TU WIEN Informatics

6D Object Tracking as a Reinforcement Learning Task

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Technische Informatik

eingereicht von

Konstantin Röhrl, BSc
Matrikelnummer 11777779

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Markus Vincze
Mitwirkung: Dipl.-Ing. Dr.techn. Dominik Bauer
Dipl.-Ing. Bernhard Neuberger

Wien, 27. März 2024

Konstantin Röhrl	Markus Vincze

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

6D Object Tracking as a Reinforcement Learning Task

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Computer Engineering

by

Konstantin Röhrl, BSc
Registration Number 11777779

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Markus Vincze
Assistance: Dipl.-Ing. Dr.techn. Dominik Bauer
Dipl.-Ing. Bernhard Neuberger

Vienna, 27th March, 2024

Konstantin Röhrl | Markus Vincze

Erklärung zur Verfassung der Arbeit

Konstantin Röhrl, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 27. März 2024

Konstantin Röhrl

Danksagung

Ich bin zutiefst dankbar für die unerschütterliche Unterstützung von meiner Familie während meiner Studienjahre. Insbesondere möchte ich mich von ganzem Herzen bei meinem Vater Leonhard für seinen Einsatz und seine Hingabe bedanken, um mir die Möglichkeit zu bieten, meine akademische Laufbahn zu verfolgen. Auch bei meiner Schwester Theresa möchte ich mich herzlichst dafür bedanken, dass sie immer an mich geglaubt hat und mit lieben Worten mich immer ermutigt hat. Ohne die Unterstützung dieser beiden Personen wäre für mich ein Abschluss des Studiums unmöglich gewesen.

Des weiteren möchte ich mich beim Leiter der Vision For Robotics (V4R) Gruppe Markus Vincze dafür bedanken, dass er mir die Möglichkeit gegeben hat, die Masterarbeit in seiner Gruppe zu erstellen. Während des gesamten Prozesses der Fertigung der Masterarbeit erhielt ich von Dominik Bauer immer detaillierte und konstruktive Rückmeldungen, was mir immens dabei half, mein eigenes Verständnis zu formen und wertvolle Einblicke in diesem Feld zu erhalten. Seine Anregungen und Expertise haben mich motiviert und auch inspiriert, meine eigenen Ideen einzubringen. Ich möchte außerdem die essentielle Rolle von Bernhard Neuberger für diese Masterarbeit hervorheben. Sein tiefgreifendes Verständnis im Feld der Robotik ermöglichten das schnelle und einfache Einbetten meiner Methode in eine reale Robotik-Anwendung. Insbesondere zu Beginn war die Demonstration am echten Roboter ein äußerst spannender Ausblich für die Masterarbeit, der sich rund 1,5 Jahre später am Ende der Arbeit als sehr zufriedenstellender Meilenstein herausstellte. Darüberhinaus möchte ich mich auch bei Timothy Patten für seine Einführung in das Gebiet der Robot Vision im gleichnamigen Kurs bedanken, wodurch mein Interesse in diesem Gebiet geweckt wurde, was mich schließlich zu Dominik Bauer führte. Ich bin außerdem dankbar für all die anderen Leute hier in der V4R Gruppe, mit denen ich immer interessante Diskussionen über alle möglichen Themen hatte und die mich auch immer bei Fragen mit wertvollem Feedback unterstützten.

Acknowledgements

I am deeply grateful for the unwavering support received from my family throughout my entire study. In particular, I would like to express my heartfelt gratitude to my father Leonhard for his commitment and dedication to enabling me to pursue my academic career. I also want to thank my sister for always believing in me and being encouraging. Without their support, finishing this study would have been impossible for me.

Furthermore, I am thankful to the head of the vision for robotics (V4R) group Markus Vincze for giving me the opportunity to write my master thesis in his group. Throughout the development of this thesis, Dominik Bauer consistently offered detailed and constructive feedback, helping me to shape my understanding and gather valuable insights in this field. His encouragement and expertise kept me motivated and inspired me to contribute my own ideas. I also want to emphasize Bernhard Neuberger's crucial role in this thesis. His profound understanding of robotics enabled quick and easy integration of my method into a real-world robotic application, which was an exciting outlook that motivated me in the early stages and later proved to be a highly satisfying milestone after working around 1.5 years on this thesis. Moreover, I am thankful to Timothy Patten for introducing me to the field of robot vision during the course of the same name, initially sparking my interest in this field and eventually connecting me with Dominik Bauer. I am also thankful to all the other people I met at the V4R group, having interesting discussions on- and off-topic and always provided me with useful feedback whenever I had questions.

Kurzfassung

Objekte in Videos zu lokalisieren ist eine wichtige Aufgabe in dynamischen Umgebungen, insbesondere in der Robotik, wenn sich entweder die Position und Orientierung eines Objekts oder die Kameraperspektive kontinuierlich ändern. Methoden, die Objekte in einem einzelnen Bild lokalisieren, versagen aufgrund der dynamischen Änderungen in der Umgebung. Obwohl Tracking-Methoden die zeitlichen und räumlichen Zusammenhänge über mehrere Bilder hinweg berücksichtigen, können sich Fehler im Laufe der Zeit akkumulieren und letztendlich zum Scheitern der Methode führen. Des Weiteren erschweren teilweise verdeckte Objekte, beispielsweise durch Hände oder anderen Objekte, die akkurate Lokalisation über längere Zeiträume. In manchen Szenarien, wie bei komplett verdeckten Objekten, ist es außerdem nicht möglich, die Position und Orientierung des Objekts visuell zu bestimmen, wodurch Tracking konsequenterweise versagt.

Während RGB(-D) Methoden große Datenmengen für das Training benötigen, schlagen wir einen vereinfachten Ansatz vor, der 6D Object Tracking als tiefenbasiertes Ausrichtungsproblem von niedrig aufgelösten 3D Punktwolken betrachtet. Als gemeinsame Reinforcement Learning (RL) Aufgabe berücksichtigen wir gleichzeitig Abhängigkeiten zwischen Bildern durch die Registrierung von aufeinanderfolgenden Beobachtungen in den Tiefenbildern (*Frame-To-Frame Registration*), sowie das Kompensieren von sich aufsummierenden Fehlern durch die Einbindung eines Modells als Referenz (*Frame-To-Model Refinement*). In unserem Belohnungssystem fördern wir gleichzeitig präzisere Ergebnisse über mehrerere Optimierungsschritte in einem einzigen Bild, als auch das langfristige, akkurate Lokalisieren des Objektes über mehrere Bilder hinweg. Wir verwenden ein Modell zur Berechnung von Referenz-Tiefendaten als geometrische Unsicherheit und kombinieren dies mit einer Unsicherheits-Metrik des Netzwerks, um die Methode effizient und autonom zu reinitialisieren.

Unsere Experimente zeigten, dass unser tiefenbasierter Ansatz den Unterschied zu aktuellen RGB-D Methoden verkleinert, während gleichzeitig alle anderen tiefenbasierten Methoden geschlagen werden. Insbesondere hat unsere gemeinsame Betrachtung von *Frame-To-Frame Registration* und *Frame-To-Model Refinement* bessere Ergebnisse erzielt als beide Ansätze isoliert, wodurch die Vorteile beider Methoden erfolgreich kombiniert wurden. Als Ergänzung zu unseren Experimenten an Datensätzen demonstrieren wir unsere Methode auch in einem Objekt-Übergabe Szenario von Mensch zu Roboter und zeigen somit die Anwendbarkeit in der realen Welt.

Abstract

6D object tracking is an essential task in dynamic environments, especially in applications involving robotic manipulation, where an object's pose is constantly manipulated or the camera is moving. While single-frame pose estimation fails in such a scenario, dedicated object tracking methods aim to leverage temporal and spatial coherence by exploiting priors, however, the persistence of small errors may accumulate over longer time horizons and contribute to the deterioration in tracking. The characteristics imposed by the environment, e.g. partially occluded objects due to hands or other objects, further exacerbate the challenges of accurate tracking. Moreover, heavily or fully occluded objects may occur in some scenarios, leading to a situation where tracking is unable to recover.

While RGB(-D) tracking methods rely on vast amounts of training data, we propose to learn 6D object trajectories from scratch as a simplified depth-only alignment problem, utilizing limited amounts of low-resolution 3D point clouds. In a joint Reinforcement Learning (RL) task, our novel network architecture exploits correspondences across frames by aligning consecutive observations (*frame-to-frame registration*), while accumulating errors are compensated via model-based recovery (*frame-to-model refinement*). Our multi-frame reward encourages our method to achieve close alignment in a single frame, while concurrently maintaining track of the object's pose across longer time horizons. Propagating the object's mask as a depth rendering is leveraged as a geometry-guided uncertainty metric and, in combination with the agent's uncertainty, contributes to an efficient and autonomous reinitialization heuristic.

Our experiments demonstrated, how our depth-only method closes the gap towards State of the Art (SotA) RGB-D methods and outperforms all other depth-based methods. Most importantly, fusing both subtasks contributes to improved tracking accuracy, as compared to either subtask in isolation. In addition to quantitative analysis, we showcase the efficacy of our tracking method in a robotic handover scenario, proving its practicability in dynamic real-world environments.

Contents

CHAPTER 1

Introduction

Understanding the environment is a crucial skill for any autonomous robot to fulfill its tasks safely. Every environment consists of a variety of objects - knowing their position and orientation relative to a specific anchor, such as a robot's end effector or a camera position, enables us to precisely define a robot's task. While single-frame pose estimation proves to be useful in static scenes, its efficacy is reduced in dynamic environments, where object poses are constantly manipulated, e.g. by a robot manipulator or a human. 6D object tracking extends pose estimation on this time axis, providing pose updates per time step. Considering a dynamic hand-over task, frequent pose updates enable a robot to follow the target and eventually execute a grasp.

Typically, dedicated object trackers exploit temporal and spatial coherence by matching information in consecutive frames. However, the accumulating error complicates accurate pose estimates in subsequent frames, resulting in deterioration in tracking. Conversely, single-frame pose estimation is able to recover from such errors, but invoking it in every frame is computationally expensive and may violate the tight time constraints of the tracking objective. Furthermore, computing the pose from scratch in every frame fails to exploit the temporal and spatial dependencies of object poses.

Bridging these two, we approach object tracking by 1) leveraging temporal and spatial coherence by aligning consecutive depth observations *(frame-to-frame registration)* and 2) simultaneously align with an object model acting as a reference frame *(frame-to-model refinement)*. Besides being used for (re-)initialization, the pose estimator becomes obsolete. A combination of Imitation Learning (IL) and RL efficiently fuses both subtasks and enables the agent to solve 6D object tracking as a point cloud alignment problem across multiple frames.

