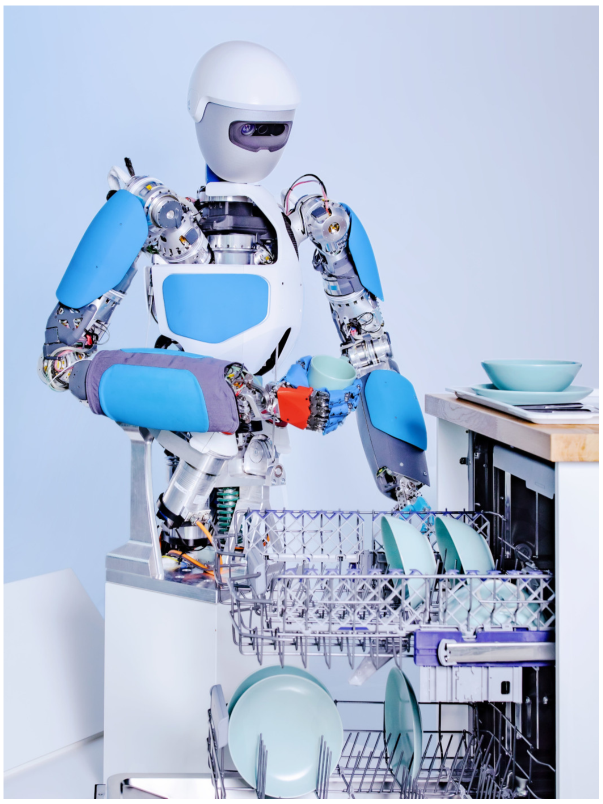


Figure 6.4.: The humanoid robot *David* pictured for the task of unloading a dishwasher.

sensors are placed inside the forearm. Similarly, for the wrist, motors are also in the forearm, but the kinematic is actuated by a mechanism that uses rods instead of tendons. Both the wrist and fingers do not provide absolute measurements. Instead, they have to be initialized in a defined configuration. This, together with inaccuracies, elastic deformation, and high coupling between sensors leads to relatively rough estimates for joint angles. For the arms and the torso, the integration of sensors is quite different. There, absolute joint encoders are directly accommodated in each joint. Consequently, they provide highly-accurate measurements. Finally, the elastic neck consists of a silicon structure that is actuated by four motors via tendons. The motion of the neck is thereby highly coupled. In addition, given the elasticity of the neck together with the weight of the head, the pose transformation depends not only on tendon measurements but also on the current configuration of the torso. Currently, machine learning is used to obtain a rough pose estimate from sensor readings.

Given this setup, it is impossible to compute the exact location of the hand relative to the camera using only forward kinematics. Instead, to facilitate the grasping and manipulation of objects, one has to use visual information and continuously predict the poses of both the hand and manipulated objects. Based on this feedback, one can then use visual servoing techniques.

6.3.2. Tracker Configuration

To use the developed tracking approach on the *David* robot, individual components from the *M3T* library had to be integrated. In general, *David* uses the *Links and Nodes (LN)* middleware to allow independent processes to communicate via topics and services. By default, *LN* is not supported by the *M3T* library. We, therefore, implemented classes that facilitate this communication. To obtain images from the robot, separate *Depth* and *Color Cameras* were implemented that read images from *LN* topics and use services to get intrinsics. Also, a *Subscriber* was created that reads a user-defined joint measurement from a topic and updates a referenced *Link*. Similarly, to communicate results, a *Publisher* was created that writes a *Link's* pose, as well as other information, to a user-defined topic. Finally, for the overall tracking process, services were implemented that are used to start the tracking, stop the tracking, and execute the detection of objects. Also, the process provides a service that configures the tracker from a *YAML* file.

To predict the hand's pose, the created configuration takes into account the entire kinematic chain from the camera to the palm. Starting with a rigid transformation from the camera, first, measurements from the neck are included. The high uncertainty of neck predictions is thereby modeled using a soft constraint with a defined threshold parameter for the rotation. Subsequently, the relative transformation from the neck to the forearm is considered using individual joint measurements. The accumulated uncertainty from joint measurements and the entire mechanical structure is then represented using a soft constraint with defined rotational and translational thresholds. The so-modeled kinematic structure constrains the space of possible poses for the forearm. It helps to significantly reduce the risk of tracking loss. In addition, given that joint measurements are updated before the optimization, the tracker only has to adjust to small inaccuracies instead of full frame-to-frame pose differences.

Finally, based on the constrained space and mean configuration, visual information from individual bodies is included. For the forearm, different region modalities that model the blue patches and the gray background are used. The two region modalities of the blue patches share a single color histogram. For the closed kinematic structure of the wrist, we consider the region information of the three main lever bodies. They are modeled by a single color histogram. In addition, for both the forearm and levers, depth modalities are configured. To constrain the relative motion of the wrist, we, again, consider joint measurements using two soft constraints with defined thresholds. Finally, for the red palm, a single region and depth modality is used. At last, the pose estimate of the palm is communicated to external processes using a publisher.

In addition to pose predictions for the palm, our tracker can also be configured to provide the pose of manipulated objects. Depending on the application and object characteristics, different detectors and modalities can be configured. For example, in the dishwasher demonstration that is pictured in Fig. 6.4, mug, plate, and bowl objects are included. In our configuration, they are considered by region and depth modalities, as well as a single automatic detector that manages initialization. However, given the modularity of the algorithm and the tracker's easy setup via *YAML* files, it is straightforward to modify configurations in order to support other objects and applications.

6.3.3. Grasp State Estimation

Once *David's* hand touches an object, constraints from contacts provide additional information. To consider such cues, a grasp state estimation approach was developed by Pfanne (2022a). Additional information about the method can also be found in preceding publications (Pfanne and Chalon 2017; Pfanne et al. 2018). To estimate the state, the approach requires contact points. For the detection, different methods exist. One way is to use current estimates of joint positions and the object's pose to compute contact points on the surface of intersecting geometry. In addition to inferring contacts from the current state estimate, the method also allows to sense contacts. For this, information from joint torque measurements and tactile sensors is considered. Because tactile sensors are not available for *David*, in our case, only joint torques are used.

The grasp state estimation predicts both joint position errors and the object pose. The approach thereby ensures that the entire grasp state is consistent. For the estimation, an *EKF* is used. It provides a probabilistic formulation that models the current belief of the grasp state by its mean and covariance. In addition to a motion model, which propagates the belief over time, the *EKF* includes a measurement step. It incorporates measurements in a probabilistic formulation that maximizes the probability of the estimated state. In case of the discussed grasp state estimation, the main measurements that are considered are distances between contact points. For valid contacts, the distance should be zero. Given the general formulation of the *EKF*, the measurement model can, however, also incorporate additional information.

To make the grasp state estimation more accurate and robust, we decided to include predictions from our tracker. In general, the measurement model requires a mean estimate, as well as a covariance matrix that models uncertainty. For our approach, both are available. We thereby use the knowledge that the negative inverse variance of a Gaussian distribution directly corresponds to the Hessian matrix. Given that in our tracker, most errors are assumed to be normally distributed, the negative inverse Hessian of the tracked object is provided as an estimate for the covariance matrix. This allows us to transfer the full probabilistic information from the tracker to the *EKF*. Finally, we also implemented a subscriber that updates the tracker's object pose from the prediction of the state estimation. Especially for cases with large occlusions, this ensures that the tracker cannot lose an object as long as the state estimation provides reasonable results. Consequently, both the tracker and state estimation benefit from the collaboration with improved accuracy and robustness.

6.3.4. Example Applications

During its development, the *M3T* library was used in various applications on the robot *David*. In the following, we present examples of the two most important scenarios:

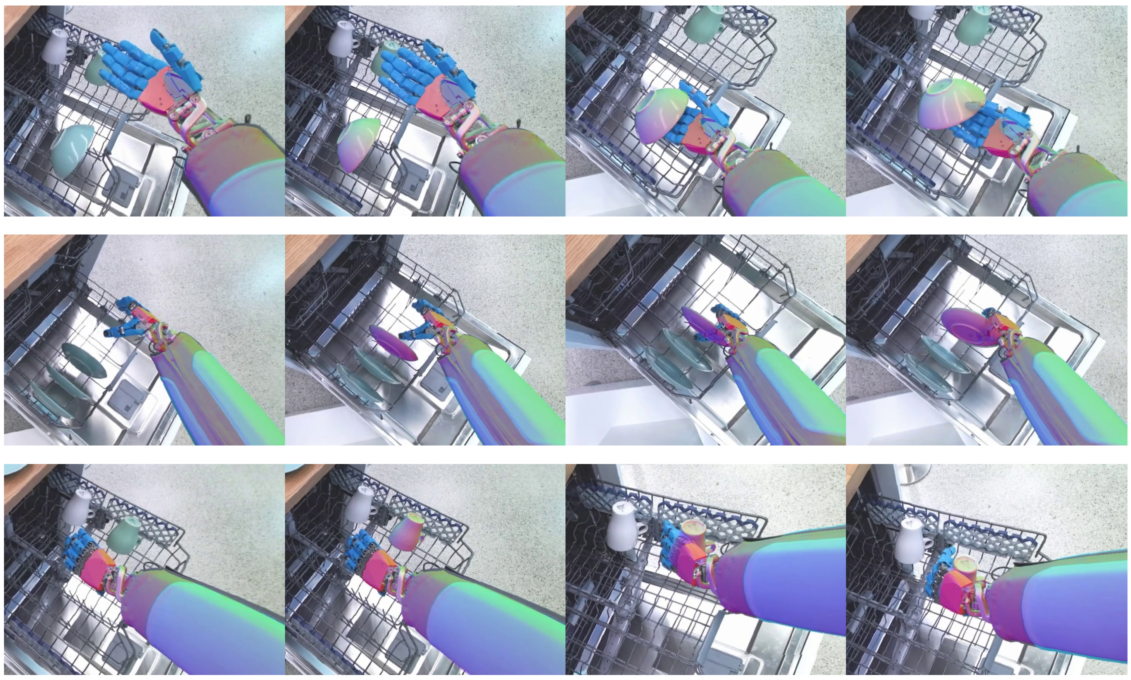


Figure 6.5.: Example images that show the grasping of objects in the *David* dishwasher demonstration. Predictions from the tracker are rendered as an overlay on top of color images. In the first row, a bowl object is shown, while the second and third row feature plate and mug objects, respectively. At the beginning of each sequence, *David* only has a rough expectation of where dishes are placed. First, it, therefore, has to detect objects that should be manipulated. Based on pose estimates for the palm and dish, the robot then approaches the object. Finally, *David* grasps the dish and moves it out of the dishwasher.

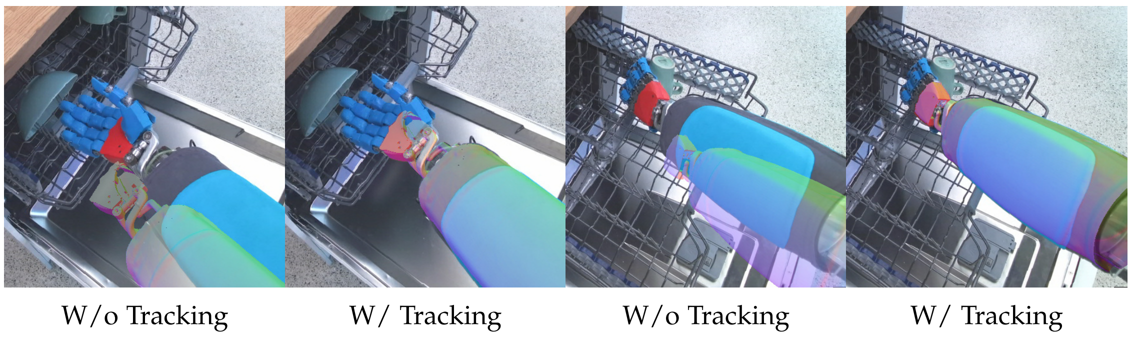


Figure 6.6.: Example images that show pose predictions from the tracker in comparison to forward kinematics without tracking as rendered overlays.

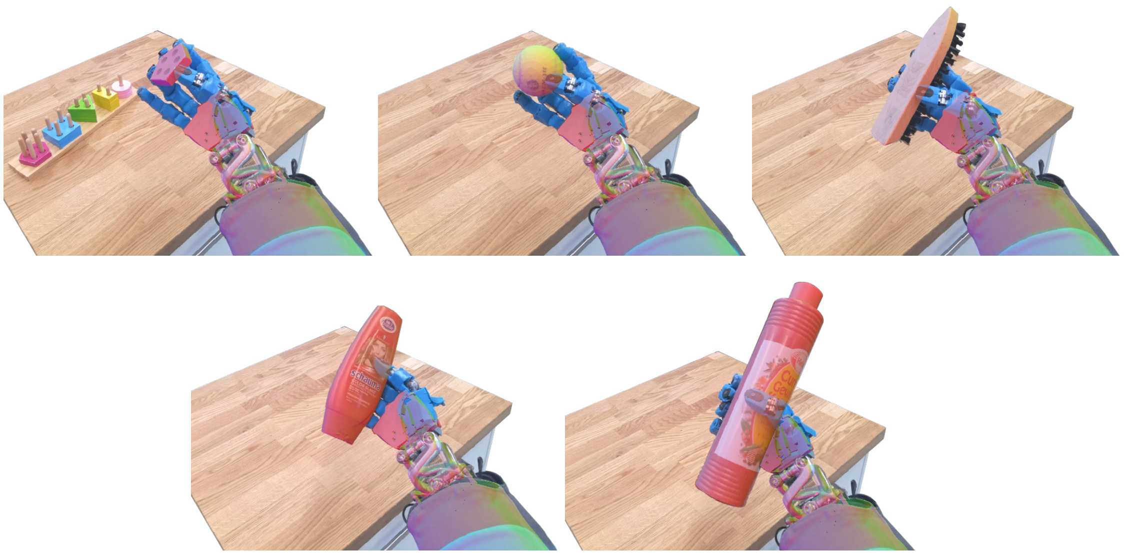


Figure 6.7.: Examples for in-hand manipulation. Pose estimates for the hand and object are rendered as an overlay on top of color images.

grasping and in-hand manipulation. In the first application, *David* is used to unload a dishwasher. For this, it has to detect individual dishes, grasp them, and place them outside the dishwasher. Example sequences that demonstrate the detection and grasping of a bowl, plate, and mug object are shown in Fig. 6.5. While for an industrial robot with perfect forward kinematics and hand-eye calibration, it might be sufficient to use global 6DoF pose estimation and perform the task blindly, for *David*, this is not an option. As explained before, the main reason is that because of various design considerations, the pose of the hand relative to the camera is not perfectly known from joint measurements alone. To illustrate this, we show examples of pose predictions for the hand with and without tracking in Fig. 6.6. The images clearly demonstrate that, without tracking, *David* would not be able to grasp detected objects.

Finally, for the task of in-hand manipulation, continuous information about the object's pose is equally important. The goal of in-hand manipulation is to move an object into a target pose within the hand using only the robot's fingers. To achieve this goal, an impedance-based object controller was implemented on *David* (Pfanne et al. 2020; Pfanne 2022b). It computes joint forces for the fingers based on the difference between the current and the desired object pose. The controller thereby uses the previously discussed approach for grasp state estimation, which utilizes our tracker's predictions for the hand and object pose. Examples of objects that were used in in-hand manipulation experiments are shown in Fig. 6.7. As for grasping, accurate in-hand manipulation is only possible by closing the perception-action loop, using pose estimates for both the object and hand.

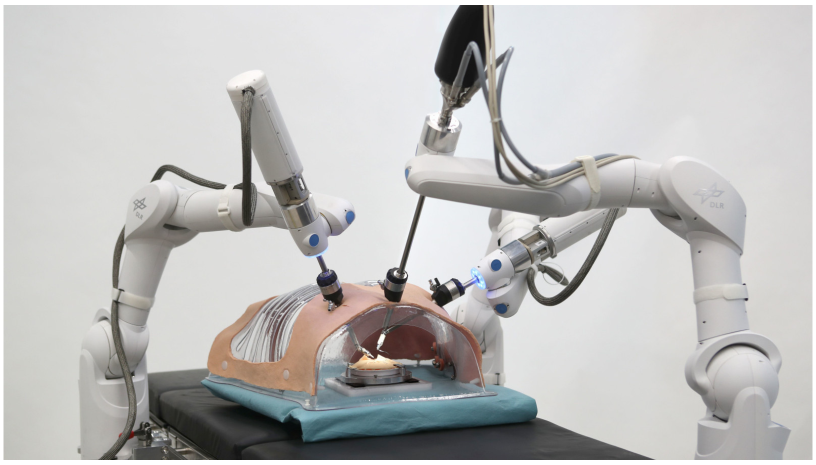


Figure 6.8.: Typical setup of the *MiroSurge* system for minimal-invasive robotic surgery with three *MIRO* robots, two *MICA* instruments, and one stereo endoscope.