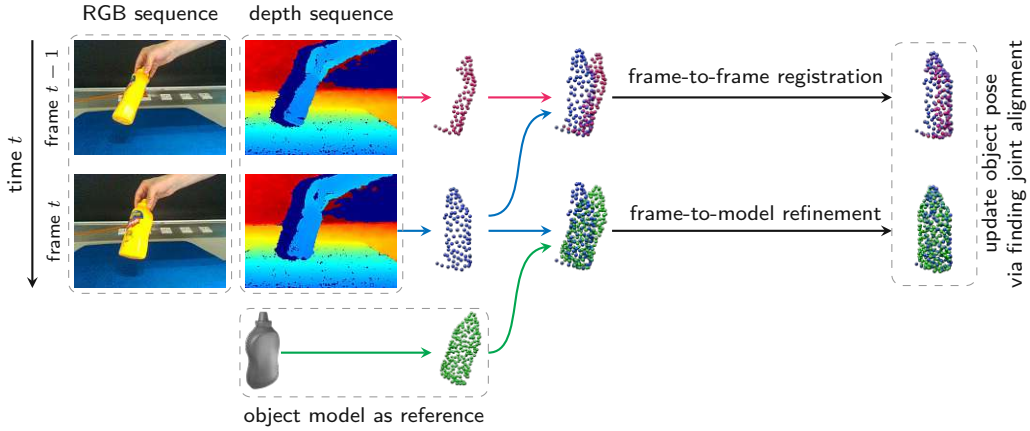

Figure 1.1: **6D Object Tracking as a Depth-Based Alignment Problem:** Our proposed approach exploits temporal and spatial correspondences by aligning noisy and error-afflicted depth observations from consecutive frames (*frame-to-frame registration*) while simultaneously aligning with an object model (*frame-to-model refinement*) as reference reduces the impact of accumulating errors. We update the pose in each frame by predicting a unified transformation that jointly solves both tasks.

1.1 Challenges and Research Questions

A common approach for 6D object tracking is to compute the temporal and spatial dependencies, e.g. by matching features in consecutive observations. Nevertheless, the tracking process may diverge from the target, e.g. due to occlusion, and the aggregated error makes future estimations a more complex problem without having any reference. Conversely, an object model guides single-frame pose estimation by acting as a reference. However, dismissing the temporal and spatial context of object poses by computing everything from scratch in each frame limits the temporal consistency and causes jittering object poses. Moreover, not relying on previous results leads to an increased computational effort and may violate the tight time constraints of the tracking objective.
The commonly chosen data modality in the field of object tracking [WB21, WTB+23, DMX+19] and pose estimation are color images (RGB) [XSNF18, LWJ+18]. This preference emerges from 1) the high resolution provided by widely available commercial-level RGB cameras, and 2) leveraging the efficacy of Convolutional Neural Networks (CNNs) or more recently Vision Transformers (ViTs) [DBK+20] previously applied to image classification tasks. Even though rich textures of objects enable accurate pose estimation with color information, illumination changes, and textureless objects limit the ability to establish corresponding RGB features. Importantly, an object's appearance in the RGB frame is also defined by its scale and distance to the camera. However, without exploiting depth information, ambiguities arise as various combinations of object scale and camera distance result in the same appearance from the camera's viewpoint. Furthermore, learning-based approaches require vast amounts of annotated and real(-istic)

RGB training data to learn the corresponding features.
On the other hand, features in depth data are not affected by the texture or illumination changes. As previously demonstrated in pose refinement applications [BPV21, BPV22], depth-based methods show stronger generalization as compared to color-based methods at the expense of introducing geometrical ambiguities, indiscriminate of symmetrical poses for lookalike shapes. For example, a scene with multiple box-shaped objects is predestined to mismatches between similar-shaped, but distinct object entities, when no accurate segmentation information is available. As a result, accurately propagating masks to keep track of the Region of Interest (ROI) is a challenging, but vital task in depth-based object tracking.
RGB-D methods aim to merge the best of both worlds. Extending RGB methods with the depth modality results in a significant accuracy gain [DMX+19]. Nevertheless, processing an additional data channel imposes further time constraints and truncates the available time window per frame. Moreover, synchronization issues in real-world applications result in the temporal misalignment of color and depth information and therefore pose a challenge to methods utilizing both modalities.
Regardless of the chosen method, heavy occlusion, e.g. by other objects, limits the inlier ratio in corresponding segments to be used for estimating the 6D pose, presenting one of the main challenges for object tracking. Furthermore, tracking may be unable to recover when gathered data points do not correspond to the object, e.g. due to full occlusion or a large pose error. In such cases, a method to identify when inevitable reinitializations are required is essential to any object tracking approach. With the absence of ground-truth data during test time, this is a challenging task and accurately triggering reinitializations requires a sophisticated approach to determine the plausibility of predictions.
This thesis addresses the just mentioned downsides by answering the following research questions:

- **RQ1**: How to approach object tracking as a reinforcement-learning problem? And how does this compare to SotA-approaches?
- **RQ2**: How to combine *frame-to-frame registration* with *frame-to-model refinement*? What impact does this combined approach have on the tracking accuracy over multiple frames as compared to either approach alone?
- **RQ3**: How robust is the object-tracker to observational noise, e.g. occlusion by other objects or hands and imperfect segmentation?
- **RQ4**: How to exploit the agent's behavior to balance the number of reinitializations with the tracking accuracy, when no annotated data is available?

1.2 Approach and Contribution

Instead of extending the complex frameworks for 6D object tracking to tackle these research questions, the work presented in this thesis aims to simplify the overall approach

and view 6D object tracking as a joint task of two depth-based point cloud alignment objectives illustrated in Fig. 1.1. This thesis proposes to embed an RL agent into a simple tracking pipeline to reuse the previously refined pose as initialization. A fundamental observation for this thesis is the complementary nature and joint consideration of two depth-based subtasks. We propose a hybrid tracking approach that exploits temporal and spatial dependencies by aligning the depth observations of two consecutive frames (*frame-to-frame registration*). Simultaneously, we align the current observation with the object model sampled under the last known pose (*frame-to-model refinement*). Importantly, incorporating the *frame-to-model refinement* subtask robustifies our approach against low overlap of consecutive observations by acting as a reference and additionally reduces the impact of the accumulating error.

In contrast to related work in 6D object tracking, we utilize only depth information for our approach. First of all, this enables us to achieve SotA performance, albeit utilizing only a fraction of available training data, as previously shown in depth-based pose refinement [BPV22], while the resulting network architecture remains efficient. We maintain the robustness of depth information by rendering the object model under its last known pose to keep track of corresponding segments. Previously, a combination of IL and RL has been applied to iterative pose refinement by reinforcing converging and pruning diverging steps during the refinement process [BPV22]. By extending this concept to incorporate an additional time axis, closer alignment is reinforced not only over refinement steps in a single frame but over longer time periods across frames.

We exploit geometric uncertainty indicators by comparing the rendered depth with the actual depth observation. In addition to that, we interpret the agent's predicted actions in the final refinement iteration as a measurement for misalignment. Intuitively, point clouds are aligned in the last refinement iteration as indicated by the agent's prediction. Otherwise, the predicted actions in the last step serve as a meaningful insight to indicate offset. Eventually, combining both uncertainty metrics enables our method to balance the number of reinitializations with the achieved tracking accuracy for finegrained tuning to specific application-dependent requirements.

We evaluate our approach against the SotA in 6D object tracking using procedures from related work. In addition, we carry out an in-depth ablation study on the YCB-V dataset [XSNF18] investigating the influence of various components and evaluate it using the standardized procedures from the BOP Toolkit [Mar23]. Moreover, we provide qualitative results on HO-3D-r [HROL20, PPL+21] and DexYCB [CYX+21], two in-hand object manipulation datasets.

Finally, our object tracking approach is demonstrated in a real-world setting. A KUKA lightweight robotic arm with a RealSense RGB-D camera mounted on its end-effector follows a target object manipulated by a human, keeping the object centered in the robot's point of view, i.e. in the camera frame, with a predefined safety margin of 50cm towards the front of the object. Within an emulated handover scenario, the robot disregards the safety margin and approaches the object to maintain a position close to the object's proximity for a potential grasp. After a certain time window, we assume the potential grasp to be finished and the robot reverts to the original safety margin of

50cm. Continuing tracking of the object concludes the demonstration of our method in a real-world application.
To summarize, this thesis is built on the following contributions:

- A novel network architecture exploits temporal and spatial coherence by fusing *frame-to-frame registration* with *frame-to-model refinement*.
- Incorporating an additional time axis in the reward reinforces closer alignment not only over multiple refinement steps but over multiple frames. Keeping track of the corresponding segments across frames is achieved by rendering the object model under the last known pose, resulting in a 2D mask used to extract the corresponding object regions.
- Inevitable reinitializations are detected by a combination of uncertainty-based triggers without relying on ground-truth data.

1.3 Organization

We provide an overview of key concepts used in this work in Chap. 2, including point cloud registration, pose estimation, pose refinement, and object tracking. In Chap. 3, we present a comprehensive summary of related work in all of those research fields. Our novel object tracking approach is presented in Chap. 4, followed by an exhaustive analysis of our contributions in Chap. 5 to evaluate against recent methods in object tracking. In addition to our experiments on datasets, we present our tracking method in a real-world robotic application in Chap. 6. We conclude this thesis in Chap 7 and give an outlook for future research.
This work may be considered as the follow-up work to SporeAgent [BPV22], thereby this thesis builds on vast amounts of the contributions of Bauer et al. presented in [BPV21, BPV22]. We explicitly mention the adopted components.

Noteworthy, the results presented in this thesis have already been published in the following publication:

- **Konstantin Röhrl**, Dominik Bauer, Timothy Patten, and Markus Vincze. TrackAgent: 6D Object Tracking via Reinforcement Learning. In *International Conference on Computer Vision Systems*, pages 323-355, 2023.

CHAPTER 2

Background

This chapter introduces concepts and their challenges along with definitions used in the entire thesis. We first introduce the core concept of this work, namely point cloud registration. Eventually, pose estimation and how it is related to pose refinement is discussed. Finally, we give an overview of 6D object tracking and show, how it builds on the previously discussed concepts.

2.1 Point Cloud Registration

One of the crucial concepts used in this work is point cloud registration. For the subsequent definitions, we follow the nomenclature introduced in [BPV21]. In general, point clouds are an unstructured set of points in the 3D space used to model surfaces and all kinds of shapes. Point cloud registration aims to realign two point clouds gathered from different observations in a common coordinate system. Considering an application in 6D object tracking, observations from two consecutive frames are offset exactly by the motion between frames, i.e. knowledge about the corresponding relative object or camera motion - 3D translation and 3D rotation - realigns both point clouds in the 3D space. The properties of using 3D data introduce various challenges as listed below:

- **Sensor Noise:** Depth data gathered from LiDAR sensors or 3D cameras is, as with every other sensor, error-afflicted and noisy. As a result, inaccuracies in the point clouds complicate the process of accurately determining correspondences.
- **Overlapping Fraction:** When observations are gathered from different viewpoints, e.g. as a robot follows a target trajectory or an object like a banana is manipulated, observations from different timestamps in general only partially overlap. Therefore, not all points in the source have a correspondence with the target and vice versa. As a result, matching the corresponding segments to estimate a transformation becomes a more complex task with decreasing overlap.

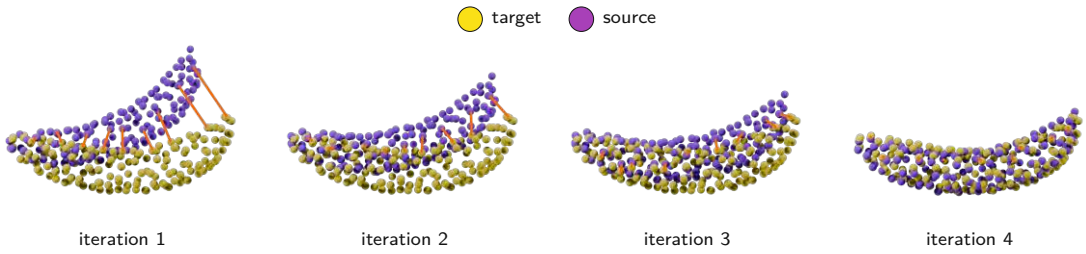

Figure 2.1: **Basic Concept of the ICP algorithm** [BM92]**:** In each iteration, points are matched to their nearest neighbor (marked in orange for a few selected points). Eventually, a transformation is found to reduce the distance between source and target. Once a certain convergence criteria, e.g. average point distance or the number of loop iterations, is fulfilled, the loop stops.

- **Outliers:** Either due to inaccurate segmentation, occlusion, or sensor noise, outliers significantly deteriorate the alignment. A method to identify and mitigate the influence of outliers leverages the robustness of any point cloud registration algorithm. Noteworthy, outlier-rejection interplays with the overlapping fraction. Pruning outliers too aggressively may complicate the registration process due to a truncated overlap.
- **Computational Limits:** To not violate any time constraints, the resolution of point clouds has to be constrained depending on the available resources in an application. Otherwise, matching points in two large-scale point clouds with a high level of detail leads to a significantly increased runtime. Recent work has shown, that 3D features can be efficiently learned from low-resolution point clouds [BPV21, BPV22], leading to less inference time. Therefore, the level of detail for any application dealing with point clouds has to be balanced with the computational capacity.

The Iterative Closest Point (ICP) algorithm [BM92] is the most fundamental work in this field and builds on a simple, yet effective two-stage loop. First, match the closest points in the source and target and eventually find a transformation minimizing the error between both point clouds. This two-step loop is executed until convergence is reached as visualized in Fig. 2.1. However, ICP may get stuck in a local optimum and therefore must be initialized with a sufficiently good estimate.

2.2 Pose Estimation

An intensely studied problem in computer vision is estimating the 6D object pose, i.e. 3D translation and 3D rotation of an object, from RGB-D images with respect to the

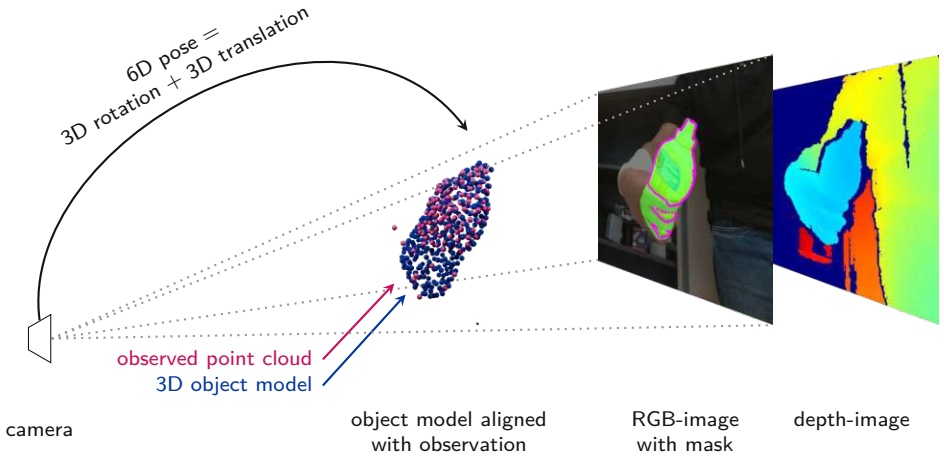

Figure 2.2: **Illustration of the 2D/3D Correspondences.** By utilizing observations from the 2D image, the task of pose estimation is to compute the 3D rotation and 3D translation (=6D pose) of an object relative to the camera. The observed 3D point cloud is retrieved by leveraging the corresponding object region from the 2D depth frame. Given an accurate pose estimate, the object model aligns with the observed point cloud in the 3D space. Illustration inspired by [Bau21].

camera position. As illustrated in Fig. 2.2, model-based pose estimation solves the 2D/3D correspondences by aligning the observation with a 3D object model, either via features in the color information [XSNF18], depth information [VLM18] or using both modalities [WMTJ21, LZZ+22].

A challenge for pose estimation is the reliable detection of all objects in an image. Furthermore, the sensitivity of color information, as utilized by the current best performing methods [Mar23], to changing illumination conditions affects the accuracy of estimated poses. Moreover, since the object may be located anywhere in the image, the huge space of potential poses typically causes a pose estimator to only yield an approximate estimate. For example, an error-afflicted pose estimate may cause a failed robotic grasp in a handover task. The additional consideration of depth information robustifies the estimation process, allowing to solve the scale invariance [AHS22] by exploiting the observed and viewpoint-dependent object depth.

According to the results of the BOP Challenge 2022 [Mar23], pose estimation solely relying on depth information caught less attention in recent years, where the current best performing depth-based pose estimator is not competitive compared to recent RGB(-D) methods.

Overall, pose estimation algorithms follow two main approaches to solve the 2D-3D correspondences. Indirect methods rely on a 3D object model and establish the 2D-3D correspondence by matching 2D key points to the 3D mesh. Commonly, the second stage includes a variant of the PnP/Ransac algorithm to determine the 6D pose. Even

Figure 2.3: **Partial Object Observation vs. Object Model:** *From Left to Right:* RGB and depth frame respectively with the spam object highlighted. The partial, noisy, and error-afflicted object view (blue) gathered from the observation. The object model (orange) represented as a 3D-mesh. The subsampled object model (orange) used to establish correspondences with the observation.

though indirect methods perform better in general, the PnP/Ransac stage is typically not differentiable, as required for learning-based approaches. Direct methods avoid this drawback by considering a point matching loss or separating the loss functions for different components.

2.3 Pose Refinement

Object pose refinement is an application of point cloud registration, typically used to refine inaccurate estimations yielded from a pose estimator. Noteworthy, the approximate pose estimate indicates the image region to be considered, allowing for a more finegrained refinement process due to a higher percentage of inliers and a constrained pose space. Color-based pose refinement aims to refine predictions by comparing rendered patches of the object model under its estimated pose with the actual RGB segment [LWJ+18]. Conversely, utilizing depth data enables to realign the 3D object model with the depth observation of the object by a point cloud registration method, e.g. the ICP algorithm [BM92].
For the scope of this thesis, we focus on depth-based pose refinement. In contrast to general point cloud registration that aligns point clouds with similar properties, the nature of either point cloud significantly differs in pose refinement. The 3D object model aims for a noise-free and complete representation of the object. Conversely, the observed object depth is viewpoint-dependent and covers only a partial view. In addition, point coordinates may be jittered due to sensor noise and outliers caused by inaccurate segmentation skew the alignment process. Importantly, pose refinement may not converge towards the ground-truth pose and possibly even result in a worsened estimate, e.g. when the initial estimate is significantly offset. Fig. 2.3 visualizes the discrepancy between the object model and a gathered depth observation.

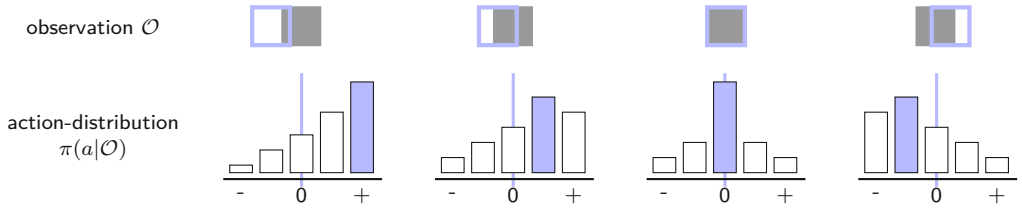

Figure 2.4: **Correspondence of Refinement Transformation and Predicted Action-Distribution.** On the top, misaligned sources (light-blue) and targets (gray) are shown to form the current observation $\mathcal{O}$. The bottom visualizes the corresponding action-distribution $\pi(a|\mathcal{O})$. The action-distribution classifies the offset between source and target into discretized misalignment bins. For example, a large offset results in a large action-step and similarily for smaller offsets. The *zero-action* indicates alignment between source and target. This figure has been adapted from [BPV21] but is self-drawn.

2.3.1 Reinforced pose refinement

RL in general is a machine learning technique, where an agent is learning by making decisions to interact with an environment. By receiving constant feedback about the impact of its own actions through a *reward-system*, the agent eventually updates its decision-making paradigm, i.e. its *policy*, aiming to maximize the received reward.
Iterative registration, as defined in Eq. 2.4, predicts n relative transformations to realign source and target, also referred to as a *refinement trajectory*. Reinforced pose refinement aims to determine this *refinement trajectory* by iteratively selecting the most probable refinement action a_i from a policy π to achieve closer alignment of the current source X_i' and target Y in each iteration i, based on the current observation $\mathcal{O} = (X_i', Y)$. To reinforce closer alignment over multiple refinement steps, the reward system must encourage converging and prune diverging steps. Typically, the action space is defined by a number of discretized steps to manipulate points on one of the six degrees of freedom, i.e. action-distributions to select the most suitable actions from are predicted for each axis of translation and rotation respectively.
To summarize, the goal of reinforced pose refinement is to predict an action-distribution $\pi(a|\mathcal{O})$ for a given observation $\mathcal{O}(X', Y)$ to select the most suitable action to realign source X' with target Y in a minimum number of steps. Fig. 2.4 visualizes the correspondence of the action-distribution with varying offsets of source and target.