6.4. The MiroSurge System

The following section discusses the integration of the tracker into the *MiroSurge* system for minimal-invasive robotic surgery. For this, first, a basic description of the system and the used robots is provided. This is followed by an explanation of the tracker's configuration. Finally, we present experimental results for the tracking of the *MICA* instrument and discuss future use cases.

6.4.1. Robot System

The *MiroSurge* system features multiple *MIRO* robots, which are highly-versatile robot arms with 7 degrees of freedom (Hagn et al. 2008; Hagn et al. 2010). In general, such robots can be used in various surgical applications, with the most prominent being minimal-invasive robotic surgery. In a typical setup, two *MICA* instruments are mounted on two robot arms, while the third *MIRO* robot features a stereo endoscope. It provides images at a framerate of 50 Hz. An example of the setup is shown in Fig. 6.8. In general, *MICA* instruments consist of a drive unit and a task-specific tool (Thielmann et al. 2010). While various tools can be used that provide different functionality, in the following, we focus on the most prominent version. It includes pliers and a 2 degrees of freedom wrist, which is mounted at the end of a long and thin shaft. Thanks to the general setup and its flexible configuration with different tools, the *MiroSurge* system is highly versatile and can be used in various scenarios.

Currently, the system is operated by a surgeon who views images from the endoscope via a 3D display. *MIRO* robots and *MICA* instruments are thereby operated using two haptic input devices. In the future, surgeons should be assisted with virtual fixtures or automatic functionality. For this, end effector poses relative to the endoscope have to be known. However, mechanical structures along the kinematic chain, which includes instruments, robots, mounts, and the table, are not perfectly stiff. Also, while the *MIRO* robot provides accurate measurements from joint encoders, for the *MICA* system, the situation is quite different. Joints for the instrument's wrist and pliers are connected to sensors and motors in the drive unit via long tendons and a mechanical tool interface. Both elastic elongation in tendons and inaccuracies in guides are difficult to model. As a consequence, estimates for the instrument's joint angles are imprecise.

Given the described design, it is impossible to compute the exact location of the end effector using forward kinematics alone. To overcome this problem, like the surgeon, we want to use visual information to estimate the *MICA* instrument's pose and configuration. Based on those predictions, one can then place virtual fixtures for the surgeon or even completely automate individual tasks using visual servoing techniques.

6.4.2. Tracker Configuration

In the following, we describe how individual components from the *M3T* library were integrated. Like the robot *David*, the *MiroSurge* system uses the *LN* middleware for processes to communicate via topics and services. To facilitate this type of communication, specific implementations were created for required classes. First, a *Subscriber* was developed that incorporates user-defined joint measurements and updates a referenced *Link* based on this information. Also, for the communication of results, the same *Publisher* class was used as for *David*, which writes the pose and other information of a referenced *Link* to a user-defined topic. Finally, a *Color Camera* was created that allows to obtain color images from a single camera of the stereo endoscope.

To predict the pose and configuration of the *MICA* end effector, the entire kinematic chain from the camera to the pliers is considered. First, the relative transformation from the endoscope, which is mounted on one of the *MIRO* robots, to the shaft of the *MICA* instrument, which is attached to another *MIRO* robot, is computed. The relative pose between the robots on the operating table is assumed to be known. Starting from the computed transformation, the accumulated uncertainty of the shaft's pose is considered using a soft constraint with a defined rotational and translational threshold. The constraint limits the space of possible poses for the end effector and significantly helps to reduce the risk of tracking loss.

Starting from the shaft, visual information is considered using region modalities for each body. Metallic components from the wrist and the shaft thereby share a single color histogram. Similarly, a single color histogram is also used for the pliers. Also, the final wrist link, which features a plastic cylinder in addition to metallic parts, is modeled by two regions. To constrain the motion of the wrist and the pliers relative to predictions from joint measurements, soft constraints with defined rotational thresholds are configured. In addition, another soft constraint prohibits the intersection of the pliers. Finally, a publisher communicates the pose estimate of the *MICA* end effector's final wrist link to external processes. Note that while it is not yet required, publishing other information, such as the configuration of the pliers, would also be possible.

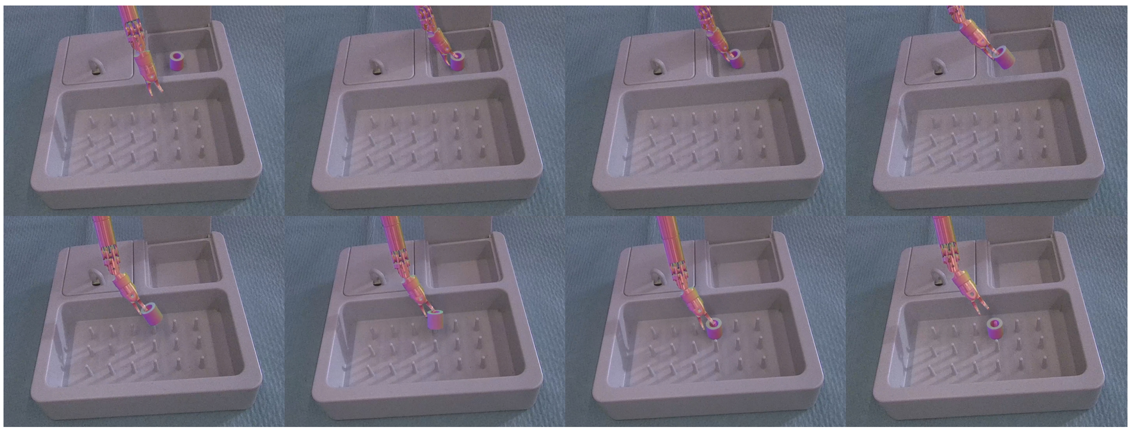


Figure 6.9.: Example images showing the manipulation of a cylinder in a *Lübecker Toolbox* module using the *MICA* instrument. Both the cylinder and robot end effector are tracked. Estimates from the tracker are rendered as an overlay on top of color images.

In addition to predicting the pose of the *MICA* end effector, the tracker can also be configured to estimate the poses of external objects. For example, in the training of surgeons for minimal-invasive surgery, the *Lübecker Toolbox* is used. It includes exercises that consider the manipulation of cylinders within different toolbox modules. To provide automatic capabilities for such scenarios, it is required to track the robot end effector, a toolbox module, and cylinders. As before, given the modularity of the tracker, this can be achieved by simply adjusting the configuration in the *YAML* file.

6.4.3. Example Experiments

While pose estimates from our tracker were not yet integrated into any final application on the *MiroSurge* system, it is envisioned to use predictions for the placement of virtual fixtures and to facilitate automatic capabilities. Virtual fixtures can, for example, limit the space in which the end effector can be operated, prohibiting unintended movement and increasing safety. Also, given a target, it could be possible to anticipate desired end effector poses and support the surgeon in reaching an appropriate robot configuration. Finally, it might even be possible to automate individual tasks completely.

Based on those goals, experiments were conducted where the integrated tracker was used to predict the pose and configuration of the *MICA* end effector, as well as a cylinder that should be manipulated. The detection of objects was realized using a manual detector. Example sequences of the experiment are shown in Fig. 6.9. For better visualization, we do not track the *Lübecker Toolbox* module. Note, however, that tracking the object would not be a problem. Given the depicted sequence, it is easy to imagine an application where manipulation tasks that are typically executed by a surgeon during training are fully automated. As in the example, our tracker would thereby estimate the pose and configuration of all relevant objects. Predictions could then be used by control algorithms to facilitate the desired manipulation and close the perception-action loop.

7 Conclusion

7.1. Summary

For robotic manipulation, it is often assumed that the world is static and forward kinematics are perfectly known. This expectation significantly limits both the complexity of manipulation tasks and the design of robotic hardware. For example, the static world assumption expects an environment without unpredictable changes, where the motion of all objects is fully deterministic. For many applications and tasks, this is impossible to fulfill. Moreover, the requirement of perfect forward kinematics prohibits many robotic designs, including lightweight structures, elastic elements, or tendon-driven joints. To overcome such limitations, we believe that it is essential to close the perception-action loop and adopt more human-like strategies, such as visual servoing. Techniques thereby typically require continuous visual feedback for relevant objects as well as the robot's end effector. To provide such information in the form of estimated poses and configurations, the presented work developed and studied 3D object tracking techniques that comply with requirements for advanced robotic manipulation.

After a detailed survey of existing methods, we found that region-based tracking techniques are particularly well-suited for robotic manipulation. The main reason is that most robots, as well as many man-made objects, do not provide rich texture, which is not a problem for region-based methods. Also, they are robust to motion blur and background clutter. However, most methods feature computationally expensive dense formulations that contradict our requirements for efficiency and speed. To overcome such limitations, we developed a sparse region-based approach that is highly efficient. It uses correspondence lines to predict discrete probability distributions for the object's contour location. In addition to efficiency, one of the main advantages compared to dense methods is that, because of the one-dimensionality of correspondence lines, it is possible to sample values over probability distributions in reasonable time. This allows us to better consider uncertainty and estimate accurate first- and second-order derivatives for fast-converging Newton optimization. Also, to take into account the segmentation's local and global uncertainty, we proposed novel smoothed step functions and provided a detailed theoretical analysis. Finally, based on individual correspondence lines, the joint probability is maximized using Newton optimization with Tikhonov regularization and a sparse representation of the object geometry. In multiple experiments, we found that the resulting approach is not only significantly more efficient than previous methods but also that it achieves superior tracking results.

In addition to region, one other source of information that can be used for the tracking of textureless objects is depth. Also, if it is available, texture provides highly-valuable data that constrains an object's pose. Furthermore, many objects do not have a uniform surface. Instead, they feature multiple distinct materials and colors, which results in additional region information. Based on the observation that those sources are highly complementary, we developed a multi-modality tracking approach. It features a probabilistic formulation that integrates an *ICP*-based depth modality, a keypoint-based texture modality, and an extension of our region approach to multi-region tracking. For the method, only a textureless mesh of the object is required. This is highly advantageous for its usability. Also, the approach is able to consider multiple cameras. In an extensive evaluation, we found that the resulting algorithm outperforms all existing methods with respect to efficiency and speed. Furthermore, even for purely geometry-based tracking, which only uses region and depth, our approach achieves state-of-the-art results. The conducted experiments also provide new insights into deep learning-based methods. They show that such approaches do not yet surpass classical techniques, even with a textured 3D model, significant computational resources, and vast training data. Finally, we also demonstrated that our algorithm is able to provide better results than global 6DoF pose estimation methods at a fraction of the computational cost. This highlights the general value of 3D object tracking. Also, to improve results for global 6DoF pose estimation, our approach can be used for highly-efficient pose refinement.

In general, robots consist of multiple links that are connected by joints. Also, many real-world objects, such as furniture and appliances, combine individual bodies. In order to successfully track such objects, it is highly important to consider kinematic information. We, therefore, developed a novel framework that allows the extension of existing 6DoF tracking algorithms to multi-body objects. The approach provides a single formulation that projects the six degrees of freedom of individual bodies to a minimal parameterization. In addition, it facilitates the integration of pose constraints. Given the efficiency of projection and the flexibility of constraints, the framework combines the best of two worlds. For the developed constraints, we were able to prove that they enforce an exact kinematic solution for which pose differences converge in a single iteration. To the best of our knowledge, it is the first formulation that allows to accurately model closed kinematic structures. Finally, with the proposed framework, we extended the previously developed multi-modality tracking approach to multi-body tracking. Also, for the evaluation of the algorithm, we created the *RTB* dataset. It is a highly-realistic synthetic dataset that features various robots and a large number of sequences. Given the dataset, we were able to conduct multiple experiments that demonstrate the excellent performance of our framework and multi-body tracker.

To mirror the flexibility of our approach in the algorithm's implementation, we developed the multi-body, multi-modality, and multi-camera tracking library *M3T*, which we made available as open-source software[1]. It features a highly-modular architecture that allows the flexible combination of various components, such as cameras, links,

[1]`https://github.com/DLR-RM/3DObjectTracking`

constraints, modalities, viewers, detectors, refiners, publishers, and subscribers. For automatic initialization, the library integrates a deep learning-based global detection and pose estimation algorithm. Also, we developed an approach that allows the incorporation of joint measurements with varying accuracy and reliability. It combines a Gaussian error model and inequality conditions that ensure that the error is only considered outside a user-defined interval.

Finally, for the real-world validation of our approach, we provided two example applications. In the first, *M3T* was integrated into the humanoid robot *David*, where it is used to predict the poses of manipulated objects and the robot's palm. To improve robustness, the algorithm tracks the closed kinematic structure of the palm, forearm, and wrist. Also, it integrates joint measurements of varying quality. Moreover, the tracker provides pose and uncertainty predictions to a grasp state estimation approach that takes into account finger contacts. The resulting system was successfully employed in multiple demonstrations, facilitating grasping and in-hand manipulation. In the second application, *M3T* was integrated into the *MiroSurge* system. It is used to predict the poses of manipulated objects and the pose and configuration of the *MICA* instrument. Like for *David*, joint measurements are employed to constrain the pose and configuration of the end effector. The resulting system showed great promise for the placement of virtual fixtures and the facilitation of automatic capabilities.

In summary, in this thesis, we developed a highly-modular 3D object tracking approach that complies with requirements for robotic manipulation. It outperforms existing methods on multiple datasets, both with respect to quality and efficiency. The overall approach was implemented in the open-source library *M3T*, which can be configured for numerous applications. The library not only supports a wide range of kinematic structures, object characteristics, and camera setups but also allows for an easy extension with new capabilities. Given those properties and the response to already published code[1], we are confident that our approach will find many applications in the real world. In many cases, it will thereby help to close the perception-action loop and allow robots to adopt more human-like strategies for object manipulation.

7.2. Future Work

While we developed a reliable and versatile 3D object tracking approach for robotic manipulation, multiple directions for improvements and extensions remain. For the developed approach, it could, for example, be useful to consider information from non-static cameras that are connected to links. Also, for some objects and scenarios, it might be advantageous to incorporate additional information. For example, it is possible to develop new modalities based on techniques such as direct optimization or edge features. Similarly, it might also be possible to improve existing modalities. For the region modality, one could, for example, replace global color histograms with more descriptive region statistics. Furthermore, to improve the tracking in cases of large frame-to-frame pose differences, a motion model could be integrated. However, while we believe that those smaller improvements are worth pursuing, in our opinion, the following main research directions are most promising for the future:

Long-term Consistency Because of the simpler task, 3D object tracking is more efficient and provides more consistent results than global 6DoF pose estimation. However, given that only local information is considered, even the best 3D object tracker will lose an object if it moves outside the image or is fully occluded. To recover from such cases, global 6DoF pose estimation is required. In a typical scenario, the human operator thereby detects tracking loss, triggers a new global estimation, assesses if the pose is valid, and reinitializes the tracker. Given the level of manual intervention, this process is far from optimal. Nevertheless, even if it is replicated in software, we believe that it is not perfect. The main reason is that one always needs to decide on a single prediction, even in cases of very little information. To mitigate this problem, a possible solution is the integration of global pose estimation with multi-hypothesis tracking that continuously spawns new estimates, evaluates existing ones, and destroys unlikely or converging hypotheses. As a result, in case of uncertainty, the algorithm would be able to consider multiple options until more information is available. In addition to automatic reinitialization, this has the potential to further improve long-term consistency.

Contact Information In the real world, objects cannot intersect. In addition to visual information, this basic principle further constrains an object's pose. For example, when grasping an object, the space of possible locations is significantly reduced by the robot's hand or gripper. In this thesis, we already showed the potential of contact information on the robot *David*, where we integrated tracking results into an existing grasp state estimation approach that is based on geometry. However, while both algorithms exchange final predictions, the optimization is conducted independently. To further improve results, the tracker needs to consider contact information in the same way as visual measurements. We, therefore, believe that a general approach is needed that prevents object interpenetration and that is deeply integrated into the tracker. In addition, like the grasp state estimation of Pfanne (2022a), the approach should also enforce known object contacts, which can be estimated using force feedback sensors or tactile skin. In summary, this could create a single approach that allows robots, like humans, to consider both visual and tactile feedback.