2.4 Object Tracking

6D object tracking extends single-frame pose estimation with respect to the time axis for a continuous prediction of object poses by taking priors into account. In dynamic environments, frequent updates of object poses enable applications in robotic manipulation [MKR+19, KMI+18, THDT21], augmented reality [MUS16] or autonomous driving [ESLG10, MSL23].

In a dynamic hand-over scenario, single-shot pose estimation fails when objects are constantly manipulated due to being held by human hands. Additionally, if the camera is mounted to the robot's end effector, following the target requires frequent pose updates to align with the camera frame.
Interestingly, the more sophisticated task of tracking hands, which involves tracking multiple hand joints, enables the efficient creation of demonstration data used to train a robot policy [HJN+20]. Similarly, high-frequency object poses are a prerequisite for augmented reality to (re-)align virtual objects with their anchor in the real world. In that field, real-time 6D object tracking makes the transition from reality to the virtual world more seamless.
As previously mentioned, heavy occlusion poses a threat to the efficacy of object tracking methods. The insufficient amount of inliers results in less robust correspondences and therefore missmatches, ultimately leading to increased pose errors. Addressing this issue, a reinitialization strategy is a crucial component for any object tracking approach. Such a strategy enables tracking methods to recover even when an object is entirely occluded, e.g. by hands. Since annotated data is not available during test time, either the behavior of the method itself needs to be exploited or additional plausibility checks, such as rendering-based verification [BPV20], are implemented.

2.5 Task Definition

2.5.1 Point Cloud Registration

Mathematically speaking, the observed source X' is offset from the target Y by an unknown rigid transformation $T' = [R' \in SO(3), t' \in \mathbb{R}^3]$. Moreover, the observed source X' is defined as

$$X' = T' \otimes X \tag{2.1}$$

Similarly, the aligned source X is yielded by transforming the observed source X' with the transformation T.

$$X = T \otimes X' \tag{2.2}$$

As a result, to realign both observed source X' and target Y, one needs to find the transformation $T = T'^{-1}$. One-shot registration [WS19] refers to this process being executed in a single step. Otherwise, iterative registration retrieves the corresponding transformation T by carrying out n updates. Each update i transforms the observed source X'_{i-1} by a transformation T'_i.

$$X_i = T'_i \otimes X'_{i-1} \tag{2.3}$$

$$X_n = \underbrace{T'_n \otimes ... \otimes T'_1}_{T'} \otimes X'_0 = X \tag{2.4}$$

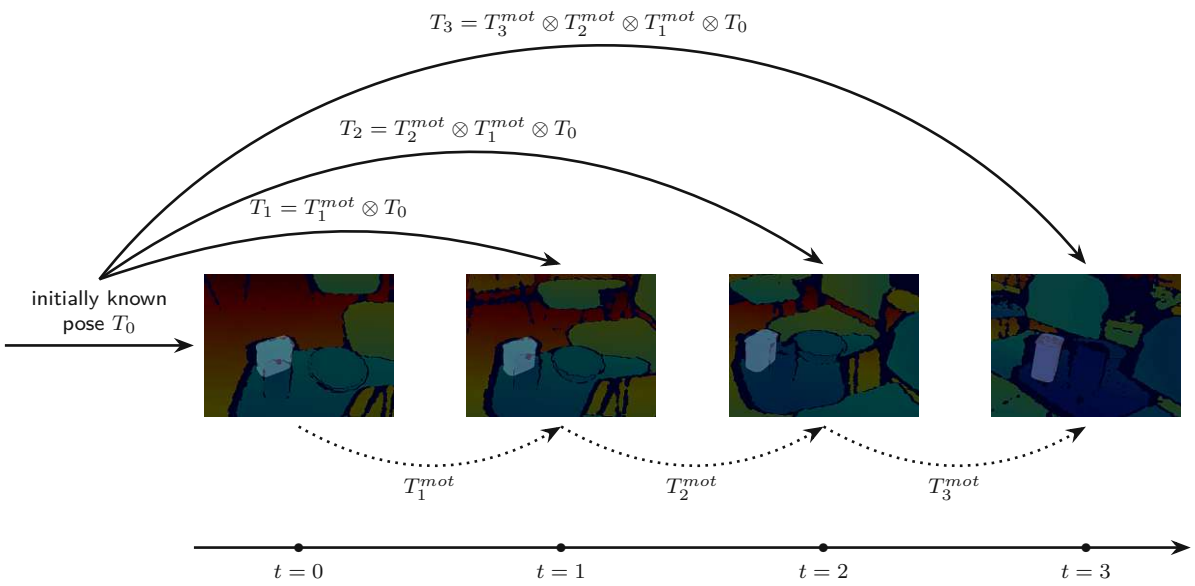

Figure 2.5: **Relative vs. Absolute Pose Transformations:** Relative transformations (dotted lines) indicate the object motion between frames. Absolute poses describe the 3D translation and 3D rotation of the object with respect to the camera position. Starting off from an initially known pose T_0, the absolute object pose T_τ for time step τ is computed by a concatenation of all relative transformations in the interval $t = (0, \tau]$.

In general, dividing the registration process into multiple substeps *(iterative registration)* may yield a higher accuracy than through one-shot registration due to the potential to correct initial errors iteratively. Still, an error-afflicted estimate in a single step complicates reaching alignment in subsequent steps and may even result in iterative divergence in the worst case.

2.5.2 Object Tracking

Let us assume, that the initial object pose T_0 is given with respect to the camera position. To align with the next frame, the absolute object pose T_1 for the subsequent timestep $\tau = 1$ is computed by finding a relative rigid transformation T_1^{mot} describing the motion between frames, e.g. via aligning both depth observations with a point cloud registration algorithm. More generally, the pose T_τ for time step τ is computed by concatenating all previous relative transformations T_t^{mot} for $t \in (0, \tau]$ to transform the initial pose T_0 as visualized in Fig. 2.5.

$$T_\tau = T_\tau^{mot} \otimes T_{\tau-1}^{mot} \otimes ... \otimes T_1^{mot} \otimes T_0 \tag{2.5}$$

CHAPTER 3

Related Work

Building on an exhaustive literature review, we present how the concepts introduced in Chap. 2 are approached by recent methods. Specifically, we put increased attention on the concepts of pose refinement and object tracking, as they build the core for this thesis.

3.1 Point Cloud Registration

A substantial contribution in this field was the ICP algorithm [BM92], as explained in Sec. 2.1. Go-ICP [YlCJ16] uses a Branch-and-Bound algorithm to determine upper and lower bounds for the error metric, ensuring the classical ICP algorithm to not get stuck in a local optimum. As implemented in NVIDIA's Isaac SDK - a framework to build virtual robotic applications [MVeSTT19] - as an object pose refinement algorithm, Rusinkiewicz [Rus19] proposes an ICP-variant to leverage surface normals as an additional cue for a symmetric error function to minimize the point-to-plane error. Li et al. [LHZA22] further optimized this symmetric error function by incorporating the surface normals' rotation, resulting in improved registration accuracy with equal computational effort.
Recent years have shown a trend towards using Deep Neural Networks (DNNs) for the general task of point cloud registration. A crucial contribution for this field is PointNet [QSMG17], enabling to learn a more meaningful feature representation of the unstructured data points due to the symmetric max pooling function. The authors of PointNetLK [AGRL19] labeled PointNet [QSMG17] as a 'learnable imaging function' and paired it with a method used in image alignment, a modified variant of the Lucas & Kanade algorithm, to then be applied in iterative point cloud registration. In their work, the first stage classifies characteristics and contributes to a global feature vector to eventually compute a suitable transformation that closer aligns source and target. Building on the concepts of PointNetLK [AGRL19], the method of Bauer et al., namely ReAgent [BPV21], considers point cloud registration as a classification of misalignment and combines a simplified PointNet [QSMG17] architecture with a succeeding policy

network. A combination of IL and RL with a Chamfer-distance guided reward results in SotA accuracy. To robustify the matching process in partially overlapping point clouds, Arnold et al. [AMD21] fuses self- and cross-attention to improve matching corresponding features.

3.2 Pose Estimation

By discretizing the pose space, Kehl et al. translated pose estimation to a classification problem [KMT+17]. Inspired how humans learn properties in the real world from a 2D perception without annotations, Self6D [WML+21] utilizes differentiable rendering to contribute to a self-supervised learning framework by aiming for 1) visual and 2) geometrical alignment with the observation.
Focus was also put on making the PnP/Ransac stage differentiable for indirect methods [BR19]. However, this tradeoff causes a more complex training strategy. On the other hand, Hu et al. replaced the PnP/Ransac stage with a neural network inspired by PointNet [QSMG17] to regress the 6D pose.
The winner of the BOP challenge 2022 [Mar23], namely GDRNPP [WMTJ21, LZZ+22], fuses direct and indirect methods together into one approach. In the first step, the network takes the ROI, estimated by an object detector, as input and predicts geometric feature maps. A subsequent Patch-PnP CNN directly regresses the 6D object pose from dense correspondences based on the initially predicted geometric feature maps. In the BOP Challenge 2022 [Mar23], GDRNPP achieves the highest accuracy for RGB-only compared to all other competitors. Additionally, it also has been the fastest approach in its category.
2D object detections are of high relevance for 6D pose estimation, as they are typically used in the first stage to identify object instances. ZebraPose [SSF+22] matches a hierarchical binary encoding of the object's surface to a precomputed surface code to eventually solve for the 6D pose using a Ransac/PnP stage.

3.3 Pose Refinement

DeepIM [LWJ+18] uses a CNN to iteratively predict relative transformations by rendering the object under its currently estimated pose and matching it with the observed RGB image. To increase robustness for small objects, DeepIM operates on zoomed-in and up-sampled image patches to compare with the rendering. PoseRBPF [DMX+19] uses rao-blackwellized particle filtering in combination with an auto-encoder network to estimate the 6D object pose. Each particle corresponds to a translation hypothesis to be compared with the observation, while a rotation likelihood is computed by matching with precomputed embeddings. Treating pose refinement as an action-decision process, Busam et al. [BJN20] refine estimated poses by selecting suitable actions in each iteration for closer alignment of the RGB observation with a rendered reference. As the follow-up work for ReAgent [BPV21], SporeAgent [BPV22] incorporates physical constraints to robustify their depth-only pose refinement in multi-object scenarios such as in the YCB-V

dataset [XSNF18]. Considering non-floating and non-intersecting objects along with their symmetry axes contributes to more plausible object poses, while the approach itself only utilizes a hundredth of the available YCB-V training data. Building on the classical matching of RGB observations with its rendered counterpart, RePOSE [ILK+21] optimizes the rendering process by substituting RGB textures with a learnable feature texture to directly obtain a feature map. Especially in multi-object scenes, the achieved runtime is affected by the computational cost of feature extraction. However, the authors of RePOSE [ILK+21] achieved significantly reduced runtime as compared to other recent methods by fusing rendering and feature extraction into a single step. DeepRM [AS23] builds on the DeepIM framework and utilizes a network with recurring connections, allowing to leverage and propagate information across refinement steps. Similarly, RNNPose [XLZ+22] exploits recurrent network connections for building a correspondence field based on learned 2D/3D correspondences to predict the pose best describing the observation.