Data-driven Techniques In most applications, real-world data with accurate pose annotations is not available. Nevertheless, with procedural rendering pipelines, it is possible to create highly-realistic synthetic datasets in reasonable time. While it can still be tedious to obtain accurate object models, create sensible trajectories, have realistic camera data, and define reasonable object interactions, for some applications, it might be worth the effort. In the simplest scenario, such data could be used for the automatic tuning of existing methods. However, in most cases, it would probably be utilized for the training of deep neural networks. Given the expressive power of such methods, we believe that, in the long term, they will outperform conventional algorithms with respect to tracking quality. However, with high computational cost, it is not clear if they can also compete regarding efficiency. In this respect, it might be important to consider where the deployment of neural networks is most beneficial. For example, current render-and-compare approaches predict pose updates directly from image differences. Given that changes between images are mapped by the object's geometry and pose, the network has to implicitly encode the entire geometry, which is probably inefficient. Finally, we want to highlight the big advantage of Newton optimization. It not only provides fast convergence but is also highly modular. Instead of creating a fully deep learning-based algorithm that only uses neural networks, we believe that it might be advantageous to combine predictions from deep neural networks with existing formulations to predict gradient vectors and Hessian matrices. The pose could then be optimized using conventional Newton optimization. As a consequence, the advantage of modularity would be preserved, and principles presented in this work could still be used.

Appendices

A. Extended Probabilistic Model

In the following, we establish the relation between the smoothed step functions proposed in Section 3.2.4 and an extended probabilistic model with $m \in \{m_\text{f}, m_\text{b}, m_\text{n}\}$. For the derivation, we start from an extended definition of the pixel-wise posterior probability

$$p(m_i \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid m_i)p(m_i)}{\sum_{j\in\{\text{f},\text{b},\text{n}\}} p(\boldsymbol{y} \mid m_j)p(m_j)}, \quad i \in \{\text{f}, \text{b}, \text{n}\}, \tag{A.1}$$

where, in contrast to Eq. (3.4), a noise model m_n is considered in addition to the foreground and background models m_f and m_b. Using the parameter $\alpha_\text{h} \in [0, 0.5]$, the model probabilities are defined as

$$p(m_\text{f}) = p(m_\text{b}) = \alpha_\text{h}, \tag{A.2}$$

$$p(m_\text{n}) = 1 - 2\alpha_\text{h}. \tag{A.3}$$

For the conditional color probability of the noise model, the conditional probabilities with respect to the foreground and background are simply combined as follows

$$p(\boldsymbol{y} \mid m_\text{n}) = \frac{1}{2}\big(p(\boldsymbol{y} \mid m_\text{f}) + p(\boldsymbol{y} \mid m_\text{b})\big). \tag{A.4}$$

Introducing the definitions from Eqs. (A.2) to (A.4) into Eq. (A.1) and performing some simplifications results in the following pixel-wise posterior probabilities for the foreground and background model

$$p(m_i \mid \boldsymbol{y}) = \frac{2\alpha_\text{h} p(\boldsymbol{y} \mid m_i)}{p(\boldsymbol{y} \mid m_\text{f}) + p(\boldsymbol{y} \mid m_\text{b})}, \quad i \in \{\text{f}, \text{b}\}, \tag{A.5}$$

and the following constant pixel-wise posterior probability for the noise model

$$p(m_\text{n} \mid \boldsymbol{y}) = p(m_\text{n}) = 1 - 2\alpha_\text{h}. \tag{A.6}$$

Similar to Eq. (3.6), the extended posterior probability can be calculated as follows

$$p(d \mid r, \boldsymbol{y}) \propto \sum_{i\in\{\text{f},\text{b},\text{n}\}} p(r \mid d, m_i)p(m_i \mid \boldsymbol{y}). \tag{A.7}$$

To abbreviate some of the terms in Eq. (A.5), the following definition for pixel-wise posterior probabilities from Eq. (3.5) is introduced

$$p_i(r) = \frac{p(\boldsymbol{y} \mid m_i)}{p(\boldsymbol{y} \mid m_\text{f}) + p(\boldsymbol{y} \mid m_\text{b})}, \quad i \in \{\text{f}, \text{b}\}. \tag{A.8}$$

Together with this abbreviation, we use the derived pixel-wise posterior probabilities from Eqs. (A.5) and (A.6) to write the posterior probability in Eq. (A.7) as follows

$$p(d \mid r, \boldsymbol{y}) \propto\ 2\alpha_\mathrm{h} h_\mathrm{f}(r-d) p_\mathrm{f}(r) + 2\alpha_\mathrm{h} h_\mathrm{b}(r-d) p_\mathrm{b}(r) + \frac{1}{2}(1-2\alpha_\mathrm{h}), \tag{A.9}$$

where a constant probability $p(r \mid d, m_\mathrm{n}) = \frac{1}{2}$ was adopted to model the indifference of the line coordinate r given the noise model m_n, and where the smoothed step functions h_f and h_b model the line coordinate probabilities $p(r \mid d, m_\mathrm{f})$ and $p(r \mid d, m_\mathrm{b})$. Using a general definition of symmetric smoothed step functions

$$h_\mathrm{f}(x) = \frac{1}{2} - f(x), \tag{A.10}$$

$$h_\mathrm{b}(x) = \frac{1}{2} + f(x), \tag{A.11}$$

and the identity

$$p_\mathrm{f}(r) + p_\mathrm{b}(r) = 1, \tag{A.12}$$

we extend Eq. (A.9) as follows

$$\begin{aligned} p(d \mid r, \boldsymbol{y}) \propto\ & \alpha_\mathrm{h} p_\mathrm{f}(r) - 2\alpha_\mathrm{h} f(r-d) p_\mathrm{f}(r) + \alpha_\mathrm{h} p_\mathrm{b}(r) + 2\alpha_\mathrm{h} f(r-d) p_\mathrm{b}(r) \\ & + \frac{1}{2}\big(p_\mathrm{f}(r) + p_\mathrm{b}(r)\big) - \alpha_\mathrm{h}\big(p_\mathrm{f}(r) + p_\mathrm{b}(r)\big). \end{aligned} \tag{A.13}$$

This can then be simplified to

$$p(d \mid r, \boldsymbol{y}) \propto\ \left(\frac{1}{2} - 2\alpha_\mathrm{h} f(r-d)\right) p_\mathrm{f}(r) + \left(\frac{1}{2} + 2\alpha_\mathrm{h} f(r-d)\right) p_\mathrm{b}(r). \tag{A.14}$$

Finally, after introducing the slope function $f(x) = \frac{1}{2}\tanh\left(\frac{x}{2s_\mathrm{h}}\right)$, we obtain

$$p(d \mid r, \boldsymbol{y}) \propto\ \left(\frac{1}{2} - \alpha_\mathrm{h}\tanh\left(\frac{r-d}{2s_\mathrm{h}}\right)\right) p_\mathrm{f}(r) + \left(\frac{1}{2} + \alpha_\mathrm{h}\tanh\left(\frac{r-d}{2s_\mathrm{h}}\right)\right) p_\mathrm{b}(r), \tag{A.15}$$

which is the same probability function as the one derived in Section 3.2.4. Note, however, that instead of a noise model m_n, in Section 3.2.4 the smoothed step functions h_f and h_b from Eqs. (3.14) and (3.15) were used to take into account a defined constant uncertainty. In conclusion, this shows that introducing a noise model m_n and using the foreground and background probabilities $p(m_\mathrm{f}) = p(m_\mathrm{b}) = \alpha_\mathrm{h}$ is equivalent to the incorporation of a simple amplitude parameter α_h in the smoothed step functions.

B. Derivative of Log-Posterior

To analyze the posterior probability distribution, it is desirable to have a closed-form solution that allows an easy interpretation. In the following, we will thus derive a general formulation for the first-order derivative of the log-posterior, which will then be used in Appendix C to calculate the posterior probability distribution for specific parameter configurations. For the derivation, we assume a contour at the line center and perfect step functions for the pixel-wise posterior probabilities defined by

$$p_{\mathrm{f}}(r) = \frac{1}{2} - \frac{1}{2}\operatorname{sgn}(r), \tag{B.1}$$

$$p_{\mathrm{b}}(r) = \frac{1}{2} + \frac{1}{2}\operatorname{sgn}(r). \tag{B.2}$$

Also, we consider infinitesimally small pixels and write the posterior probability distribution from Eq. (3.16) in continuous form for an infinite correspondence line

$$p(d \mid \omega, \boldsymbol{l}) \propto \prod_{r=-\infty}^{\infty} \big(h_{\mathrm{f}}(r-d)p_{\mathrm{f}}(r) + h_{\mathrm{b}}(r-d)p_{\mathrm{b}}(r)\big)^{\mathrm{d}r}. \tag{B.3}$$

Starting from those assumptions, we first convert the product integral to the classical Riemann integral

$$p(d \mid \omega, \boldsymbol{l}) \propto \exp\bigg(\int_{r=-\infty}^{\infty} \ln\big(h_{\mathrm{f}}(r-d)p_{\mathrm{f}}(r) + h_{\mathrm{b}}(r-d)p_{\mathrm{b}}(r)\big)\,\mathrm{d}r\bigg). \tag{B.4}$$

The integral is then split at $r = 0$, and the pixel-wise posterior probabilities from Eqs. (B.1) and (B.2) are introduced

$$p(d \mid \omega, \boldsymbol{l}) \propto \exp\bigg(\int_{r=-\infty}^{0} \ln\big(h_{\mathrm{f}}(r-d)\big)\,\mathrm{d}r + \int_{r=0}^{\infty} \ln\big(h_{\mathrm{b}}(r-d)\big)\,\mathrm{d}r\bigg). \tag{B.5}$$

Finally, we substitute $x = r - d$ to write

$$p(d \mid \omega, \boldsymbol{l}) \propto \exp\bigg(\int_{x=-\infty}^{-d} \ln\big(h_{\mathrm{f}}(x)\big)\,\mathrm{d}x + \int_{x=-d}^{\infty} \ln\big(h_{\mathrm{b}}(x)\big)\,\mathrm{d}x\bigg). \tag{B.6}$$

The first-order derivative with respect to d of the log-posterior can now be calculated using Leibniz's rule for differentiation under the integral

$$\frac{\partial \ln\big(p(d \mid \omega, \boldsymbol{l})\big)}{\partial d} = -\ln\big(h_{\mathrm{f}}(-d)\big) + \ln\big(h_{\mathrm{b}}(-d)\big). \tag{B.7}$$

We then adopt the definitions of the smoothed step functions from Eqs. (3.14) and (3.15) to write

$$\frac{\partial \ln\big(p(d \mid \omega, \boldsymbol{l})\big)}{\partial d} = -\ln\bigg(\frac{1}{2} - \alpha_{\mathrm{h}}\tanh\Big(\frac{-d}{2s_{\mathrm{h}}}\Big)\bigg) + \ln\bigg(\frac{1}{2} + \alpha_{\mathrm{h}}\tanh\Big(\frac{-d}{2s_{\mathrm{h}}}\Big)\bigg). \tag{B.8}$$

Finally, using the inverse hyperbolic tangent

$$2\tanh^{-1}(x) = -\ln\Big(\frac{1}{2} - \frac{x}{2}\Big) + \ln\Big(\frac{1}{2} + \frac{x}{2}\Big), \tag{B.9}$$

one is able to write the following closed-form expression for the first-order derivative of the log-posterior

$$\frac{\partial \ln\big(p(d \mid \omega, \boldsymbol{l})\big)}{\partial d} = -2\tanh^{-1}\bigg(2\alpha_{\mathrm{h}}\tanh\Big(\frac{d}{2s_{\mathrm{h}}}\Big)\bigg). \tag{B.10}$$

C. Closed-form Posteriors

Building on Appendix B, we derive closed-form posterior probability distributions for the two edge cases with either the amplitude parameter $\alpha_\mathrm{h} = \frac{1}{2}$ or the slope parameter $s_\mathrm{h} \to 0$. We thereby start from the closed-form first-order derivative of the log-posterior that is given in Eq. (B.10). The full distribution can then be calculated using the following integration with a subsequent normalization

$$p(d \mid \omega, \boldsymbol{l}) \propto \exp\left(\int \frac{\partial \ln\big(p(d \mid \omega, \boldsymbol{l})\big)}{\partial d} \,\mathrm{d}d \right). \tag{C.1}$$

For the case with an amplitude parameter $\alpha_\mathrm{h} = \frac{1}{2}$, the first-order derivative in Eq. (B.10) simplifies to

$$\frac{\partial \ln\big(p(d \mid \omega, \boldsymbol{l})\big)}{\partial d} = -\frac{d}{s_\mathrm{h}}. \tag{C.2}$$

Introducing this term in Eq. (C.1) and calculating the integral leads to the following expression for the posterior probability distribution

$$p(d \mid \omega, \boldsymbol{l}) \propto \exp\left(-\frac{d^2}{2s_\mathrm{h}} \right). \tag{C.3}$$

Because the posterior probability distribution has to be a valid PDF that integrates to one, there is only one possible solution. Knowing that, except for a constant scaling factor, the function looks like a Gaussian, the final solution can only be the Gaussian distribution itself

$$p(d \mid \omega, \boldsymbol{l}) = \frac{1}{\sqrt{2\pi s_\mathrm{h}}} \exp\left(-\frac{d^2}{2s_\mathrm{h}} \right). \tag{C.4}$$

For configurations with a slope parameter $s_\mathrm{h} \to 0$, the first-order derivative in Eq. (B.10) simplifies to

$$\frac{\partial \ln\big(p(d \mid \omega, \boldsymbol{l})\big)}{\partial d} = -2\tanh^{-1}(2\alpha_\mathrm{h})\,\mathrm{sgn}(d). \tag{C.5}$$

Introducing this term in Eq. (C.1) and calculating the integral leads to the following closed-form posterior probability distribution

$$p(d \mid \omega, \boldsymbol{l}) \propto \exp\big(-2\tanh^{-1}(2\alpha_\mathrm{h})|d| \big). \tag{C.6}$$

Similar to the previous derivation, we know that, with the exception of a constant scaling factor, the function is equal to a Laplace distribution, which again is a valid PDF. Introducing the scale parameter b, the final solution can therefore only be the following Laplace distribution

$$p(d \mid \omega, \boldsymbol{l}) = \frac{1}{2b} \exp\left(-\frac{|d|}{b} \right), \quad b = \frac{1}{2\tanh^{-1}(2\alpha_\mathrm{h})}. \tag{C.7}$$

D. Inverse-Variance Weighting

In the following, we demonstrate that the derivatives for the local optimization that were defined in Section 3.3.5 can be derived using inverse-variance weighting and a constant curvature of $1/\alpha_\mathrm{s}$ for the second-order derivative. Instead of the joint posterior probability defined in Eq. (3.32), we start with an energy function that combines probabilities from individual correspondence lines using inverse-variance weighting

$$E(\boldsymbol{\theta}) = \sum_{i=1}^{n_\mathrm{c}} \frac{1}{\sigma_i{}^2} \ln\big(p(d_{\mathrm{s}i}(\boldsymbol{\theta}) \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})\big). \tag{D.1}$$

Based on this function, the gradient vector and the Hessian matrix are calculated as the first- and second-order derivatives with respect to $\boldsymbol{\theta}$

$$\boldsymbol{g}^\top = \sum_{i=1}^{n_\mathrm{c}} \frac{1}{\sigma_i{}^2} \frac{\partial \ln\big(p(d_{\mathrm{s}i} \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})\big)}{\partial d_{\mathrm{s}i}} \frac{\partial d_{\mathrm{s}i}}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{0}}, \tag{D.2}$$

$$\boldsymbol{H} \approx \sum_{i=1}^{n_\mathrm{c}} \frac{1}{\sigma_i{}^2} \frac{\partial^2 \ln\big(p(d_{\mathrm{s}i} \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})\big)}{\partial d_{\mathrm{s}i}{}^2} \left(\frac{\partial d_{\mathrm{s}i}}{\partial \boldsymbol{\theta}}\right)^\top \left(\frac{\partial d_{\mathrm{s}i}}{\partial \boldsymbol{\theta}}\right)\bigg|_{\boldsymbol{\theta}=\boldsymbol{0}}. \tag{D.3}$$

For the first-order derivative of the scaled contour distance $d_{\mathrm{s}i}$, the derivations from Eqs. (3.39) and (3.40) can be used. In contrast to Eq. (3.43), we use the definition of finite differences without a weighting term to calculate the first-order derivative of the log-posterior

$$\frac{\partial \ln\big(p(d_{\mathrm{s}i} \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})\big)}{\partial d_{\mathrm{s}i}} \approx \ln\left(\frac{p(d_{\mathrm{s}i}^+ \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})}{p(d_{\mathrm{s}i}^- \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})}\right), \tag{D.4}$$

where $d_{\mathrm{s}i}^-$ and $d_{\mathrm{s}i}^+$ are again the two discrete contour distances that are closest to $d_{\mathrm{s}i}(\boldsymbol{\theta})$. Because the variance is already considered in the energy function, we simply define a constant curvature for the second-order derivative of the log-posterior

$$\frac{\partial^2 \ln\big(p(d_{\mathrm{s}i} \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})\big)}{\partial d_{\mathrm{s}i}{}^2} \approx \frac{1}{\alpha_\mathrm{s}}. \tag{D.5}$$

Knowing that constant scaling terms do not affect the Newton optimization, both the gradient vector and the Hessian matrix can be multiplied with the step size α_s. Together with the inverse variance $1/\sigma_i{}^2$ that is already present in Eqs. (D.2) and (D.3), this results in exactly the same expressions for the gradient vector and the Hessian matrix as defined in Section 3.3.5. In conclusion, the derivation therefore shows that weighting the first-order derivative in Eq. (3.43) with a factor $\alpha_\mathrm{s}/\sigma_i{}^2$ is the same as using inverse-variance weighting and a constant curvature of $1/\alpha_\mathrm{s}$ for the second-order derivative.