3.4 Object Tracking

From a broader perspective, all object tracking approaches solve a pose estimation/refinement problem at their core. Vice versa, algorithms solving single-frame pose estimation/refinement algorithms can be embedded into simple tracking pipelines by reusing the previous frame's estimate as initialization.

A crucial component for any object tracker is an approach to identify when tracking deteriorates. Intuitively, the pose error decreases with each iterative update, therefore the last refinement iteration ideally yields the identity matrix, i.e. does not manipulate the pose. The authors of DeepIM exploit this fact and trigger a reinitialization as soon as the average translation and rotation in the last refinement iteration over the last 10 frames exceed a certain threshold. Specifically dedicated to the tracking objective, PoseRBPF [DMX+19] incorporates a motion model to propagate particles to the current step by exploiting motion priors.

In recent years, the acquisition of synthetic datasets including annotated 6D poses through the utilization of a photorealistic renderer [DSW+19] gained significant popularity. However, transferring knowledge from a simulation to the real world remains a challenge due to uncertainties, imperfections and other influences not grasped by the simulation. Wen et al. addressed this issue in se(3)-tracknet [WMRB20] with a smart feature-encoding disentanglement technique that refines the ability to generalize to real data from simulation data (*sim2real gap*) using domain randomization. Aiming to reduce the *sim2real gap*, physical constraints are considered during the object placement, and synthetic depth frames are post-processed to mimic the artifacts caused by a real depth sensor.

The authors of ICG [SST22] fuse two probability density functions (PDFs), for color and depth information respectively. Forming the region-based PDF, color histograms distinguish the object from the background to calculate correspondence lines for the object's contour. Heavily inspired by ICP [BM92], geometric cues in the depth image are

used to contribute to the depth-based PDF. In a joint optimization, the most plausible pose is found for the current frame.
BundleTrack [WB21] enables 6D object tracking for novel objects without relying on an object model. Relative transformations are found by matching key points in consecutive frames. A keyframe memory pool stores distinct object views, allowing for global optimization of previously computed poses. Wen et al. fused the BundleTrack pipeline with a neural object field to autonomously reconstruct objects in the follow-up work BundleSDF [WTB+23]. The neural object field grabs all distinct views of the object to concurrently model the geometric shape and its texture.

CHAPTER 4

TrackAgent: Reinforced Object Tracking

Within this chapter, we explain the link from pose refinement to 6D object tracking as an introduction to the concept of our method. We start by giving an overview of our TrackAgent pipeline to eventually discuss our contributions. To combine *frame-to-frame registration* with *frame-to-model refinement*, we present three fusion variants with varying complexity. Moreover, we extend the reward system of a single-frame pose refinement method [BPV22] to additionally achieve closer alignment across longer time horizons across frames. Exploiting information within our pipeline for an autonomous reinitialization heuristic concludes this chapter.

4.1 From Pose Refinement to 6D Object Tracking

To achieve accurate results in the field of pose estimation, a typical procedure is to utilize a pose estimation method for yielding an approximate estimate, to be refined by a pose refinement method. Such a procedure stems from the characteristics of both a pose estimation and pose refinement method.
A pose estimator has to identify and localize object entities in the entire image region, resulting in a huge set of potential object poses and consequently difficult to solve accurately. Moreover, changing illumination, occlusion, and noise further complicate obtaining precise estimates.
In contrast, the initial guess by the pose estimator constrains the pose space and enables refinement methods to focus on a specific image region, e.g. by rendering the object model under the initial estimate to yield a 2D mask. Even though outliers are still likely to be present, their ratio compared to inliers is significantly decreased in contrast to operating on the entire image, as pose estimation must do. Consequently, pose refinement leverages the error-afflicted estimate for a more finegrained alignment process to predict

Figure 4.1: **Refining Object Poses from Approximate Estimates:** This graphic illustrates the estimates generated by a pose estimator [XSNF18] (green contours) and refined by a pose refinement method [LWJ+18] (red contours). Without the additional refinement step, applications like grasping would fail. For our tracking approach, we obtain a rough estimate in the current frame by leveraging the previously refined pose, resulting in less computational effort by exploiting prior information. Taken from [LWJ+18].

precise object poses.
As illustrated in Fig. 4.1, an approximate estimate serves as a meaningful cue on the image region to be considered, but the accuracy is insufficient for applications like grasping and therefore requires an additional processing step. Similarly to refining approximate pose estimations by a pose estimator, we consider the refined pose of the previous frame as a sufficiently accurate initialization for the current frame, eventually to be refined to align the object pose with the current frame. Such an approach enables us to 1) invoke the pose estimator only when inevitable, i.e. for (re-)initialization, resulting in lower computational effort and 2) induce temporal correspondences by reusing the previous pose.

4.2 TrackAgent - Contributions

This thesis proposes to embed an RL agent used for pose refinement into a tracking pipeline. By reusing previous pose estimates as initialization for the current frame, the tracking pipeline builds on previous computations while the pose estimator is only required for (re-)initializations. A hybrid architecture for the RL agent fuses *frame-to-frame registration* and *frame-to-model refinement* to approach 6D object tracking as a depth-based point cloud alignment problem across frames. The joint solving of both subtasks stems from its complementary nature. Temporal and spatial coherence is ensured by registering

consecutive depth observations from the same (noisy) distribution via *frame-to-frame registration* . Simultaneously, the *frame-to-model refinement* subtask incorporates an object model as a reference, reducing the impact of accumulating error for our method. By acting as a guidance for the alignment process, the object model additionally robustifies the *frame-to-frame registration* subtask to low-overlap of corresponding observations, enabling tracking even when establishing correspondences across frames fails.
Recently, iterative pose refinement methods [BPV22] leverage RL to reinforce closer alignment over multiple steps for a single frame. For the tracking objective, it is desired to maintain close alignment over multiple frames, not only steps in a single frame. For our reward system, we achieve such a behavior via a joint consideration of the obtained alignment 1) over multiple refinement steps in a single frame and 2) across multiple frames by considering the final estimates per frame. While this allows our method to keep track of the pose across longer time periods, maintaining track of the corresponding image segments is also crucial to (depth-based) object tracking. We yield a 2D mask for the object by rendering the object model under its previously estimated pose, allowing us to segment the observation as a point cloud from the current depth frame. Importantly, this mask is predestined to include outliers, as it only leverages the refined pose from the previous frame instead of the current one. To address this issue, we incorporate strategies in our method to classify and reject outliers, resulting in a mitigated influence of outliers on the overall alignment process.
Since object tracking may deteriorate in challenging scenarios, e.g. due to heavy occlusion or inaccurate segmentation, our method considers its own uncertainty to autonomously reinitialize implausible poses. We control the number of reinitializations with the achieved tracking accuracy by a joint consideration of 1) the geometric uncertainty stemming from a comparison of the observed depth with a rendered reference and 2) the agent's own uncertainty indicated by the predicted action-distribution

4.3 Defining the Subtasks

We define the depth frame at time step t as D_t. We retrieve the object observation as a 3D point cloud P_t from the depth frame D_t using the rendered mask M_t and the cam intrinsics. As a reference, the object model O is uniformly sampled under the previously known pose T_t.
The *frame-to-frame registration* subtask is defined as realigning consecutive observations, i.e. align P_t with P_{t-1}, by finding a suitable transformation T_t^{reg}. As defined in Eq. 2.5, the object's pose T_τ for time step τ is computed by a concatenation of all prior relative transformations over $t \in (0, \tau]$ for a given initial pose T_0. Analogous for *frame-to-model refinement*, the goal is to find a transformation T_t^{ref} to realign the observation P_t with the object model sampled under the previously refined pose $O_t = T_{t-1} \otimes O_{t-1}$.
By fusing these two, our goal is to compute a single transformation T_t^{mot}, that simultaneously fulfills the objectives of either subtask, i.e. $T_t^{mot} = T_t^{ref} = T_t^{reg}$. We characterize this as a *guided registration process*, where realigning consecutive observations is guided by, and concurrently adjusted to align with, the object model as a reference.

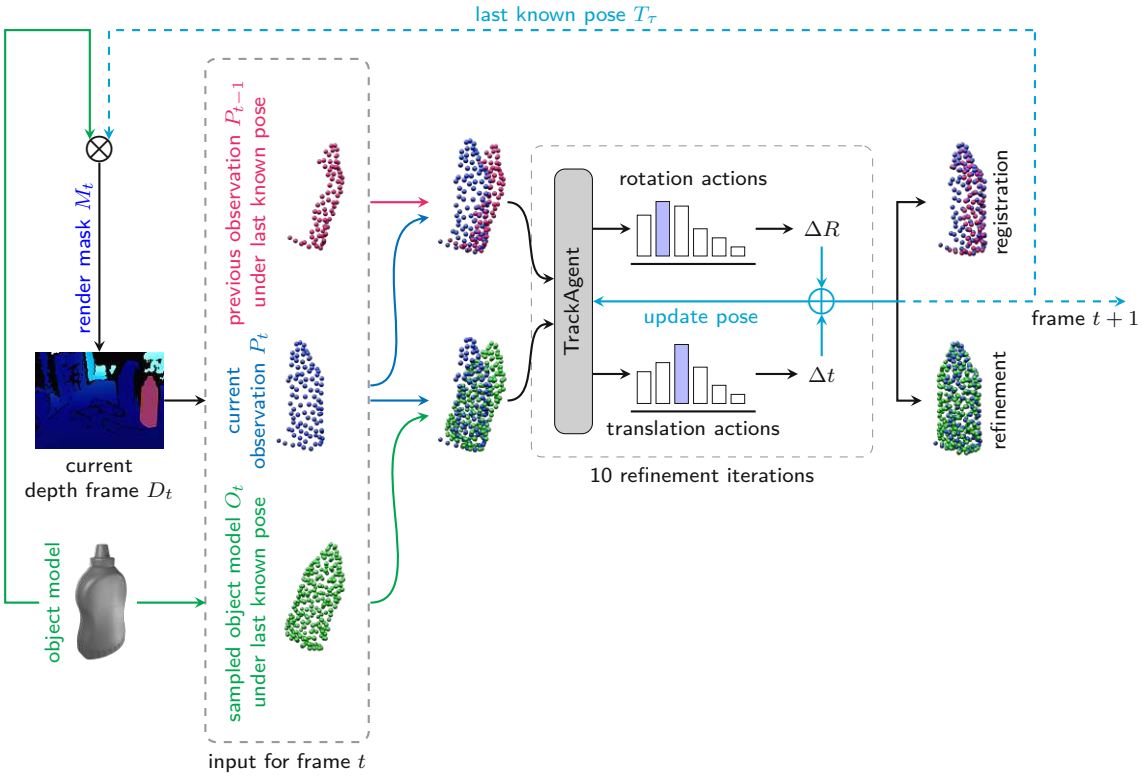

Figure 4.2: **Overview of the TrackAgent-Pipeline:** Using the last known pose and the object model enables us to render a mask to obtain the point cloud P_t from the current depth frame. The subsampled object model O_t and previous depth observation P_{t-1} are transformed with the last refined pose. In a joint task, TrackAgent iteratively finds a unified transformation to closer align both observations (*frame-to-frame registration*) as well as the current observation with the object model (*frame-to-model refinement*). The current observation P_t is forwarded to the next frame $t+1$ to align with the observation in the next frame.

4.4 The Tracking Pipeline

We initialize our pipeline, for the sake of comparison, with the ground-truth pose and a 2D mask segmenting the object region. In every frame D_t, we obtain a 2D mask M_t by rendering the object model under the last known pose, enabling us to keep track of the corresponding segment in the depth frame to extract a point cloud P_t. TrackAgent iteratively predicts actions for translation and rotation to closer align the object pose with the current frame by finding a joint alignment of the current observation P_t with 1) the previous observation P_{t-1} (*frame-to-frame registration*) and 2) the object model O_t (*frame-to-model refinement*), both sampled under the last known pose T_τ. As an additional cue, TrackAgent takes the object category as input. After 10 iterations, our

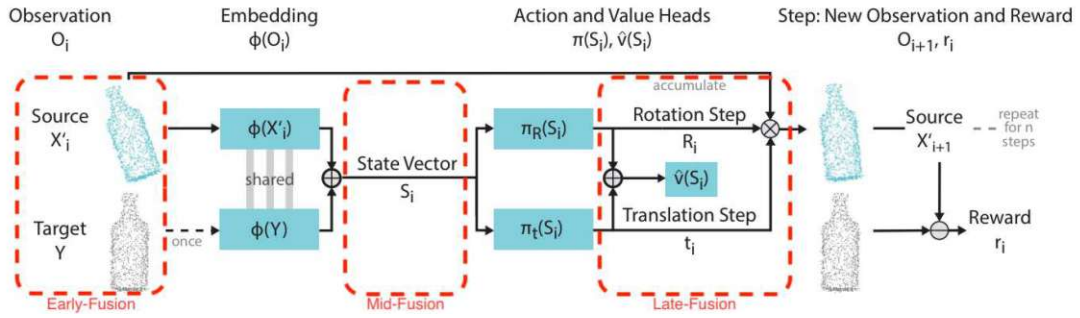

Figure 4.3: **Network-Architecture of ReAgent:** Global features in the point clouds are classified using two encoders with shared weights in the embedding stage. Concatenating the embeddings of both point clouds contributes to the current state vector to be processed by a subsequent policy network to predict action-distributions for rotation and translation. Our proposed key points for fusing *frame-to-model refinement* and *frame-to-frame registration* are highlighted in red. The illustration is taken from [BPV21].

final result, the relative transformation T_t^{mot}, is derived from the predicted actions, allowing to retrieve the absolute object pose T_τ with respect to the camera. The object pose is reused as an initialization in the next frame. Moreover, TrackAgent utilizes the manipulated current observation P_t for the *frame-to-frame registration* objective in the next frame. An overview of our pipeline is illustrated in Fig 4.2.
Noteworthy, TrackAgent is self-supervised and considers its own uncertainty along with a geometrical uncertainty to identify, when tracking is deteriorating.

4.5 Fusion Variants

Our novel network architecture stems from recent methods that consider point cloud registration as an RL task, previously applied to point cloud registration [BPV21] and reinforced pose refinement [BPV22]. The architecture in both methods may be grouped into two separate stages. First, an embedding stage uses two PointNet-alike encoders [QSMG17] to classify global features per point cloud for a more robust data representation. Concatenating both global feature vectors contributes to the state vector. The subsequent second stage, the policy network, processes the state vector and predicts two separate action-distributions to pick the most favorable actions for translation and rotation respectively, resulting in closer alignment of source and target.
Within this thesis, we are extending this two-stage pipeline to incorporate an additional point cloud as input to simultaneously solve two point cloud alignment tasks. For the fusion of both subtasks, we consider potential key points 1) at the input side (*early-fusion*), 2) after the embedding stage (*mid-fusion*) and 3) after the policy embeddings (*late-fusion*). Serving as a baseline for our fusion-variants, Fig. 4.3 illustrates the architecture of ReAgent [BPV21], along with our proposed key points.

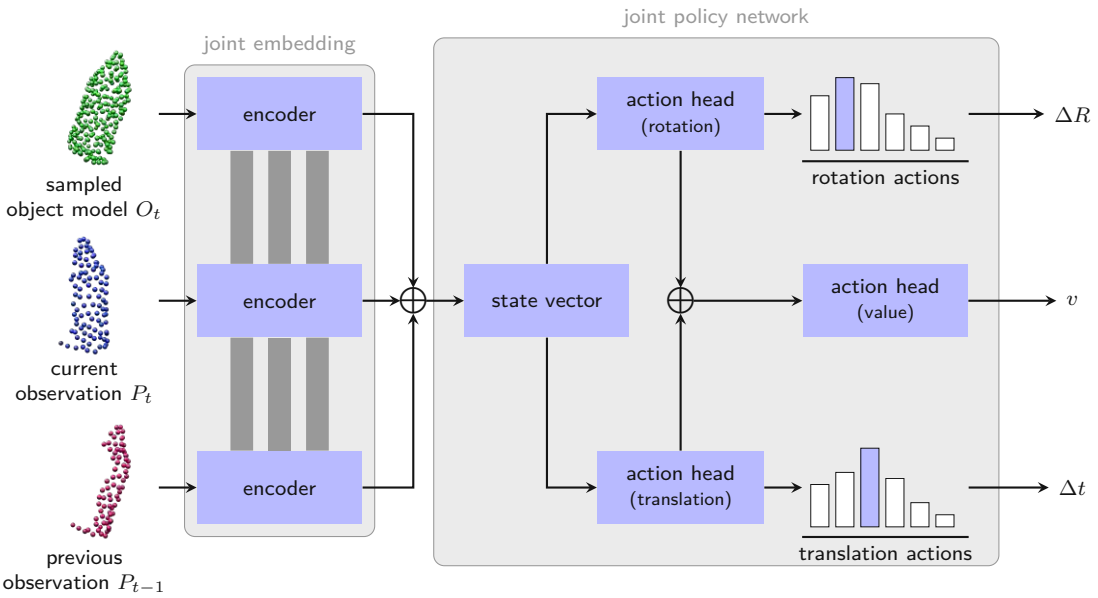

Figure 4.4: **Early-Fusion Variant:** We fuse both subtasks at the input side by incorporating an additional encoder, allowing the features of the previous observation P_{t-1} to contribute to the state vector. Sharing weights across all encoders jointly encodes the *frame-to-model refinement* and *frame-to-frame registration* subtasks, however, the ability for finegrained optimization to either subtask is limited.

Details of our fusion variants are listed below:

1. **Early-Fusion:** The early-fusion variant fuses the subtasks at the input side and adds another encoder to classify features in the previous observation P_{t-1}. Importantly, all encoders share the same weights. As a result, the early-fusion variant has to compromise the encoder weights to ensure compatibility for both subtasks, rather than finetuning them specifically to either of the subtask's objectives. Besides that, all global feature vectors are concatenated to build the state vector of our joint solving and the subsequent policy network remains unchanged. We illustrate the network in Fig. 4.4.

2. **Mid-Fusion:** Introducing two distinct encoder branches allows the mid-fusion variant to optimize the weights for both *frame-to-model refinement* and *frame-to-frame registration*, as the weights are not shared across all encoders. The subtasks are fused after the encoding stage with both global feature vectors contributing to an improved state representation. We adapt the baseline policy network to process the concatenated state vector for a unified output. Fig. 4.5 visualizes the mid-fusion architecture.

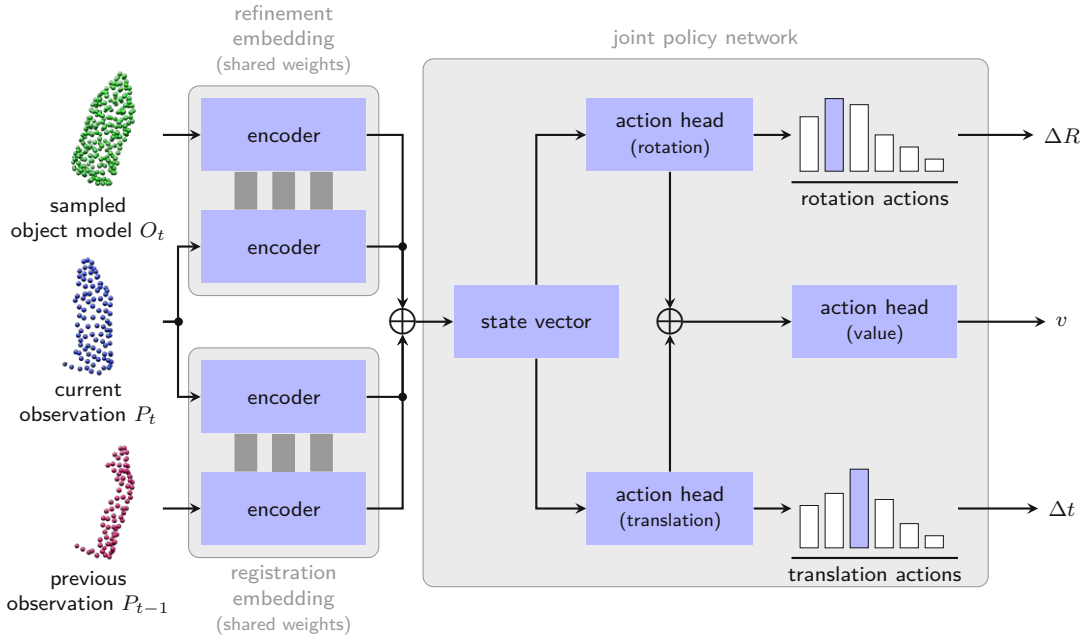

Figure 4.5: **Mid-Fusion Variant:** We circumvent limitations in the encoding stage by introducing two separate branches, each encoding one of the subtasks. This enables the mid-fusion variant to adjust the encoder weights to either of the subtasks. Concatenating the resulting state vectors of both *frame-to-model refinement* and *frame-to-frame registration* contributes to a more meaningful state representation for the policy network.