E. Derivatives of Constraint Equations

In the following, we derive the first-order derivatives of the extended constraint equation $\bar{\boldsymbol{b}}(\boldsymbol{\theta}_a, \boldsymbol{\theta}_b)$ with respect to $\boldsymbol{\theta}_a$ and $\boldsymbol{\theta}_b$. We thereby consider the rotational constraint

${}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}(\boldsymbol{\theta}_a, \boldsymbol{\theta}_b)$ and the translational constraint ${}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{B}}(\boldsymbol{\theta}_a, \boldsymbol{\theta}_b)$ separately. Constraint equations are introduced in Section 5.3.3.

E.1. Rotational Constraint

Starting from Eq. (5.18), the variated rotation between the coordinate frames A and B can be expressed using the exponential map as follows

$$\exp\left([{}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}(\boldsymbol{\theta}_{\mathrm{r}a}, \boldsymbol{\theta}_{\mathrm{r}b})]_\times\right) = \exp\left([-{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_a}\boldsymbol{\theta}_{\mathrm{r}a}]_\times\right)\exp\left([{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_b}\boldsymbol{\theta}_{\mathrm{r}b}]_\times\right)\exp\left([{}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}]_\times\right), \tag{E.1}$$

The vectors $\boldsymbol{\theta}_{\mathrm{r}a}$ and $\boldsymbol{\theta}_{\mathrm{r}b}$ are the rotational components of the pose variation vectors $\boldsymbol{\theta}_a$ and $\boldsymbol{\theta}_b$. The constant rotation vector ${}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}$ is evaluated at $\boldsymbol{\theta}_{\mathrm{r}a} = \boldsymbol{\theta}_{\mathrm{r}b} = \boldsymbol{0}$. Note that for the calculation of Eq. (E.1), the following relation was used

$$\boldsymbol{R}\exp\left([\boldsymbol{x}]_\times\right)\boldsymbol{R}^{-1} = \exp\left([\boldsymbol{R}\boldsymbol{x}]_\times\right). \tag{E.2}$$

Knowing that we only need to calculate the derivative with respect to either $\boldsymbol{\theta}_{\mathrm{r}a}$ or $\boldsymbol{\theta}_{\mathrm{r}b}$ while the other is zero, we reduce Eq. (E.1) to take a single variation vector $\boldsymbol{\theta}$ and write

$$\exp\left([{}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}(\boldsymbol{\theta})]_\times\right) = \exp\left([\boldsymbol{\theta}]_\times\right)\exp\left([{}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}]_\times\right). \tag{E.3}$$

For the vector $\boldsymbol{\theta}$, one can then simply substitute one of the two rotated variation vectors $\boldsymbol{\theta} = -{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_a}\boldsymbol{\theta}_{\mathrm{r}a}$ or $\boldsymbol{\theta} = {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_b}\boldsymbol{\theta}_{\mathrm{r}b}$.

While the relation in Eq. (E.3) looks relatively simple, it is not trivial to calculate the rotation vector ${}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}(\boldsymbol{\theta})$ from $\boldsymbol{\theta}$ and ${}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}$. Fortunately, a closed-form solution exists that dates back to Olinde Rodrigues. For this, each rotation vector has to be split into an axis and an angle. Using the angles $\gamma, \theta, \alpha \in \mathbb{R}$ and the axes $\boldsymbol{n}, \boldsymbol{v}, \boldsymbol{e} \in \mathbb{R}^3$ with $\|\boldsymbol{n}\|_2 = \|\boldsymbol{v}\|_2 = \|\boldsymbol{e}\|_2 = 1$, we define ${}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}(\boldsymbol{\theta}) = \gamma\boldsymbol{n}$, $\boldsymbol{\theta} = \theta\boldsymbol{v}$, and ${}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}} = \alpha\boldsymbol{e}$. Based on those values, one can then define the following equations

$$\cos\left(\frac{\gamma}{2}\right) = \cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\alpha}{2}\right) - \sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\alpha}{2}\right)\boldsymbol{v}^\top\boldsymbol{e}, \tag{E.4}$$

$$\sin\left(\frac{\gamma}{2}\right)\boldsymbol{n} = \sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\alpha}{2}\right)\boldsymbol{v} + \cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\alpha}{2}\right)\boldsymbol{e} + \sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\alpha}{2}\right)\boldsymbol{v}\times\boldsymbol{e}. \tag{E.5}$$

For a detailed derivation and geometrical explanation of this relation, we point interested readers to Altmann (2005). To express the required vector ${}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}(\boldsymbol{\theta})$, we define two variables $x = \cos(\frac{\gamma}{2})$ and $\boldsymbol{y} = \sin(\frac{\gamma}{2})\boldsymbol{n}$. They are the right-hand sides of Eqs. (E.4) and (E.5). Based on their reformulation $\gamma = 2\cos^{-1}(x)$ and $\boldsymbol{n} = \sin(\frac{\gamma}{2})^{-1}\boldsymbol{y}$, one can write

$${}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}(\boldsymbol{\theta}) = \gamma\boldsymbol{n} = \frac{2\cos^{-1}(x)}{\sin(\cos^{-1}(x))}\boldsymbol{y} = \frac{2\cos^{-1}(x)}{\sqrt{1-x^2}}\boldsymbol{y}. \tag{E.6}$$

Finally, starting from this definition, we calculate the first-order derivative with respect to the scalar parameter θ, which is the magnitude of the variation vector $\boldsymbol{\theta}$

$$\frac{\partial {}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}}{\partial\theta} = \frac{\frac{-2}{\sqrt{1-x^2}}\sqrt{1-x^2} - 2\cos^{-1}(x)\frac{-x}{\sqrt{1-x^2}}}{1-x^2}\frac{\partial x}{\partial\theta}\boldsymbol{y} + \frac{2\cos^{-1}(x)}{\sqrt{1-x^2}}\frac{\partial\boldsymbol{y}}{\partial\theta}. \tag{E.7}$$

Also, based on the right-hand sides of Eqs. (E.4) and (E.5), we calculate

$$\frac{\partial x}{\partial \theta} = -\frac{\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\alpha}{2}\right)}{2} - \frac{\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\alpha}{2}\right)}{2}\boldsymbol{v}^\top \boldsymbol{e}, \tag{E.8}$$

$$\frac{\partial \boldsymbol{y}}{\partial \theta} = \frac{\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\alpha}{2}\right)}{2}\boldsymbol{v} - \frac{\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\alpha}{2}\right)}{2}\boldsymbol{e} + \frac{\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\alpha}{2}\right)}{2}\boldsymbol{v}\times\boldsymbol{e}. \tag{E.9}$$

By evaluating the derivatives at $\theta = 0$, for which, according to Eqs. (E.4) and (E.5), $x = \cos(\frac{\alpha}{2})$ and $\boldsymbol{y} = \sin(\frac{\alpha}{2})\boldsymbol{e}$, the following equations are obtained

$$\left.\frac{\partial\, {}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}}{\partial \theta}\right|_{\theta=0} = \frac{-2+\alpha\frac{\cos(\frac{\alpha}{2})}{\sin(\frac{\alpha}{2})}}{\sin\left(\frac{\alpha}{2}\right)^2}\sin\left(\frac{\alpha}{2}\right)\boldsymbol{e}\left.\frac{\partial x}{\partial \theta}\right|_{\theta=0} + \frac{\alpha}{\sin\left(\frac{\alpha}{2}\right)}\left.\frac{\partial \boldsymbol{y}}{\partial \theta}\right|_{\theta=0} \tag{E.10}$$

$$= \frac{-2+\alpha\cot\left(\frac{\alpha}{2}\right)}{\sin\left(\frac{\alpha}{2}\right)}\boldsymbol{e}\left.\frac{\partial x}{\partial \theta}\right|_{\theta=0} + \frac{\alpha}{\sin\left(\frac{\alpha}{2}\right)}\left.\frac{\partial \boldsymbol{y}}{\partial \theta}\right|_{\theta=0}, \tag{E.11}$$

$$\left.\frac{\partial x}{\partial \theta}\right|_{\theta=0} = -\frac{1}{2}\sin\left(\frac{\alpha}{2}\right)\boldsymbol{v}^\top\boldsymbol{e}, \tag{E.12}$$

$$\left.\frac{\partial \boldsymbol{y}}{\partial \theta}\right|_{\theta=0} = \frac{1}{2}\cos\left(\frac{\alpha}{2}\right)\boldsymbol{v} + \frac{1}{2}\sin\left(\frac{\alpha}{2}\right)\boldsymbol{v}\times\boldsymbol{e}. \tag{E.13}$$

Introducing Eqs. (E.12) and (E.13) in Eq. (E.11) and extracting the pose variation's rotation axis $\boldsymbol{v}$ results in the final expression for the first-order derivative

$$\left.\frac{\partial\, {}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}}{\partial \theta}\right|_{\theta=0} = \left(1-\frac{\alpha}{2}\cot\left(\frac{\alpha}{2}\right)\right)\boldsymbol{e}\boldsymbol{v}^\top\boldsymbol{e} + \frac{\alpha}{2}\cot\left(\frac{\alpha}{2}\right)\boldsymbol{v} + \frac{\alpha}{2}\boldsymbol{v}\times\boldsymbol{e}, \tag{E.14}$$

$$= \left(\frac{\alpha}{2}\cot\left(\frac{\alpha}{2}\right)\boldsymbol{I} - \frac{\alpha}{2}[\boldsymbol{e}]_\times + \left(1-\frac{\alpha}{2}\cot\left(\frac{\alpha}{2}\right)\right)\boldsymbol{e}\boldsymbol{e}^\top\right)\boldsymbol{v}. \tag{E.15}$$

Equation (E.15) shows that the derivative with respect to the rotation angle θ consists of a linear combination of column vectors with the rotation axis $\boldsymbol{v}$. For the rotation vector $\boldsymbol{\theta} = \theta\boldsymbol{v}$, the rotation axis $\boldsymbol{v}$ projects the scalar angle θ in a similar fashion. Knowing that

$$\frac{\partial\, {}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}}{\partial \theta} = \frac{\partial\, {}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}}{\partial \boldsymbol{\theta}}\frac{\partial \boldsymbol{\theta}}{\partial \theta} = \frac{\partial\, {}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}}{\partial \boldsymbol{\theta}}\boldsymbol{v}, \tag{E.16}$$

we are finally able to extract the first-order derivative with respect to the rotation vector $\boldsymbol{\theta}$ from Eq. (E.15) and write

$$\left.\frac{\partial\, {}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{0}} = \frac{\alpha}{2}\cot\left(\frac{\alpha}{2}\right)\boldsymbol{I} - \frac{\alpha}{2}[\boldsymbol{e}]_\times + \left(1-\frac{\alpha}{2}\cot\left(\frac{\alpha}{2}\right)\right)\boldsymbol{e}\boldsymbol{e}^\top. \tag{E.17}$$

For the originally required derivatives of the rotational constraint ${}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}(\boldsymbol{\theta}_a,\boldsymbol{\theta}_b)$ with respect to the rotational pose variation vectors $\boldsymbol{\theta}_{\mathrm{r}a}$ and $\boldsymbol{\theta}_{\mathrm{r}b}$, one simply substitutes $\boldsymbol{\theta} = -{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_a}\boldsymbol{\theta}_{\mathrm{r}a}$ and $\boldsymbol{\theta} = {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_b}\boldsymbol{\theta}_{\mathrm{r}b}$. Considering the partial derivatives $\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\theta}_{\mathrm{r}a}} = -{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_a}$ and

$\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\theta}_{rb}} = {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_b}$, we then obtain the following final results for the first-order derivatives of the rotational constraint

$$\left.\frac{\partial {}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}}{\partial \boldsymbol{\theta}_a}\right|_{\boldsymbol{\theta}_{\mathrm{k}}=\mathbf{0}} = \begin{bmatrix} -\boldsymbol{C}\,{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_a} & \mathbf{0} \end{bmatrix}, \tag{E.18}$$

$$\left.\frac{\partial {}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}}{\partial \boldsymbol{\theta}_b}\right|_{\boldsymbol{\theta}_{\mathrm{k}}=\mathbf{0}} = \begin{bmatrix} \boldsymbol{C}\,{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_b} & \mathbf{0} \end{bmatrix}. \tag{E.19}$$

The matrix $\boldsymbol{C}$, which we call variation matrix, is thereby defined as

$$\boldsymbol{C} = \frac{\alpha}{2}\cot\left(\frac{\alpha}{2}\right)\boldsymbol{I} - \frac{\alpha}{2}[\boldsymbol{e}]_\times + \left(1 - \frac{\alpha}{2}\cot\left(\frac{\alpha}{2}\right)\right)\boldsymbol{e}\boldsymbol{e}^\top. \tag{E.20}$$

It is the partial derivative of $\left.\frac{\partial {}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\mathbf{0}}$ and is computed using the angle and axis of the current rotation vector ${}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}} = \alpha\boldsymbol{e}$. The variation matrix $\boldsymbol{C}$ describes how the rotation vector ${}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}$ changes for the variation with a subsequent infinitesimal rotation. Properties of the matrix are derived in Appendix F.

E.2. Translational Constraint

For the translational constraint, we again start from Eq. (5.18) to formulate the variated translation between the coordinate frames A and B. Using the knowledge that derivatives are either calculated with respect to $\boldsymbol{\theta}_{\mathrm{a}}$ or $\boldsymbol{\theta}_{\mathrm{b}}$, one can write

$${}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{B}}(\boldsymbol{\theta}_a) = {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_a}\big(\boldsymbol{R}(-\boldsymbol{\theta}_{\mathrm{r}a})({}_{\mathrm{M}_a}\boldsymbol{t}_{\mathrm{B}} - \boldsymbol{\theta}_{\mathrm{t}a})\big) + {}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{M}_a}, \tag{E.21}$$

$${}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{B}}(\boldsymbol{\theta}_b) = {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_b}\big(\boldsymbol{R}(\boldsymbol{\theta}_{\mathrm{r}b})\,{}_{\mathrm{M}_b}\boldsymbol{t}_{\mathrm{B}} + \boldsymbol{\theta}_{\mathrm{t}b}\big) + {}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{M}_b}, \tag{E.22}$$

where the vectors $\boldsymbol{\theta}_{\mathrm{t}a}$ and $\boldsymbol{\theta}_{\mathrm{t}b}$ are the translational components of the variation vectors $\boldsymbol{\theta}_a$ and $\boldsymbol{\theta}_b$.

Using the Taylor series expansion of the exponential map and the knowledge that $[\boldsymbol{x}]_\times\boldsymbol{y} = -[\boldsymbol{y}]_\times\boldsymbol{x}$, we define the following general relation

$$\begin{aligned} \boldsymbol{R}(\boldsymbol{\theta})\boldsymbol{t} &= \exp(\boldsymbol{\theta})\boldsymbol{t} \qquad &(E.23)\\ &= \big(\boldsymbol{I} + [\boldsymbol{\theta}]_\times + \mathcal{O}(\boldsymbol{\theta}^2)\big)\boldsymbol{t} \qquad &(E.24)\\ &= \boldsymbol{t} - [\boldsymbol{t}]_\times\boldsymbol{\theta} + \mathcal{O}(\boldsymbol{\theta}^2)\boldsymbol{t}, \qquad &(E.25) \end{aligned}$$

where the expression $\mathcal{O}(\boldsymbol{\theta}^2)$ models higher-order terms. Using this relation in Eqs. (E.21) and (E.22) and neglecting higher-order terms, it is straightforward to calculate the final results for the first-order derivatives of the translational constraint

$$\left.\frac{\partial {}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{B}}}{\partial \boldsymbol{\theta}_a}\right|_{\boldsymbol{\theta}_{\mathrm{k}}=\mathbf{0}} = \begin{bmatrix} {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_a}\,[{}_{\mathrm{M}_a}\boldsymbol{t}_{\mathrm{B}}]_\times & -{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_a} \end{bmatrix}, \tag{E.26}$$

$$\left.\frac{\partial {}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{B}}}{\partial \boldsymbol{\theta}_b}\right|_{\boldsymbol{\theta}_{\mathrm{k}}=\mathbf{0}} = \begin{bmatrix} -{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_b}\,[{}_{\mathrm{M}_b}\boldsymbol{t}_{\mathrm{B}}]_\times & {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_b} \end{bmatrix}. \tag{E.27}$$

F. Properties of the Variation Matrix

The variation matrix $\boldsymbol{C}$, which was derived in Appendix E.1, describes how the rotation vector ${}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}$ changes with the variation of a *subsequent* infinitesimal rotation. It is computed according to Eq. (E.20) and depends on the rotational difference between the coordinate frames A and B. In the following, we investigate its relation to the rotation matrix ${}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}}$ and the rotation vector ${}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}$. For brevity, we thereby drop the subscripts A and B in the notation and simply write $\boldsymbol{R}$ and $\boldsymbol{r}$.