3. **Late-Fusion:** The late-fusion variant builds on the encoding stage of the mid-fusion variant, but, instead of fusing the global features, the state vectors are processed by two individual policy networks dedicated to *frame-to-frame registration* and *frame-to-model refinement* respectively. Importantly, the components of the policy network, i.e. the action-head, had to be split up, resulting in generating policy embeddings via embedding-heads in the policy networks followed by merge-heads to process both subtasks' embeddings and predicting a joint action-distribution. We illustrate the late-fusion variant in Fig. 4.6.

4.6 Learning 6D Object Trajectories from Scratch

The problem at the core of our method, point cloud registration, is difficult to solve by leveraging RL and may not converge towards a global optimum, given the large action-space. Moreover, incorporating an additional time axis for the tracking objective and jointly learning the feature representation contributes to an even more complex task to solve.
Recent methods [BPV21, BPV22] kickstart RL by additionally infusing IL to learn an

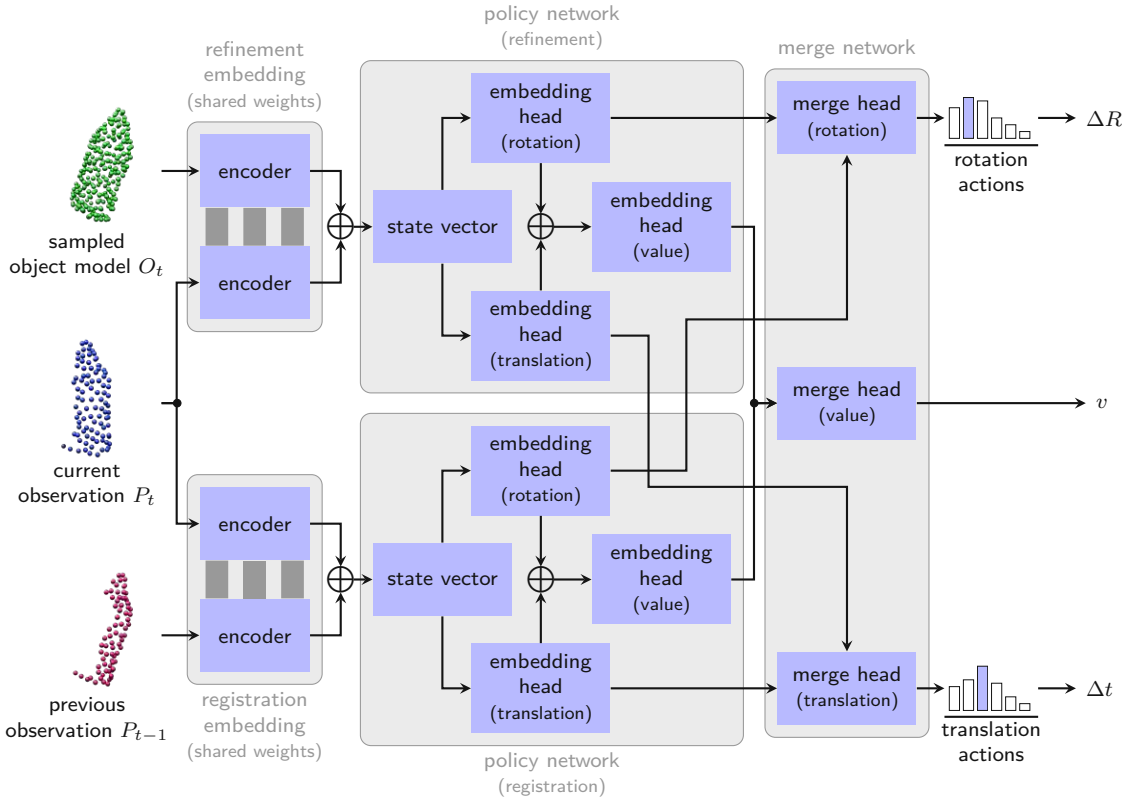

Figure 4.6: **Late-Fusion Variant:** We further extend the encoding stage of the mid-fusion variant with individual policy networks to predict policy embeddings per subtask. A subsequent merge stage fuses *frame-to-frame registration* and *frame-to-model refinement* to predict unified action-distributions for rotation and translation respectively. The late-fusion variant, our most complex strategy, may be interpreted as cloning the baseline network [BPV22] for either subtask to eventually merge results at the end for a unified output.

optimal policy. IL aims to learn from demonstration, i.e. replicate the policy of an expert that has knowledge about the current, best action.
Conversely, solely learning from an expert policy limits the agent, as actions diverging from the expert's reference are pruned. Without additional objectives, the agent is unable to outperform the expert.
For our object tracking method, we adapt the learning scheme from previous methods [BPV22], which leverages a combination of RL and IL. Dedicated to the tracking objective, we extend this reward system to simultaneously reinforce closer alignment 1) over steps in a single frame *as well as* 2) over a longer time period, i.e. multiple frames.

4.6.1 Learning from Demonstration (Imitation Learning)

In IL, feedback of an expert may be received via algorithmic solutions that, for example, exploit annotation information or via human demonstration. One of its variants is Behavioral Cloning (BC), where, in every single step, the expert feedback serves as a reference, similar to ground-truth labels in supervised learning.
The expert policy in [BPV22] leverages the annotated ground-truth object poses to return a transformation, that monotonously decreases the error in every step. Previous methods [BPV21] were challenged by symmetry axes, as they introduce multiple indistinguishable views for certain objects. For example, a cylindric object may be rotated arbitrarily around its central symmetry axis, while consistently yielding a plausible pose for our depth-based approach. As a consequence, a symmetry-aware expert, as introduced in [BPV22], is crucial for our object tracking approach to guide the alignment process towards the *closest*, plausible pose to not cause confusion for the agent.

4.6.2 Reinforcement Learning

Instead of utilizing some form of expert knowledge, RL defines the global objective via a reward-function that evaluates the efficacy of the current decision. Gathering the chosen actions and their outcome enables to update the agent's policy, aiming to maximize the reward. This may be compared to the agent playing a game, with the constant goal of increasing the score by learning from previous decisions and their efficacy.
In single-frame pose refinement, the main objective is to closer align the point clouds in each step, with the overall goal of computing the relative transformation contributing to an accurate pose estimate. Presented in [BPV21], Bauer et al. reinforced such a behavior via a consideration of the mean Chamfer Distance d_i from the currently observed source $X_i^{'}$ and the ground-truth source X, given the current refinement iteration i. The reward-function, denoted as r, is defined as follows:

$$r = \begin{cases} +0.5, & d_i < d_{i-1}, \\ -0.1, & d_i = d_{i-1}, \\ -0.6, & d_i > d_{i-1} \end{cases} \tag{4.1}$$

Accordingly, we define the Chamfer Distance d as denoted below:

$$d(X, Y) = \frac{1}{|X|} \sum_{x \in X} \min_{y \in Y} ||x - y|| \tag{4.2}$$

The reward r encourages steps yielding a closer alignment ($d_i < d_{i-1}$) and prunes diverging steps ($d_i > d_{i-1}$). Moreover, pausing steps ($d_i = d_{i-1}$) prolong the alignment and consequently recieve, compared to diverging steps, a small penalty. To avoid oscillation, the penalty for diverging steps is set to be larger than the reward for converging steps, as this may be exploited by the agent to boost the achieved reward at the initial stages of the training.
Even though close alignment is reinforced in individual frames, pose errors may accumulate

Figure 4.7: **Single-Frame Perspective and Multi-Frame Perspective for Reward Computation:** Single-frame pose refinement methods [BPV22] encourage closer alignment over multiple refinement steps. Similarly, we extend this approach to be suitable for 6D object tracking by additionally considering and maintaining the alignment across multiple frames.

over longer time horizons across frames during tracking, thereby complicating future predictions. As illustrated in Fig. 4.7, we simultaneously address 1) achieving closer alignment in a single frame, as defined in Eq. 4.1, along with 2) maintaining close alignment across sequences of frames. We achieve the latter via reusing the same reward system as defined in Eq. 4.1, but, instead of considering a sequence of steps in a single frame, we gather the final steps across frames. Intuitively, we want to encourage estimates, that prove themselves to be useful as initialization for the next frame. Vice versa, estimates leading to deterioration in tracking are pruned. Reusing the same reward also favors the simplicity of our framework, as incorporating this additional time axis only requires an additional buffer.

4.6.3 Fusing IL and RL using Proximal Policy Optimization

For our tracking framework, we rely on the findings of previous works [BPV21, BPV22] and leverage Proximal Policy Optimization (PPO) [SWD+17] to fuse IL, based on a symmetry-aware expert policy, with RL using a Chamfer Distance guided reward that simultaneously encourages close alignment over refinement steps in a single frame and over a wider time horizon across frames.
By solely learning from the expert policy, the agent is able to accurately (re-)align point clouds. However, significant changes to this policy caused by RL may cause deteriorating efficacy, therefore we desire to stick close to the policy learned by IL. PPO enables such a behavior by truncating the policy changes in each update, allowing slight modifications guided by the RL objective.
We adapt the algorithm presented in [BPV21] to be suitable for our tracking approach. As listed in Alg. 4.1, we use two distinct replay buffers during training to collect samples of 1) refinement trajectories in a single frame and 2) object pose trajectories across multiple frames. These trajectories are gathered by following the agent's predictions and consequently increase in accuracy as the policy improves. The collected samples in those buffers are then processed separately by Alg. 4.2 to optimize the policy.
We compute a cross-entropy loss of the predicted action-distribution with respect to the ground-truth actions by the expert. For the PPO-loss, we rely on the formulation in [BPV21]. Bauer et al. pair the PPO definition from [SWD+17] with an entropy term to motivate exploration. The agent's estimated value together with Generalized Advantage Computation (GAE) contributes to calculating the advantage $\hat{A}$, as used for the PPO-loss.