Given the rotation vector $\boldsymbol{r} = \alpha\boldsymbol{e}$ with the angle α and the normalized axis $\boldsymbol{e}$, the rotation matrix can be computed using the Rodrigues formula

$$\boldsymbol{R} = \boldsymbol{I} + \sin(\alpha)[\boldsymbol{e}]_\times + 2\sin\left(\frac{\alpha}{2}\right)^2 [\boldsymbol{e}]_\times^2. \tag{F.1}$$

With the knowledge that $[\boldsymbol{e}]_\times\boldsymbol{e} = \boldsymbol{0}$, we are able to calculate the following product for the rotation and variation matrix

$$\begin{aligned}\boldsymbol{RC} = &\frac{\alpha}{2}\cot\left(\frac{\alpha}{2}\right)\boldsymbol{I} - \frac{\alpha}{2}[\boldsymbol{e}]_\times + \left(1 - \frac{\alpha}{2}\cot\left(\frac{\alpha}{2}\right)\right)\boldsymbol{e}\boldsymbol{e}^\top + \frac{\alpha}{2}\sin(\alpha)\cot\left(\frac{\alpha}{2}\right)[\boldsymbol{e}]_\times \\ &- \frac{\alpha}{2}\sin(\alpha)[\boldsymbol{e}]_\times^2 + \alpha\sin\left(\frac{\alpha}{2}\right)^2\cot\left(\frac{\alpha}{2}\right)[\boldsymbol{e}]_\times^2 - \alpha\sin\left(\frac{\alpha}{2}\right)^2[\boldsymbol{e}]_\times^3.\end{aligned} \tag{F.2}$$

Using the relations

$$\frac{\alpha}{2}\sin(\alpha)\cot\left(\frac{\alpha}{2}\right) = \alpha - \alpha\sin\left(\frac{\alpha}{2}\right)^2, \tag{F.3}$$

$$\alpha\sin\left(\frac{\alpha}{2}\right)^2\cot\left(\frac{\alpha}{2}\right) = \frac{\alpha}{2}\sin(\alpha), \tag{F.4}$$

$$[\boldsymbol{e}]_\times^3 = -[\boldsymbol{e}]_\times, \tag{F.5}$$

one can simplify Eq. (F.2) and write

$$\boldsymbol{RC} = \frac{\alpha}{2}\cot\left(\frac{\alpha}{2}\right)\boldsymbol{I} + \frac{\alpha}{2}[\boldsymbol{e}]_\times + \left(1 - \frac{\alpha}{2}\cot\left(\frac{\alpha}{2}\right)\right)\boldsymbol{e}\boldsymbol{e}^\top \tag{F.6}$$

$$= \boldsymbol{C}^\top, \tag{F.7}$$

which shows that the product of the rotation and variation matrix is simply the transposed variation matrix. Looking at Eqs. (E.4) and (E.5), one notices that changing the order of the rotations $\boldsymbol{r}$ and $\boldsymbol{\theta}$ only changes the sign in the cross product term in Eq. (E.5). Deriving the corresponding variation matrix then leads to the same expression as in Eq. (F.6), which is simply the transposed of the original variation matrix $\boldsymbol{C}$. The transposed variation matrix $\boldsymbol{C}^\top$ therefore describes how the rotation vector $\boldsymbol{r}$ changes with the variation of a *preceding* infinitesimal rotation. Based on this interpretation, Eq. (F.7) shows that rotating the variation matrix $\boldsymbol{C}$ with the rotation matrix $\boldsymbol{R}$ changes the order in which the infinitesimal rotation $\boldsymbol{\theta}$ is applied to the rotation $\boldsymbol{r}$.

Similarly, transposing the relation in Eq. (F.7) and using the knowledge that for valid rotation matrices $\boldsymbol{R}^\top = \boldsymbol{R}^{-1}$, one is able to write

$$\boldsymbol{RC} = \boldsymbol{C}^\top \;\Leftrightarrow\; \boldsymbol{C}^\top\boldsymbol{R}^\top = \boldsymbol{C} \;\Leftrightarrow\; \boldsymbol{C}^\top = \boldsymbol{CR}. \tag{F.8}$$

This shows that $\boldsymbol{R}$ and $\boldsymbol{C}$ are commutative. Hence, applying the rotation matrix $\boldsymbol{R}$ on either side of the variation matrix $\boldsymbol{C}$ changes the location of the infinitesimal rotation from a *subsequent* to a *preceding* rotation. Finally, based on the relations in Eq. (F.8), we are able to express the rotation matrix with respect to the variation matrix as follows

$$\boldsymbol{R} = \boldsymbol{C}^\top\boldsymbol{C}^{-1} = \boldsymbol{C}^{-1}\boldsymbol{C}^\top. \tag{F.9}$$

The equation shows that because $\boldsymbol{C}$ and $\boldsymbol{C}^\top$ consider the variation with respect to the frames before and after the rotation, the matrix $\boldsymbol{R}$ is implicitly encoded in the variation matrix $\boldsymbol{C}$. Also, Eq. (F.9) shows that the variation matrix is normal, with $\boldsymbol{CC}^\top = \boldsymbol{C}^\top\boldsymbol{C}$. However, at the same time, experiments show that, with $\boldsymbol{C}^\top \neq \boldsymbol{C}$ and $\boldsymbol{C}^\top \neq \boldsymbol{C}^{-1}$, it is neither symmetric nor orthogonal.

Finally, we also want to analyze how the rotation vector $\boldsymbol{r}$ is connected to the variation matrix $\boldsymbol{C}$. For this, the product $\boldsymbol{Cr}$ is computed. With the relations $[\boldsymbol{e}]_\times\boldsymbol{e} = \boldsymbol{0}$ and $\boldsymbol{e}^\top\boldsymbol{e} = 1$, we calculate

$$\boldsymbol{Cr} = \left(\frac{\alpha}{2}\cot\left(\frac{\alpha}{2}\right)\boldsymbol{I} - \frac{\alpha}{2}[\boldsymbol{e}]_\times + \left(1 - \frac{\alpha}{2}\cot\left(\frac{\alpha}{2}\right)\right)\boldsymbol{ee}^\top\right)\alpha\boldsymbol{e} \tag{F.10}$$

$$= \left(\frac{\alpha}{2}\cot\left(\frac{\alpha}{2}\right) + 1 - \frac{\alpha}{2}\cot\left(\frac{\alpha}{2}\right)\right)\alpha\boldsymbol{e}, \tag{F.11}$$

$$= \boldsymbol{r}. \tag{F.12}$$

Given the knowledge that the rotation vector $\boldsymbol{r}$ is the eigenvector of the rotation matrix $\boldsymbol{R}$ for the eigenvalue $\lambda = 1$, one is able to define the relation $\boldsymbol{r} = \boldsymbol{Rr}$. Together with the results from Eq. (F.7), we can therefore show that

$$\boldsymbol{Cr} = \boldsymbol{r} \;\Leftrightarrow\; \boldsymbol{CRr} = \boldsymbol{r} \;\Leftrightarrow\; \boldsymbol{C}^\top\boldsymbol{r} = \boldsymbol{r}. \tag{F.13}$$

This demonstrates that the rotation vector $\boldsymbol{r}$ is an eigenvector of both the original and transposed variation matrix $\boldsymbol{C}$. Hence, it is an eigenvector of both the variation matrix for a *subsequent* and *preceding* infinitesimal rotation.

G. Constraint Convergence

Given the developed constraint equations and first-order derivatives, in the following, we analyze how fast the Newton method converges to a compliant kinematic result. For this, two bodies with an initial rotational or translational difference of ${}_\mathrm{A}\boldsymbol{R}_\mathrm{B}$ or ${}_\mathrm{A}\boldsymbol{t}_\mathrm{B}$ are considered. Based on this initial error, the relative pose update for a full rotational or translational constraint is analyzed. Experiments, which demonstrate that the obtained mathematical results also hold for more general cases, are provided in Section 5.5.4.

G.1. Rotational Constraint

Without loss of generality, we assume that constraint coordinate frames are equal to model frames with $\mathrm{A} = \mathrm{M}_a$ and $\mathrm{B} = \mathrm{M}_b$. Also, the translation of both bodies is fixed and the combined variation $\boldsymbol{\theta}_\mathrm{k}^\top = \begin{bmatrix}\boldsymbol{\theta}_{\mathrm{r}a}^\top & \boldsymbol{\theta}_{\mathrm{r}b}^\top\end{bmatrix}$ only considers rotation. Based on those assumptions, the constraint equation and constraint Jacobian can be written as

$$\boldsymbol{b}_\mathrm{k} = {}_\mathrm{A}\boldsymbol{r}_\mathrm{B}, \tag{G.1}$$

$$\boldsymbol{B}_\mathrm{k} = \begin{bmatrix}-\boldsymbol{C} & \boldsymbol{C}\,{}_\mathrm{A}\boldsymbol{R}_\mathrm{B}\end{bmatrix}. \tag{G.2}$$

Introducing both statements in Eq. (5.8) and given some arbitrary Hessian matrices and gradient vectors for body a and b, we obtain the following linear equation for a single Newton step

$$\begin{bmatrix}-\boldsymbol{H}_a & \boldsymbol{0} & \boldsymbol{C}^\top \\ \boldsymbol{0} & -\boldsymbol{H}_b & -{}_\mathrm{A}\boldsymbol{R}_\mathrm{B}^\top \boldsymbol{C}^\top \\ \boldsymbol{C} & -\boldsymbol{C}\,{}_\mathrm{A}\boldsymbol{R}_\mathrm{B} & \boldsymbol{0}\end{bmatrix} \begin{bmatrix}\boldsymbol{\theta}_{\mathrm{r}a} \\ \boldsymbol{\theta}_{\mathrm{r}b} \\ \boldsymbol{\lambda}\end{bmatrix} = \begin{bmatrix}\boldsymbol{g}_a \\ \boldsymbol{g}_b \\ {}_\mathrm{A}\boldsymbol{r}_\mathrm{B}\end{bmatrix}. \tag{G.3}$$

Based on the first two lines of the equation, one can directly write

$$\boldsymbol{\theta}_{\mathrm{r}a} = \boldsymbol{H}_a^{-1}(\boldsymbol{C}^\top\boldsymbol{\lambda} - \boldsymbol{g}_a), \tag{G.4}$$

$$\boldsymbol{\theta}_{\mathrm{r}b} = \boldsymbol{H}_b^{-1}(-{}_\mathrm{A}\boldsymbol{R}_\mathrm{B}^\top \boldsymbol{C}^\top\boldsymbol{\lambda} - \boldsymbol{g}_b), \tag{G.5}$$

where the only unknown is $\boldsymbol{\lambda}$. Subsequently, introducing Eqs. (G.4) and (G.5) in the last line of Eq. (G.3) leads to the following expression

$$\boldsymbol{C}(\boldsymbol{X}+\boldsymbol{Y})\boldsymbol{C}^\top\boldsymbol{\lambda} + \boldsymbol{C}(-\boldsymbol{x}+\boldsymbol{y}) = {}_\mathrm{A}\boldsymbol{r}_\mathrm{B}, \tag{G.6}$$

where the matrices $\boldsymbol{X} = \boldsymbol{H}_a^{-1}$ and $\boldsymbol{Y} = {}_\mathrm{A}\boldsymbol{R}_\mathrm{B}\,\boldsymbol{H}_b^{-1}\,{}_\mathrm{A}\boldsymbol{R}_\mathrm{B}^\top$, as well as the vectors $\boldsymbol{x} = \boldsymbol{H}_a^{-1}\boldsymbol{g}_a$ and $\boldsymbol{y} = {}_\mathrm{A}\boldsymbol{R}_\mathrm{B}\,\boldsymbol{H}_b^{-1}\boldsymbol{g}_b$ are used for conciseness. Rewriting this equation leads to an expression for $\boldsymbol{\lambda}$ that only depends on gradient vectors, Hessian matrices, and the current rotational difference

$$\boldsymbol{\lambda} = \boldsymbol{C}^{-\top}(\boldsymbol{X}+\boldsymbol{Y})^{-1}(\boldsymbol{x}-\boldsymbol{y}+\boldsymbol{C}^{-1}\,{}_\mathrm{A}\boldsymbol{r}_\mathrm{B}). \tag{G.7}$$

Finally, using the proof from Appendix F, which shows that ${}_\mathrm{A}\boldsymbol{r}_\mathrm{B}$ is an eigenvector of the variation matrix $\boldsymbol{C}$ and that $\boldsymbol{C}^{-1}\,{}_\mathrm{A}\boldsymbol{r}_\mathrm{B} = {}_\mathrm{A}\boldsymbol{r}_\mathrm{B}$, we can further simplify and write

$$\boldsymbol{\lambda} = \boldsymbol{C}^{-\top}(\boldsymbol{X}+\boldsymbol{Y})^{-1}(\boldsymbol{x}-\boldsymbol{y}+{}_\mathrm{A}\boldsymbol{r}_\mathrm{B}). \tag{G.8}$$

To assess the convergence of our method, we compute how the rotational difference changes for a single Newton step. Using the axis-angle representation, the change in rotation is thereby calculated as

$$\Delta\boldsymbol{R} = \exp\left([-\boldsymbol{\theta}_{\mathrm{r}a}]_\times\right)\exp\left([{}_\mathrm{A}\boldsymbol{R}_\mathrm{B}\,\boldsymbol{\theta}_{\mathrm{r}b}]_\times\right). \tag{G.9}$$

By introducing Eqs. (G.4) and (G.5), one can then write

$$\Delta\boldsymbol{R} = \exp\left([-\boldsymbol{X}\boldsymbol{C}^\top\boldsymbol{\lambda}+\boldsymbol{x}]_\times\right)\exp\left([-\boldsymbol{Y}\boldsymbol{C}^\top\boldsymbol{\lambda}-\boldsymbol{y}]_\times\right). \tag{G.10}$$

Extending this equation with $\boldsymbol{Y} - \boldsymbol{Y}$ and $\boldsymbol{X} - \boldsymbol{X}$ leads to

$$\begin{aligned}\Delta\boldsymbol{R} = \exp&\left(\frac{1}{2}\left[(\boldsymbol{Y} - \boldsymbol{X} - \boldsymbol{X} - \boldsymbol{Y})\boldsymbol{C}^\top\boldsymbol{\lambda} + 2\boldsymbol{x}\right]_\times\right)\\ \exp&\left(\frac{1}{2}\left[(\boldsymbol{X} - \boldsymbol{Y} - \boldsymbol{X} - \boldsymbol{Y})\boldsymbol{C}^\top\boldsymbol{\lambda} - 2\boldsymbol{y}\right]_\times\right).\end{aligned} \tag{G.11}$$

Subsequently, we introduce $\boldsymbol{\lambda}$ from Eq. (G.8) and multiply it with the expression $(-\boldsymbol{X} - \boldsymbol{Y})\boldsymbol{C}^\top$ in Eq. (G.11) to obtain

$$\begin{aligned}\Delta\boldsymbol{R} = \exp&\left(\frac{1}{2}\left[(\boldsymbol{Y} - \boldsymbol{X})\boldsymbol{C}^\top\boldsymbol{\lambda} + \boldsymbol{x} + \boldsymbol{y} - {}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}\right]_\times\right)\\ \exp&\left(\frac{1}{2}\left[(\boldsymbol{X} - \boldsymbol{Y})\boldsymbol{C}^\top\boldsymbol{\lambda} - \boldsymbol{x} - \boldsymbol{y} - {}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}\right]_\times\right),\end{aligned} \tag{G.12}$$

which has the following highly symmetric structure

$$\Delta\boldsymbol{R} = \exp\left(\frac{1}{2}\left[\boldsymbol{z} - {}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}\right]_\times\right)\exp\left(\frac{1}{2}\left[-\boldsymbol{z} - {}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}\right]_\times\right). \tag{G.13}$$

Finally, observing that the two exponents are commutative and knowing that commutative exponents can be summed, we get the following result for the rotational difference

$$\Delta\boldsymbol{R} = \exp\left([-{}_{\mathrm{A}}\boldsymbol{r}_{\mathrm{B}}]_\times\right) = {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}}^{-1} \tag{G.14}$$

This shows that a single iteration of the Newton method is enough to minimize an initial rotational error of ${}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}}$ to zero and obtain an exact kinematic solution.

G.2. Translational Constraint

Without loss of generality, we again assume that constraint coordinate frames are equal to model frames with $\mathrm{A} = \mathrm{M}_a$ and $\mathrm{B} = \mathrm{M}_b$. Furthermore, the rotation of both bodies is fixed and the combined variation vector $\boldsymbol{\theta}_{\mathrm{k}}^\top = \begin{bmatrix}\boldsymbol{\theta}_{\mathrm{t}a}^\top & \boldsymbol{\theta}_{\mathrm{t}b}^\top\end{bmatrix}$ only considers translation. The constraint equation and constraint Jacobian can then be written as

$$\boldsymbol{b}_{\mathrm{k}} = {}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{B}}, \tag{G.15}$$

$$\boldsymbol{B}_{\mathrm{k}} = \begin{bmatrix}-\boldsymbol{I} & {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}}\end{bmatrix}. \tag{G.16}$$

Introducing both statements in Eq. (5.8), we obtain the following linear equation for a single Newton step

$$\begin{bmatrix}-\boldsymbol{H}_a & \boldsymbol{0} & \boldsymbol{I}^\top\\ \boldsymbol{0} & -\boldsymbol{H}_b & -{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}}^\top\\ \boldsymbol{I} & -{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}} & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{\theta}_{\mathrm{t}a}\\ \boldsymbol{\theta}_{\mathrm{t}b}\\ \boldsymbol{\lambda}\end{bmatrix} = \begin{bmatrix}\boldsymbol{g}_a\\ \boldsymbol{g}_b\\ {}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{B}}\end{bmatrix}. \tag{G.17}$$

Based on the first two rows of this relation, variation vectors are computed as

$$\boldsymbol{\theta}_{\mathrm{t}a} = \boldsymbol{H}_a^{-1}(\boldsymbol{\lambda} - \boldsymbol{g}_a), \tag{G.18}$$

$$\boldsymbol{\theta}_{\mathrm{t}b} = \boldsymbol{H}_b^{-1}(-{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}}^{\top}\,\boldsymbol{\lambda} - \boldsymbol{g}_b). \tag{G.19}$$

For conciseness, we again define the matrices $\boldsymbol{X} = \boldsymbol{H}_a^{-1}$ and $\boldsymbol{Y} = {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}}\,\boldsymbol{H}_b^{-1}{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}}^{\top}$, as well as the vectors $\boldsymbol{x} = \boldsymbol{H}_a^{-1}\boldsymbol{g}_a$ and $\boldsymbol{y} = {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}}\,\boldsymbol{H}_b^{-1}\boldsymbol{g}_b$. We then introduce Eqs. (G.18) and (G.19) into the last row of Eq. (G.17) to compute the following relation for $\boldsymbol{\lambda}$ that includes only known variables

$$\boldsymbol{\lambda} = (\boldsymbol{X} + \boldsymbol{Y})^{-1}(\boldsymbol{x} - \boldsymbol{y} + {}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{B}}). \tag{G.20}$$

To assess the convergence, we consider the change of the translational difference between frame A and B and write

$$\Delta\boldsymbol{t} = -\boldsymbol{\theta}_{\mathrm{t}a} + {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}}\,\boldsymbol{\theta}_{\mathrm{t}b}. \tag{G.21}$$

Introducing the variation vectors from Eqs. (G.18) and (G.19) then leads to the relation

$$\Delta\boldsymbol{t} = -(\boldsymbol{X} + \boldsymbol{Y})\boldsymbol{\lambda} + \boldsymbol{x} - \boldsymbol{y}. \tag{G.22}$$

Finally, with the definition of $\boldsymbol{\lambda}$ from Eq. (G.20), we are able to write the following result

$$\Delta\boldsymbol{t} = -{}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{B}}. \tag{G.23}$$

Similar to the rotational case, this demonstrates that a single iteration of the Newton method is enough to overcome the translational error ${}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{B}}$ and obtain an exact kinematic solution.

H. Adjoint Equivalence

In the following, we demonstrate that if the constraint coordinate frames A and B are equal, the first-order derivatives of the constraint equation in Eqs. (5.22) and (5.23). reduce to an adjoint representation. Given that in such a case the variation matrix $\boldsymbol{C}$ is equal to the identity matrix, only the partial derivatives of the translational difference ${}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{B}}$ with respect to the rotational variation vectors $\boldsymbol{\theta}_{\mathrm{r}a}$ and $\boldsymbol{\theta}_{\mathrm{r}b}$ differ from the adjoint representation defined in Eq. (5.11). However, we can show that

$$\begin{aligned}
{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_a}\,[{}_{\mathrm{M}_a}\boldsymbol{t}_{\mathrm{B}}]_\times &= {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_a}\,[{}_{\mathrm{M}_a}\boldsymbol{t}_{\mathrm{A}}]_\times && \text{(H.1)}\\
&= {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_a}\,[{}_{\mathrm{M}_a}\boldsymbol{t}_{\mathrm{A}}]_\times\,{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_a}^{-1}\,{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_a} && \text{(H.2)}\\
&= [{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_a}\,{}_{\mathrm{M}_a}\boldsymbol{t}_{\mathrm{A}}]_\times\,{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_a} && \text{(H.3)}\\
&= -[{}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{M}_a}]_\times\,{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_a}, && \text{(H.4)}
\end{aligned}$$

and similarly that

$$-{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_b}\,[{}_{\mathrm{M}_b}\boldsymbol{t}_{\mathrm{B}}]_\times = [{}_{\mathrm{A}}\boldsymbol{t}_{\mathrm{M}_b}]_\times\,{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_b}. \tag{H.5}$$

This demonstrates that if A and B are equal, the derivatives in Eqs. (5.22) and (5.23) turn into $-\operatorname{Ad}({}_{\mathrm{A}}\boldsymbol{T}_{\mathrm{M}_a})$ and $\operatorname{Ad}({}_{\mathrm{A}}\boldsymbol{T}_{\mathrm{M}_b})$.

I. Derivatives of Orthogonality Constraints

In the following, we calculate the first-order derivatives of the rotational orthogonality constraint equation. They are required for the calculation of the constraint Jacobian in Eq. (5.21). Starting from Eq. (5.32) the constraint equation is defined as

$$b_{\mathrm{r}ijab}(\boldsymbol{\theta}_a, \boldsymbol{\theta}_b) = {}_{\mathrm{A}}\boldsymbol{e}_i^\top \exp\left([-{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_a}\,\boldsymbol{\theta}_{\mathrm{r}a}]_\times\right) \exp\left([{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_b}\,\boldsymbol{\theta}_{\mathrm{r}b}]_\times\right) {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}}\,{}_{\mathrm{B}}\boldsymbol{e}_j, \tag{I.1}$$

with the tuple $(i, j) \in \{(\mathrm{x}, \mathrm{y}), (\mathrm{y}, \mathrm{z}), (\mathrm{z}, \mathrm{x})\}$. Similar to Eqs. (E.24) and (E.25), we first linearize using the Taylor series expansion $\exp([\boldsymbol{\theta}]_\times) \approx \boldsymbol{I} + [\boldsymbol{\theta}]_\times$ and then use $[\boldsymbol{x}]_\times \boldsymbol{y} = -[\boldsymbol{y}]_\times \boldsymbol{x}$. With this, the following first-order derivatives can be calculated

$$\left.\frac{\partial b_{\mathrm{r}ijab}}{\partial \boldsymbol{\theta}_{\mathrm{r}a}}\right|_{\boldsymbol{\theta}_\mathrm{k}=\boldsymbol{0}} = {}_{\mathrm{A}}\boldsymbol{e}_i^\top [{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}}\,{}_{\mathrm{B}}\boldsymbol{e}_j]_\times\, {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_a}, \tag{I.2}$$

$$\left.\frac{\partial b_{\mathrm{r}ijab}}{\partial \boldsymbol{\theta}_{\mathrm{r}b}}\right|_{\boldsymbol{\theta}_\mathrm{k}=\boldsymbol{0}} = {}_{\mathrm{A}}\boldsymbol{e}_i^\top [{}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}}\,{}_{\mathrm{B}}\boldsymbol{e}_j]_\times\, {}_{\mathrm{A}}\boldsymbol{R}_{\mathrm{M}_b}. \tag{I.3}$$

Knowing that first-order derivatives with respect to the translational variations $\boldsymbol{\theta}_{\mathrm{t}a}$ and $\boldsymbol{\theta}_{\mathrm{t}b}$ are zero, we are able to compute the constraint Jacobians $\boldsymbol{B}_{ab}$ required in the Newton optimization.

Bibliography

Alahi, A., R. Ortiz, and P. Vandergheynst (2012). "FREAK: Fast Retina Keypoint." In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 510–517.

Alcantarilla, P. F., A. Bartoli, and A. J. Davison (2012). "KAZE Features." In: *European Conference on Computer Vision*, pp. 214–227.

Altmann, S. L. (2005). *Rotations, Quaternions, and Double Groups*. Courier Corporation.

Armstrong, M. and A. Zisserman (1995). "Robust Object Tracking." In: *Asian Conference on Computer Vision*, pp. 58–61.

Baker, S. and I. Matthews (2004). "Lucas-Kanade 20 Years On: A Unifying Framework." In: *International Journal of Computer Vision* 56.3, pp. 221–255.

Bay, H., T. Tuytelaars, and L. Van Gool (2006). "SURF: Speeded Up Robust Features." In: *European Conference on Computer Vision*, pp. 404–417.

Besl, P. J. and N. D. McKay (1992). "A Method for Registration of 3-D Shapes." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2, pp. 239–256.

Bibby, C. and I. D. Reid (2008). "Robust Real-Time Visual Tracking Using Pixel-Wise Posteriors." In: *European Conference on Computer Vision*, pp. 831–844.

Blais, G. and M. D. Levine (1995). "Registering Multiview Range Data to Create 3D Computer Objects." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17.8, pp. 820–824.

Bohg, J., J. Romero, A. Herzog, and S. Schaal (2014). "Robot Arm Pose Estimation through Pixel-Wise Part Classification." In: *IEEE International Conference on Robotics and Automation*, pp. 3143–3150.

Brachmann, E., A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother (2017). "DSAC - Differentiable RANSAC for Camera Localization." In: *IEEE Conference on Computer Vision and Pattern Recognition*.

Brachmann, E., F. Michel, A. Krull, M. Y. Yang, S. Gumhold, and C. Rother (2016). "Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image." In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3364–3372.

Bregler, C. and J. Malik (1998). "Tracking People with Twists and Exponential Maps." In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 8–15.

Brown, J. A. and D. W. Capson (2012). "A Framework for 3D Model-Based Visual Tracking Using a GPU-Accelerated Particle Filter." In: *IEEE Transactions on Visualization and Computer Graphics* 18.1, pp. 68–80.

Brox, T., B. Rosenhahn, J. Gall, and D. Cremers (2010). "Combined Region and Motion-Based 3D Tracking of Rigid and Articulated Objects." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.3, pp. 402–415.

Bugaev, B., A. Kryshchenko, and R. Belov (2018). "Combining 3D Model Contour Energy and Keypoints for Object Tracking." In: *European Conference on Computer Vision*, pp. 55–70.

Calli, B., A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar (2015). "The YCB Object and Model Set: Towards Common Benchmarks for Manipulation Research." In: *International Conference on Advanced Robotics*, pp. 510–517.

Calonder, M., V. Lepetit, C. Strecha, and P. Fua (2010). "BRIEF: Binary Robust Independent Elementary Features." In: *European Conference on Computer Vision*, pp. 778–792.

Campos, C., R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós (2021). "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM." In: *IEEE Transactions on Robotics* 37.6, pp. 1874–1890.

Campos, T. E. de, B. J. Tordoff, and D. W. Murray (2006). "Recovering Articulated Pose: A Comparison of Two Pre and Postimposed Constraint Methods." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.1, pp. 163–168.

Caron, G., A. Dame, and E. Marchand (2014). "Direct Model Based Visual Tracking and Pose Estimation Using Mutual Information." In: *Image and Vision Computing* 32.1, pp. 54–63.

Chen, Y. and G. Medioni (1992). "Object Modelling by Registration of Multiple Range Images." In: *Image and Vision Computing* 10.3, pp. 145–155.

Choi, C. and H. I. Christensen (2012a). "Robust 3D Visual Tracking Using Particle Filtering on the Special Euclidean Group: A Combined Approach of Keypoint and Edge Features." In: *The International Journal of Robotics Research* 31.4, pp. 498–519.

Choi, C. and H. I. Christensen (2012b). "3D Textureless Object Detection and Tracking: An Edge-based Approach." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3877–3884.

Choi, C. and H. I. Christensen (2013). "RGB-D Object Tracking: A Particle Filter Approach on GPU." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1084–1091.

Cifuentes, C. G., J. Issac, M. Wüthrich, S. Schaal, and J. Bohg (2017). "Probabilistic Articulated Real-Time Tracking for Robot Manipulation." In: *IEEE Robotics and Automation Letters* 2.2, pp. 577–584.

Comport, A. I., E. Marchand, M. Pressigout, and F. Chaumette (2006). "Real-Time Markerless Tracking for Augmented Reality: The Virtual Visual Servoing Framework." In: *IEEE Transactions on Visualization and Computer Graphics* 12.4, pp. 615–628.

Comport, A. I., É. Marchand, and F. Chaumette (2007). "Kinematic Sets for Real-time Robust Articulated Object Tracking." In: *Image and Vision Computing* 25.3, pp. 374–391.

Crivellaro, A. and V. Lepetit (2014). "Robust 3D Tracking with Descriptor Fields." In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3414–3421.

Dambreville, S., R. Sandhu, A. Yezzi, and A. Tannenbaum (2008). "Robust 3D Pose Estimation and Efficient 2D Region-Based Segmentation from a 3D Shape Prior." In: *European Conference on Computer Vision*, pp. 169–182.

Davison, A. J. (2003). "Real-Time Simultaneous Localisation and Mapping with a Single Camera." In: *Proceedings Ninth IEEE International Conference on Computer Vision*, 1403–1410 vol.2.

Davison, A. J., I. D. Reid, N. D. Molton, and O. Stasse (2007). "MonoSLAM: Real-Time Single Camera SLAM." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6, pp. 1052–1067.

Demirdjian, D., T. Ko, and T. Darrell (2003). "Constraining Human Body Tracking." In: *Ninth IEEE International Conference on Computer Vision*, 1071–1078 vol.2.

Deng, X., J. Geng, T. Bretl, Y. Xiang, and D. Fox (2022). "iCaps: Iterative Category-Level Object Pose and Shape Estimation." In: *IEEE Robotics and Automation Letters* 7.2, pp. 1784–1791.

Deng, X., A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox (2021). "PoseRBPF: A Rao–Blackwellized Particle Filter for 6-D Object Pose Tracking." In: *IEEE Transactions on Robotics* 37.5, pp. 1328–1342.

DeTone, D., T. Malisiewicz, and A. Rabinovich (2018). "SuperPoint: Self-Supervised Interest Point Detection and Description." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 337–349.

Deutschmann, B., J. Reinecke, and A. Dietrich (2022). "Open Source Tendon-driven Continuum Mechanism: A Platform for Research in Soft Robotics." In: *IEEE 5th International Conference on Soft Robotics*, pp. 54–61.

Dewaele, G., F. Devernay, and R. Horaud (2004). "Hand Motion from 3D Point Trajectories and a Smooth Surface Model." In: *European Conference on Computer Vision*. Ed. by T. Pajdla and J. Matas, pp. 495–507.

Diebel, J., K. Reutersward, S. Thrun, J. Davis, and R. Gupta (2004). "Simultaneous Localization and Mapping with Active Stereo Vision." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 4, pp. 3436–3443.

Dorai, C., G. Wang, A. Jain, and C. Mercer (1998). "Registration and Integration of Multiple Object Views for 3D Model Construction." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.1, pp. 83–89.

Drummond, T. and R. Cipolla (2002). "Real-Time Visual Tracking of Complex Structures." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.7, pp. 932–946.

Dusmanu, M., I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler (2019). "D2-Net: A Trainable CNN for Joint Description and Detection of Local Features." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8092–8101.

Elsayed, M. (2021). "Real-Time Texture-based 3D Object Tracking for Advanced Robotic Manipulation." MA thesis. Technical University of Munich.

Engel, J., V. Koltun, and D. Cremers (2018). "Direct Sparse Odometry." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.3, pp. 611–625.

Fischler, M. A. and R. C. Bolles (1981). "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography." In: *Communications of the ACM* 24.6, pp. 381–395.

Fitzgibbon, A. W. (2003). "Robust Registration of 2D and 3D Point Sets." In: *Image and Vision Computing* 21.13-14, pp. 1145–1153.

Forster, C., M. Pizzoli, and D. Scaramuzza (2014). "SVO: Fast Semi-Direct Monocular Visual Odometry." In: *IEEE International Conference on Robotics and Automation*, pp. 15–22.

Garon, M. and J.-F. Lalonde (2017). "Deep 6-DOF Tracking." In: *IEEE Transactions on Visualization and Computer Graphics* 23.11, pp. 2410–2418.

Gavrila, D. and L. Davis (1996). "3-D Model-based Tracking of Humans in Action: A Multi-view Approach." In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 73–80.

Ge, L., Z. Ren, Y. Li, Z. Xue, Y. Wang, J. Cai, and J. Yuan (2019). "3D Hand Shape and Pose Estimation From a Single RGB Image." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10825–10834.

Ge, R. and G. Loianno (2021). "VIPose: Real-time Visual-Inertial 6D Object Pose Tracking." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4597–4603.

Gennery, D. B. (1992). "Visual Tracking of Known Three-Dimensional Objects." In: *International Journal of Computer Vision* 7.3, pp. 243–270.

Godin, G., M. Rioux, and R. Baribeau (1994). "Three-dimensional Registration Using Range and Intensity Information." In: *Videometrics III*. Vol. 2350, pp. 279–290.

Grebenstein, M., A. Albu-Schäffer, T. Bahls, M. Chalon, O. Eiberger, W. Friedl, R. Gruber, S. Haddadin, U. Hagn, R. Haslinger, H. Höppner, S. Jörg, M. Nickl, A. Nothhelfer, F. Petit, J. Reill, N. Seitz, T. Wimböck, S. Wolf, T. Wüsthoff, and G. Hirzinger (2011). "The DLR Hand Arm System." In: *IEEE International Conference on Robotics and Automation*, pp. 3175–3182.

Hagn, U., R. Konietschke, A. Tobergte, M. Nickl, S. Jörg, B. Kübler, G. Passig, M. Gröger, F. Fröhlich, U. Seibold, L. Le-Tien, A. Albu-Schäffer, A. Nothhelfer, F. Hacker, M. Grebenstein, and G. Hirzinger (2010). "DLR MiroSurge: A Versatile System for Research in Endoscopic Telesurgery." In: *International Journal of Computer Assisted Radiology and Surgery* 5.2, pp. 183–193.

Hagn, U., M. Nickl, S. Jörg, G. Passig, T. Bahls, A. Nothhelfer, F. Hacker, L. Le-Tien, A. Albu-Schäffer, R. Konietschke, et al. (2008). "The DLR MIRO: A Versatile Lightweight Robot for Surgical Applications." In: *Industrial Robot: An International Journal* 35.4, pp. 324–336.

Han, S., B. Liu, R. Cabezas, C. D. Twigg, P. Zhang, J. Petkau, T.-H. Yu, C.-J. Tai, M. Akbay, Z. Wang, A. Nitzan, G. Dong, Y. Ye, L. Tao, C. Wan, and R. Wang (2020). "MEgATrack: Monochrome Egocentric Articulated Hand-Tracking for Virtual Reality." In: *ACM Transactions on Graphics* 39.4.

Harris, C. and C. Stennett (1990). "RAPID - A Video Rate Object Tracker." In: *British Machine Vision Conference*, pp. 15.1–15.6.

Harris, C. and M. Stephens (1988). "A Combined Corner and Edge Detector." In: *Proceedings of the Alvey Vision Conference*, pp. 23.1–23.6.

Haugaard, R. L. and A. G. Buch (2022). "SurfEmb: Dense and Continuous Correspondence Distributions for Object Pose Estimation With Learnt Surface Embeddings." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6749–6758.

He, Y., H. Huang, H. Fan, Q. Chen, and J. Sun (2021). "FFB6D: A Full Flow Bidirectional Fusion Network for 6D Pose Estimation." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3003–3013.

He, Y., W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun (2020). "PVN3D: A Deep Point-Wise 3D Keypoints Voting Network for 6DoF Pose Estimation." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11629–11638.

Hebert, P., N. Hudson, J. Ma, T. Howard, T. Fuchs, M. Bajracharya, and J. Burdick (2012). "Combined Shape, Appearance and Silhouette for Simultaneous Manipulator and Object Tracking." In: *International Conference on Robotics and Automation*, pp. 2405–2412.

Hexner, J. and R. R. Hagege (2016). "2D-3D Pose Estimation of Heterogeneous Objects Using a Region Based Approach." In: *International Journal of Computer Vision* 118.1, pp. 95–112.

Hinterstoisser, S., V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab (2013). "Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes." In: *Asian Conference on Computer Vision*, pp. 548–562.

Hirschmüller, H. (2005). "Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information." In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 2, pp. 807–814.

Hodaň, T., M. Sundermeyer, B. Drost, Y. Labbé, E. Brachmann, F. Michel, C. Rother, and J. Matas (2020). "BOP Challenge 2020 on 6D Object Localization." In: *European Conference on Computer Vision Workshops*, pp. 577–594.

Hogg, D. (1983). "Model-based Vision: A Program to See a Walking Person." In: *Image and Vision Computing* 1.1, pp. 5–20.

Huang, H., F. Zhong, and X. Qin (2022). "Pixel-Wise Weighted Region-Based 3D Object Tracking using Contour Constraints." In: *IEEE Transactions on Visualization and Computer Graphics* 28.12, pp. 4319–4331.

Huang, H., F. Zhong, Y. Sun, and X. Qin (2020). "An Occlusion-aware Edge-Based Method for Monocular 3D Object Tracking using Edge Confidence." In: *Computer Graphics Forum* 39.7, pp. 399–409.

Imperoli, M. and A. Pretto (2015). "D^{2}CO: Fast and Robust Registration of 3D Textureless Objects Using the Directional Chamfer Distance." In: *International Conference on Computer Vision Systems*, pp. 316–328.

Isard, M. and A. Blake (1998). "CONDENSATION — Conditional Density Propagation for Visual Tracking." In: *International Journal of Computer Vision* 29.1, pp. 5–28.

Issac, J., M. Wüthrich, C. G. Cifuentes, J. Bohg, S. Trimpe, and S. Schaal (2016). "Depth-Based Object Tracking Using a Robust Gaussian Filter." In: *IEEE International Conference on Robotics and Automation*, pp. 608–615.

Izatt, G., G. Mirano, E. Adelson, and R. Tedrake (2017). “Tracking Objects with Point Clouds from Vision and Touch.” In: *International Conference on Robotics and Automation*, pp. 4000–4007.

Jurie, F. and M. Dhome (2002). “Hyperplane Approximation for Template Matching.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.7, pp. 996–1000.

Kanazawa, A., M. J. Black, D. W. Jacobs, and J. Malik (2018). “End-to-End Recovery of Human Shape and Pose.” In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7122–7131.

Kehl, W., F. Tombari, S. Ilic, and N. Navab (2017). “Real-Time 3D Model Tracking in Color and Depth on a Single CPU Core.” In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 465–473.

Klein, G. and D. W. Murray (2006). “Full-3D Edge Tracking with a Particle Filter.” In: *British Machine Vision Conference*, pp. 1119–1128.

Klingensmith, M., T. Galluzzo, C. M. Dellin, M. Kazemi, J. A. Bagnell, and N. Pollard (2013). “Closed-loop Servoing using Real-time Markerless Arm Tracking.”

Krainin, M., P. Henry, X. Ren, and D. Fox (2011). “Manipulator and Object Tracking for In-hand 3D Object Modeling.” In: *The International Journal of Robotics Research* 30.11, pp. 1311–1327.

Krull, A., F. Michel, E. Brachmann, S. Gumhold, S. Ihrke, and C. Rother (2015). “6-DOF Model Based Tracking via Object Coordinate Regression.” In: *Asian Conference on Computer Vision*, pp. 384–399.

La Cascia, M., S. Sclaroff, and V. Athitsos (2000). “Fast, Reliable Head Tracking under Varying Illumination: An Approach Based on Registration of Texture-Mapped 3D Models.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.4, pp. 322–336.

Labbé, Y., J. Carpentier, M. Aubry, and J. Sivic (2020). “CosyPose: Consistent Multi-view Multi-object 6D Pose Estimation.” In: *European Conference on Computer Vision*, pp. 574–591.

Labbé, Y., J. Carpentier, M. Aubry, and J. Sivic (2021). “Single-View Robot Pose and Joint Angle Estimation via Render & Compare.” In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1654–1663.

Lankton, S. and A. Tannenbaum (2008). “Localizing Region-Based Active Contours.” In: *IEEE Transactions on Image Processing* 17.11, pp. 2029–2039.

Lepetit, V. and P. Fua (2005). *Monocular Model-Based 3D Tracking of Rigid Objects: A Survey*. Foundations, Trends in Computer Graphics, and Vision.

Lepetit, V., F. Moreno-Noguer, and P. Fua (2009). “EPnP: An Accurate O(n) Solution to the PnP Problem.” In: *International Journal of Computer Vision* 81.2, pp. 155–166.

Leutenegger, S., M. Chli, and R. Y. Siegwart (2011). “BRISK: Binary Robust Invariant Scalable Keypoints.” In: *IEEE International Conference on Computer Vision*, pp. 2548–2555.

Li, J.-C., F. Zhong, S.-H. Xu, and X.-Y. Qin (2021). "3D Object Tracking with Adaptively Weighted Local Bundles." In: *Journal of Computer Science and Technology* 36.3, pp. 555–571.

Li, Y., G. Wang, X. Ji, Y. Xiang, and D. Fox (2018). "DeepIM: Deep Iterative Matching for 6D Pose Estimation." In: *European Conference on Computer Vision*, pp. 695–711.

Lipson, L., Z. Teed, A. Goyal, and J. Deng (2022). "Coupled Iterative Refinement for 6D Multi-Object Pose Estimation." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6728–6737.

Liu, F., Z. Wei, and G. Zhang (2021). "An Off-Board Vision System for Relative Attitude Measurement of Aircraft." In: *IEEE Transactions on Industrial Electronics* 69.4, pp. 4225–4233.

Liu, Y. and A. Namiki (2021). "Articulated Object Tracking by High-Speed Monocular RGB Camera." In: *IEEE Sensors Journal* 21.10, pp. 11899–11915.

Liu, Y., P. Sun, and A. Namiki (2020). "Target Tracking of Moving and Rotating Object by High-Speed Monocular Active Vision." In: *IEEE Sensors Journal* 20.12, pp. 6727–6744.

Lourakis, M. and X. Zabulis (2013). "Model-Based Pose Estimation for Rigid Objects." In: *Computer Vision Systems*, pp. 83–92.

Lowe, D. G. (1992). "Robust Model-based Motion Tracking Through the Integration of Search and Estimation." In: *International Journal of Computer Vision* 8.2, pp. 113–122.

Lowe, D. G. (1991). "Fitting Parameterized Three-Dimensional Models to Images." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.5, pp. 441–450.

Lowe, D. G. (2004). "Distinctive Image Features from Scale-Invariant Keypoints." In: *International Journal of Computer Vision* 60.2, pp. 91–110.

Lucas, B. D. and T. Kanade (1981). "An Iterative Image Registration Technique with an Application to Stereo Vision." In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence*. Vol. 2, pp. 674–679.

Luo, W., J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim (2021). "Multiple Object Tracking: A Literature Review." In: *Artificial Intelligence* 293, p. 103448.

Ma, J., X. Jiang, A. Fan, J. Jiang, and J. Yan (2021). "Image Matching from Handcrafted to Deep Features: A Survey." In: *International Journal of Computer Vision* 129.1, pp. 23–79.

Manhardt, F., W. Kehl, N. Navab, and F. Tombari (2018). "Deep Model-Based 6D Pose Refinement in RGB." In: *European Conference on Computer Vision*, pp. 833–849.

Marchand, É., P. Bouthemy, and F. Chaumette (2001). "A 2D–3D Model-based Approach to Real-time Visual Tracking." In: *Image and Vision Computing* 19.13, pp. 941–955.

Marougkas, I., P. Koutras, N. Kardaris, G. Retsinas, G. Chalvatzaki, and P. Maragos (2020). "How to Track Your Dragon: A Multi-attentional Framework for Real-Time RGB-D 6-DOF Object Pose Tracking." In: *European Conference on Computer Vision Workshops*. Ed. by A. Bartoli and A. Fusiello, pp. 682–699.

Masuda, T., K. Sakaue, and N. Yokoya (1996). "Registration and Integration of Multiple Range Images for 3-D Model Construction." In: *Proceedings of 13th International Conference on Pattern Recognition*, 879–883 vol.1.

Meinhardt, T., A. Kirillov, L. Leal-Taixe, and C. Feichtenhofer (2022). “Trackformer: Multi-object Tracking with Transformers.” In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8844–8854.

Michel, F., A. Krull, E. Brachmann, M. Yang, S. Gumhold, and C. Rother (2015). “Pose Estimation of Kinematic Chain Instances via Object Coordinate Regression.” In: *British Machine Vision Conference*, pp. 181.1–181.11.

Moravec, H. (1980). “Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover.” PhD thesis. Stanford University.

Mörwald, T., M. Zillich, and M. Vincze (2009). “Edge Tracking of Textured Objects with a Recursive Particle Filter.” In: *Proceedings of the Graphicon*, pp. 248–253.

Mündermann, L., S. Corazza, and T. P. Andriacchi (2007). “Accurately Measuring Human Movement Using Articulated ICP with Soft-Joint Constraints and a Repository of Articulated Models.” In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–6.

Mur-Artal, R. and J. D. Tardós (2017). “ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras.” In: *IEEE Transactions on Robotics* 33.5, pp. 1255–1262.

Newcombe, R. A., A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon (2011). “KinectFusion: Real-Time Dense Surface Mapping and Tracking.” In: *IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136.

Nickels, K. and S. Hutchinson (2001). “Model-based Tracking of Complex Articulated Objects.” In: *IEEE Transactions on Robotics and Automation* 17.1, pp. 28–36.

Pauwels, K., L. Rubio, J. Díaz, and E. Ros (2013). “Real-Time Model-Based Rigid Object Pose Estimation and Tracking Combining Dense and Sparse Visual Cues.” In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2347–2354.

Pauwels, K. and D. Kragic (2015). “SimTrack: A Simulation-based Framework for Scalable Real-time Object Pose Detection and Tracking.” In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1300–1307.

Pauwels, K., L. Rubio, and E. Ros (2014). “Real-Time Model-Based Articulated Object Pose Detection and Tracking with Variable Rigidity Constraints.” In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3994–4001.

Pellegrini, S., K. Schindler, and D. Nardi (2008). “A Generalisation of the ICP Algorithm for Articulated Bodies.” In: *British Machine Vision Conference*, pp. 87.1–87.10.

Pfanne, M. (2022a). “Grasp State Estimation.” In: *In-Hand Object Localization and Control: Enabling Dexterous Manipulation with Robotic Hands*, pp. 57–123.

Pfanne, M. (2022b). “Impedance-Based Object Control.” In: *In-Hand Object Localization and Control: Enabling Dexterous Manipulation with Robotic Hands*, pp. 125–171.

Pfanne, M. and M. Chalon (2017). “EKF-based In-hand Object Localization from Joint Position and Torque Measurements.” In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Piscataway, New Jersey: IEEE, pp. 2464–2470.

Pfanne, M., M. Chalon, F. Stulp, and A. Albu-Schäffer (2018). "Fusing Joint Measurements and Visual Features for In-Hand Object Pose Estimation." In: *IEEE Robotics and Automation Letters* 3.4, pp. 3497–3504.

Pfanne, M., M. Chalon, F. Stulp, H. Ritter, and A. Albu-Schäffer (2020). "Object-Level Impedance Control for Dexterous In-Hand Manipulation." In: *IEEE Robotics and Automation Letters* 5.2, pp. 2987–2994.

Piga, N. A., F. Bottarel, C. Fantacci, G. Vezzani, U. Pattacini, and L. Natale (2021). "MaskUKF: An Instance Segmentation Aided Unscented Kalman Filter for 6D Object Pose and Velocity Tracking." In: *Frontiers in Robotics and AI* 8.

Piga, N. A., Y. Onyshchuk, G. Pasquale, U. Pattacini, and L. Natale (2022). "ROFT: Real-Time Optical Flow-Aided 6D Object Pose and Velocity Tracking." In: *IEEE Robotics and Automation Letters* 7.1, pp. 159–166.

Pomerleau, F., F. Colas, and R. Siegwart (2015). "A Review of Point Cloud Registration Algorithms for Mobile Robotics." In: *Foundations and Trends in Robotics* 4.1, pp. 1–104.

Prisacariu, V. A., O. Kähler, D. W. Murray, and I. D. Reid (2015). "Real-Time 3D Tracking and Reconstruction on Mobile Phones." In: *IEEE Transactions on Visualization and Computer Graphics* 21.5, pp. 557–570.

Prisacariu, V. A. and I. D. Reid (2012). "PWP3D: Real-Time Segmentation and Tracking of 3D Objects." In: *International Journal of Computer Vision* 98.3, pp. 335–354.

Pulli, K. (1999). "Multiview Registration for Large Data Sets." In: *Second International Conference on 3-D Digital Imaging and Modeling*, pp. 160–168.

Pupilli, M. and A. Calway (2006). "Real-Time Camera Tracking Using Known 3D Models and a Particle Filter." In: *18th International Conference on Pattern Recognition*, pp. 199–203.

Rauch, C., T. Hospedales, J. Shotton, and M. Fallon (2018). "Visual Articulated Tracking in the Presence of Occlusions." In: *IEEE International Conference on Robotics and Automation*, pp. 643–650.

Rehg, J. M. and T. Kanade (1994). "Visual Tracking of High DOF Articulated Structures: An Application to Human Hand Tracking." In: *European Conference on Computer Vision*. Ed. by J.-O. Eklundh, pp. 35–46.

Reichert, A. E. (2021). "Tracking of Rigid Bodies in the Context of Minimally Invasive Surgical Training." BA thesis. Duale Hochschule Baden-Württemberg.

Reinecke, J., B. Deutschmann, A. Dietrich, and M. Hutter (2020). "An Anthropomorphic Robust Robotic Torso for Ventral/Dorsal and Lateral Motion With Weight Compensation." In: *IEEE Robotics and Automation Letters* 5.3, pp. 3876–3883.

Ren, C. Y., V. A. Prisacariu, O. Kähler, I. D. Reid, and D. W. Murray (2017). "Real-Time Tracking of Single and Multiple Objects from Depth-Colour Imagery Using 3D Signed Distance Functions." In: *International Journal of Computer Vision* 124.1, pp. 80–95.

Ren, C. Y. and I. D. Reid (2012). "A Unified Energy Minimization Framework for Model Fitting in Depth." In: *European Conference on Computer Vision Workshops and Demonstrations*. Ed. by A. Fusiello, V. Murino, and R. Cucchiara, pp. 72–82.

Rosenhahn, B., T. Brox, and J. Weickert (2007). "Three-Dimensional Shape Knowledge for Joint Image Segmentation and Pose Tracking." In: *International Journal of Computer Vision* 73.3, pp. 243–262.

Rosten, E. and T. Drummond (2005). "Fusing Points and Lines for High Performance Tracking." In: *IEEE International Conference on Computer Vision*. Vol. 2, pp. 1508–1515.

Rothe, J. (2022). "Combining Global and Local 6DoF Pose Estimation for Real-World Applications." MA thesis. Technical University of Munich.

Rublee, E., V. Rabaud, K. Konolige, and G. Bradski (2011). "ORB: An efficient alternative to SIFT or SURF." In: *IEEE International Conference on Computer Vision*, pp. 2564–2571.

Rusinkiewicz, S. and M. Levoy (2001). "Efficient Variants of the ICP Algorithm." In: *Third International Conference on 3-D Digital Imaging and Modeling*, pp. 145–152.

Salas-Moreno, R. F., R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison (2013). "SLAM++: Simultaneous Localisation and Mapping at the Level of Objects." In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1352–1359.

Sarlin, P.-E., D. DeTone, T. Malisiewicz, and A. Rabinovich (2020). "SuperGlue: Learning Feature Matching With Graph Neural Networks." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4937–4946.

Schmaltz, C., B. Rosenhahn, T. Brox, and J. Weickert (2012). "Region-Based Pose Tracking with Occlusions Using 3D Models." In: *Machine Vision and Applications* 23.3, pp. 557–577.

Schmidt, T., K. Hertkorn, R. Newcombe, Z. Marton, M. Suppa, and D. Fox (2015a). "Depth-Based Tracking with Physical Constraints for Robot Manipulation." In: *IEEE International Conference on Robotics and Automation*, pp. 119–126.

Schmidt, T., R. Newcombe, and D. Fox (2015b). "DART: Dense Articulated Real-time Tracking with Consumer Depth Cameras." In: *Autonomous Robots* 39.3, pp. 239–258.

Seo, B.-K., H. Park, J.-I. Park, S. Hinterstoisser, and S. Ilic (2014). "Optimal Local Searching for Fast and Robust Textureless 3D Object Tracking in Highly Cluttered Backgrounds." In: *IEEE Transactions on Visualization and Computer Graphics* 20.1, pp. 99–110.

Seo, B.-K. and H. Wuest (2016). "A Direct Method for Robust Model-Based 3D Object Tracking from a Monocular RGB Image." In: *European Conference on Computer Vision Workshops*, pp. 551–562.

Sepp, W. (2006). "Efficient Tracking in 6-DoF based on the Image-Constancy Assumption in 3-D." In: *18th International Conference on Pattern Recognition*. Vol. 3, pp. 59–62.

Shan, Y., Z. Liu, and Z. Zhang (2001). "Model-Based Bundle Adjustment with Application to Face Modeling." In: *Eighth IEEE International Conference on Computer Vision*. Vol. 2, pp. 644–651.

Simon, D. A. (1996). "Fast and Accurate Shape-Based Registration." PhD thesis. Carnegie Mellon University Pittsburgh.

Simon, G. and M.-O. Berger (1998). "A Two-stage Robust Statistical Method for Temporal Registration from Features of Various Type." In: *Sixth International Conference on Computer Vision*, pp. 261–266.

Stoiber, M. (2019). "Real-Time In-Hand Object Tracking and Sensor Fusion for Advanced Robotic Manipulation." MA thesis. Technical University of Munich.

Stoiber, M., M. Elsayed, A. E. Rechert, F. Steidle, D. Lee, and R. Triebel (2023a). "Fusing Visual Appearance and Geometry for Multi-Modality 6DoF Object Tracking." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1170–1177.

Stoiber, M., M. Pfanne, K. H. Strobl, R. Triebel, and A. Albu-Schäffer (2020). "A Sparse Gaussian Approach to Region-Based 6DoF Object Tracking." In: *Asian Conference on Computer Vision*, pp. 666–682.

Stoiber, M., M. Pfanne, K. H. Strobl, R. Triebel, and A. Albu-Schäffer (2022a). "SRT3D: A Sparse Region-Based 3D Object Tracking Approach for the Real World." In: *International Journal of Computer Vision* 130.4, pp. 1008–1030.

Stoiber, M., M. Sundermeyer, W. Boerdijk, and R. Triebel (2023b). "A Multi-body Tracking Framework - From Rigid Objects to Kinematic Structures."

Stoiber, M., M. Sundermeyer, and R. Triebel (2022b). "Iterative Corresponding Geometry: Fusing Region and Depth for Highly Efficient 3D Tracking of Textureless Objects." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6855–6865.

Sun, J., Z. Shen, Y. Wang, H. Bao, and X. Zhou (2021). "LoFTR: Detector-Free Local Feature Matching With Transformers." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8922–8931.

Sun, X., J. Zhou, W. Zhang, Z. Wang, and Q. Yu (2021). "Robust Monocular Pose Tracking of Less-Distinct Objects Based on Contour-Part Model." In: *IEEE Transactions on Circuits and Systems for Video Technology* 31.11, pp. 4409–4421.

Sundermeyer, M., Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel (2018). "Implicit 3D Orientation Learning for 6D Object Detection from RGB Images." In: *European Conference on Computer Vision*, pp. 712–729.

Tan, D. J., N. Navab, and F. Tombari (2017). "Looking Beyond the Simple Scenarios: Combining Learners and Optimizers in 3D Temporal Tracking." In: *IEEE Transactions on Visualization and Computer Graphics* 23.11, pp. 2399–2409.

Tan, D. J., T. Cashman, J. Taylor, A. Fitzgibbon, D. Tarlow, S. Khamis, S. Izadi, and J. Shotton (2016). "Fits Like a Glove: Rapid and Reliable Hand Shape Personalization." In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5610–5619.

Tan, D. J. and S. Ilic (2014). "Multi-forest Tracker: A Chameleon in Tracking." In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1202–1209.

Taylor, J., L. Bordeaux, T. Cashman, B. Corish, C. Keskin, T. Sharp, E. Soto, D. Sweeney, J. Valentin, B. Luff, A. Topalian, E. Wood, S. Khamis, P. Kohli, S. Izadi, R. Banks, A. Fitzgibbon, and J. Shotton (2016). "Efficient and Precise Interactive Hand Tracking through Joint, Continuous Optimization of Pose and Correspondences." In: *ACM Transactions on Graphics* 35.4.

Teulière, C., E. Marchand, and L. Eck (2010). "Using Multiple Hypothesis in Model-based Tracking." In: *IEEE International Conference on Robotics and Automation*, pp. 4559–4565.

Thielmann, S., U. Seibold, R. Haslinger, G. Passig, T. Bahls, S. Jörg, M. Nickl, A. Nothhelfer, U. Hagn, and G. Hirzinger (2010). "MICA - A New Generation of Versatile

Instruments in Robotic Surgery." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 871–878.

Tjaden, H., U. Schwanecke, E. Schömer, and D. Cremers (2018). "A Region-Based Gauss-Newton Approach to Real-Time Monocular Multiple Object Tracking." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.8, pp. 1797–1812.

Tola, E., V. Lepetit, and P. Fua (2010). "DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.5, pp. 815–830.

Tölgyessy, M., M. Dekan, L. Chovanec, and P. Hubinskŷ (2021). "Evaluation of the Azure Kinect and Its Comparison to Kinect V1 and Kinect V2." In: *Sensors* 21.2.

Turk, G. and M. Levoy (1994). "Zippered Polygon Meshes from Range Images." In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, pp. 311–318.

Vacchetti, L., V. Lepetit, and P. Fua (2004). "Stable Real-Time 3D Tracking Using Online and Offline Information." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.10, pp. 1385–1391.

Wang, B., F. Zhong, and X. Qin (2019). "Robust Edge-based 3D Object Tracking with Direction-based Pose Validation." In: *Multimedia Tools and Applications* 78.9, pp. 12307–12331.

Wang, C., R. Martín-Martín, D. Xu, J. Lv, C. Lu, L. Fei-Fei, S. Savarese, and Y. Zhu (2020). "6-PACK: Category-level 6D Pose Tracker with Anchor-Based Keypoints." In: *IEEE International Conference on Robotics and Automation*, pp. 10059–10066.

Wang, C., D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese (2019). "DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3338–3347.

Wang, C.-Y., A. Bochkovskiy, and H.-Y. M. Liao (2022). "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-time Object Detectors."

Wang, G., F. Manhardt, F. Tombari, and X. Ji (2021). "GDR-Net: Geometry-Guided Direct Regression Network for Monocular 6D Object Pose Estimation." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16611–16621.

Wang, G., B. Wang, F. Zhong, X. Qin, and B. Chen (2015). "Global Optimal Searching for Textureless 3D Object Tracking." In: *The Visual Computer* 31.6, pp. 979–988.

Weik, S. (1997). "Registration of 3-D Partial Surface Models Using Luminance and Depth Information." In: *International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pp. 93–100.

Wen, B., C. Mitash, B. Ren, and K. E. Bekris (2020). "se(3)-TrackNet: Data-driven 6D Pose Tracking by Calibrating Image Residuals in Synthetic Domains." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 10367–10373.

Weng, Y., H. Wang, Q. Zhou, Y. Qin, Y. Duan, Q. Fan, B. Chen, H. Su, and L. J. Guibas (2021). "CAPTRA: CAtegory-Level Pose Tracking for Rigid and Articulated Objects From Point Clouds." In: *IEEE/CVF International Conference on Computer Vision*, pp. 13209–13218.

Whelan, T., S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison (2015). "ElasticFusion: Dense SLAM Without a Pose Graph." In: *Robotics: Science and Systems*.

Widmaier, F., D. Kappler, S. Schaal, and J. Bohg (2016). "Robot Arm Pose Estimation by Pixel-wise Regression of Joint Angles." In: *IEEE International Conference on Robotics and Automation*, pp. 616–623.

Wu, P.-C., Y.-Y. Lee, H.-Y. Tseng, H.-I. Ho, M.-H. Yang, and S.-Y. Chien (2017). "A Benchmark Dataset for 6DoF Object Pose Tracking." In: *IEEE International Symposium on Mixed and Augmented Reality*, pp. 186–191.

Wuest, H., F. Vial, and D. Stricker (2005). "Adaptive Line Tracking with Multiple Hypotheses for Augmented Reality." In: *Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 62–69.

Wüthrich, M., J. Bohg, D. Kappler, C. Pfreundt, and S. Schaal (2015). "The Coordinate Particle Filter - A novel Particle Filter for High Dimensional Systems." In: *IEEE International Conference on Robotics and Automation*, pp. 2454–2461.

Wüthrich, M., C. G. Cifuentes, S. Trimpe, F. Meier, J. Bohg, J. Issac, and S. Schaal (2016). "Robust Gaussian Filtering Using a Pseudo Measurement." In: *American Control Conference*, pp. 3606–3613.

Wüthrich, M., P. Pastor, M. Kalakrishnan, J. Bohg, and S. Schaal (2013). "Probabilistic Object Tracking using a Range Camera." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3195–3202.

Xiang, Y., T. Schmidt, V. Narayanan, and D. Fox (2018). "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes." In: *Robotics: Science and Systems*.

Yi, K. M., E. Trulls, V. Lepetit, and P. Fua (2016). "LIFT: Learned Invariant Feature Transform." In: *European Conference on Computer Vision*, pp. 467–483.

Yilmaz, A., O. Javed, and M. Shah (2006). "Object Tracking: A Survey." In: *ACM Computing Surveys* 38.4, p. 13.

Zeng, F., B. Dong, Y. Zhang, T. Wang, X. Zhang, and Y. Wei (2022). "MOTR: End-to-End Multiple-Object Tracking with Transformer." In: *European Conference on Computer Vision*, pp. 659–675.

Zhang, Y., P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang (2022). "ByteTrack: Multi-object Tracking by Associating Every Detection Box." In: *European Conference on Computer Vision*, pp. 1–21.

Zhao, S., L. Wang, W. Sui, H.-y. Wu, and C. Pan (2014). "3D Object Tracking via Boundary Constrained Region-Based Model." In: *IEEE International Conference on Image Processing*, pp. 486–490.

Zheng, L., A. Leonardis, T. H. E. Tse, N. Horanyi, H. Chen, W. Zhang, and H. J. Chang (2022). "TP-AE: Temporally Primed 6D Object Pose Tracking with Auto-Encoders." In: *International Conference on Robotics and Automation*, pp. 10616–10623.

Zhong, L., M. Lu, and L. Zhang (2018). "A Direct 3D Object Tracking Method Based on Dynamic Textured Model Rendering and Extended Dense Feature Fields." In: *IEEE Transactions on Circuits and Systems for Video Technology* 28.9, pp. 2302–2315.

Zhong, L. and L. Zhang (2019). "A Robust Monocular 3D Object Tracking Method Combining Statistical and Photometric Constraints." In: *International Journal of Computer Vision* 127.8, pp. 973–992.

Zhong, L., Y. Zhang, H. Zhao, A. Chang, W. Xiang, S. Zhang, and L. Zhang (2020a). "Seeing Through the Occluders: Robust Monocular 6-DOF Object Pose Tracking via Model-guided Video Object Segmentation." In: *IEEE Robotics and Automation Letters* 5.4, pp. 5159–5166.

Zhong, L., X. Zhao, Y. Zhang, S. Zhang, and L. Zhang (2020b). "Occlusion-Aware Region-Based 3D Pose Tracking of Objects With Temporally Consistent Polar-Based Local Partitioning." In: *IEEE Transactions on Image Processing* 29, pp. 5065–5078.

Zimmermann, C. and T. Brox (2017). "Learning to Estimate 3D Hand Pose from Single RGB Images." In: *IEEE International Conference on Computer Vision*, pp. 4913–4921.

Zuo, Y., W. Qiu, L. Xie, F. Zhong, Y. Wang, and A. L. Yuille (2019). "CRAVES: Controlling Robotic Arm With a Vision-Based Economic System." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4209–4218.