4.7 Keeping Track of the Object Region

We obtain a 2D mask by rendering the object's depth using the object model and its last estimated pose. However, as the rest of the scene is not taken into account, this mask is insensitive to occlusion by other objects. Moreover, the relative transformation to align with the current frame is not known yet, forcing us to base our mask on the pose of the previous frame and consequently making our mask prone to minor spatial misalignment. While the influence of outliers is limited for the *frame-to-frame registration* objective, as it aligns observations with similar imperfections, the efficacy of the *frame-to-model refinement* objective may be reduced. In general, the presence of outliers cannot be assumed to be absent in object tracking. As a consequence, an approach to detect and reject outliers is desired to not skew the alignment process.
We aim to mitigate their influence by either 1) using a *geometry-guided mask filtering* to obtain a visibility mask or 2) incorporating a *segmentation branch*, as first introduced in [BPV22], that classifies inliers and outliers within the network by learning the properties of visible segments. We evaluate and discuss the results of our segmentation strategies in Sec. 5.4.

Algorithm 4.1: Gathering single-frame and multi-frame samples. The code is adapted from [BPV21], but modified to the needs of our tracking approach. The changes include looping over consecutive frames, operating with three point clouds as input and collecting multi-frame samples.

```
# initial frames to start tracking from are defined by the dataloader
forall initial frames do
    # iterate all initial frames
    for m consecutive frames do
        # iterate consecutive frames for tracking
        gather source P_t from current frame and target O_t
        if initial frame then
            # initially, previous source equals current source
            previous source P_{t-1} set to source P_t
        end
        for n refinement steps do
            observation O = (P_t, P_{t-1}, O_t)
            # find refinement transformation iteratively
            agent predicts π(O) and value v
            select action a from predicted policy π(O)
            take action a, recieve reward r and next P'_t
            # collect single-frame trajectories
            add sample to step buffer b_steps
            if final iteration then
                # collect multi-frame trajectories in final refinement iteration
                add sample to frame buffer b_frames
            end
            update source P_t = P'_t
        end
        update agent's policy from b_steps using Alg. 4.2
    end
    update agent's policy from b_frames using Alg. 4.2
end
```

Algorithm 4.2: Update agent's policy using the collected samples by simultaneously using IL and RL. Code taken from [BPV21]

```
compute return R, shuffle buffer b
forall samples in buffer b do
    agent predicts new policy π'(O) and value v'
    # imitate expert
    expert predicts action a*
    compute cross-entropy loss l_IL of π'(O) and a*
    # reinforce
    compute PPO loss l_RL of π'(O) and π(O)
    # update agent
    l = l_IL + α * l_RL
    backpropagate combined loss l
end
```

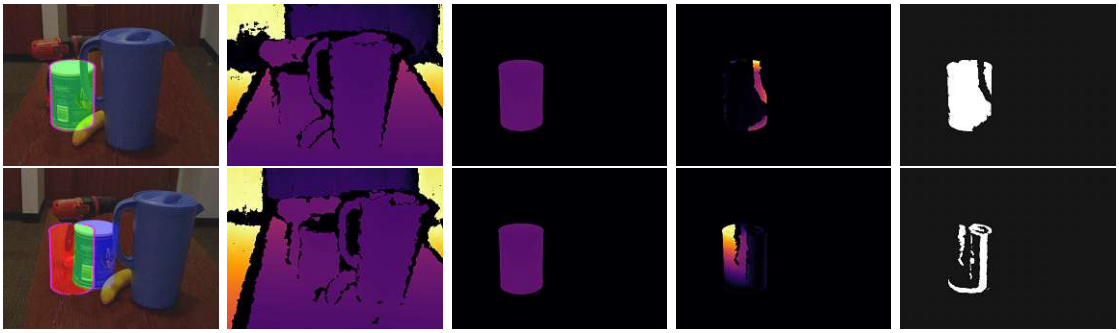

Figure 4.8: **Computation of the Visibility Mask under an Accurate and Error-Afflicted Pose:** *From Left to Right*: RGB frame with estimated pose, depth frame, object model rendered under estimated pose, depth differences, resulting visibility mask by thresholding depth differences. *From Top to Bottom*: Calculating the visibility mask under an accurate and error-afflicted pose estimation.

4.7.1 Geometry-guided mask filtering

Our rendering of the mask is, in fact, the synthetic depth of the object under the estimated pose. We approach our *geometry-guided mask filtering* via a comparison of the rendered object depth with the corresponding regions in the observed depth frame. Outliers are rejected pixelwise as soon as the difference to the reference exceeds a threshold τ_{seg}, yielding a visibility mask of the object.
Let us assume a perfect pose estimate, then depth differences in visible areas are significantly smaller than in occluded areas. We empirically determined the threshold $\tau_{seg} = 12mm$ to be the most suitable for distinguishing visible and occluded areas, e.g. caused by fingers or other objects. Importantly, this threshold includes a small safety margin to not prune the mask due to minor pose errors. Following that procedure enables us to leverage the geometric constraints implicitly given by our mask propagation, allowing us to filter implausible depth pixels prior to the generation of the point cloud. Fig. 4.8 visualizes filtering the rendered mask to obtain the visibility mask for both a precise and imprecise pose estimate.

4.7.2 PointNet-based Segmentation

As presented in SporeAgent [BPV22], Bauer et al. consider outlier pruning as a task for the embedding stage. By additionally incorporating four 1D-convolution layers, outlier points are classified and eventually rejected in the max pooling function to not contribute to the state embedding. To learn the correct classification of inliers and outliers, the ground-truth masks are used as labels to compute the cross-entropy loss.
We adopt this segmentation branch for our network from [BPV22]. However, as SporeAgent is a pose refinement method, the object model as target does not contain outliers per definition and therefore is not embedded using the segmentation branch. In contrast, our work simultaneously contains the object model and the previous observation as target.

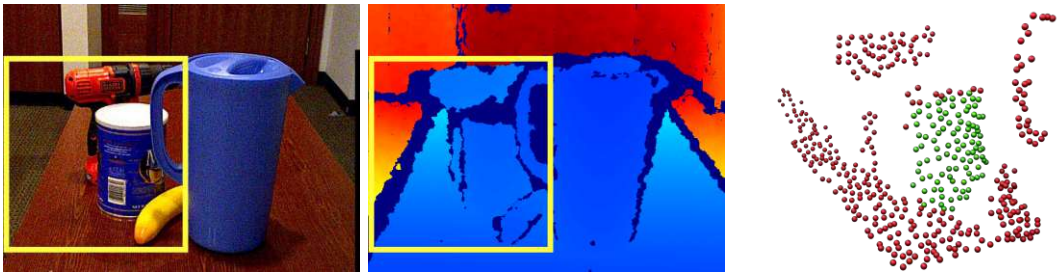

Figure 4.9: **Depth-Based Segmentation within the Encoder:** *From Left to Right:* RGB frame, depth frame, segmented point cloud. The segmentation branch is able to classify points in the point cloud and distinguish the object of interest (green) from background and other objects (red), shown for the *master chef can* object from the YCB-V dataset [XSNF18]. For the purpose of demonstration, an enlarged bounding box has been used as mask (marked in yellow) for segmenting the point cloud from the depth frame.

Consequently, to reject outliers in the previous observation as well, our network employs the segmentation branch in all encoders, except the one dedicated to the object model. We illustrate in Fig. 4.9, how the segmentation branch is able to distinguish the object from background and other objects, solely utilizing the point cloud without any color information. For the sake of demonstration, we used an enlarged bounding box as a mask to segment from the depth frame.

4.8 Keeping Track of Self-Supervised Uncertainty

In 6D object tracking, keeping track of the object poses is the main goal. Conversely, identifying when tracking deteriorates also contributes to keeping track of the poses, as it revives the tracking process when it is unable to recover, e.g. due to heavy occlusion. Especially when operating in the real world, the sensed data may be far from what the agent has seen during training, resulting in uncertainty and consequently reduced tracking accuracy.
We leverage implicit information derived from our previously computed results to contribute to two distinct uncertainty metrics, allowing our TrackAgent to supervise itself and autonomously trigger reinitializations. For our uncertainty metrics, we consider 1) the geometrical misalignment of the rendered reference, as explained in Sec. 4.7.1, fused with 2) interpreting the agent's stepsize when alignment is expected.

4.8.1 Defining Geometrical Uncertainty

Refining the mask, as explained in Sec. 4.7.1, is not only beneficial to remove outliers, but is considered by us as a geometrical uncertainty cue. Illustrated in Fig. 4.8, the fraction of mask inliers, i.e. the *visibility* of the object, significantly drops under imprecise pose estimates due to the geometrical mismatch of rendered and observed depth. However, the

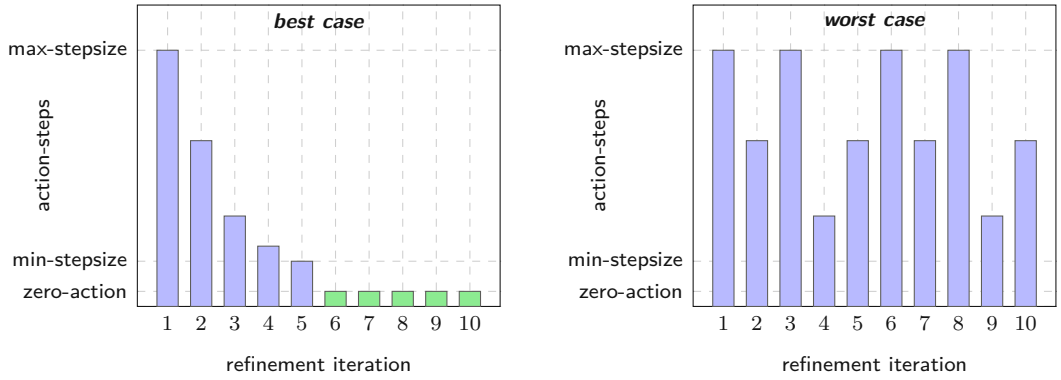

Figure 4.10: **Expected Action-Steps under Best- and Worst-Case across Refinement Iterations:** Intuitively, the misalignment and consequently the predicted action-steps decrease in each iteration until alignment is reached and therefore the zero-action is predicted (left). Conversely, losing track implies error-afflicted segmentation. As a consequence, our agent is unable to establish correspondences and tends to predict larger stepsizes, as TrackAgent is confident about not being aligned (right).

same effect is caused when objects are heavily occluded, therefore solely relying on this metric may be susceptible to falsely triggering reinitializations caused by heavy occlusion.

4.8.2 Defining Action-Based Uncertainty

Alternatively, the agent's prediction serves as a meaningful insight to the agent's own uncertainty. Our agent categorizes misalignment into *action-steps* to iteratively align the point clouds more closely. Ideally, the selected action-step is reduced in each iteration until alignment is reached. Therefore, our agent is supposed to predict a so called *zero-action* in the last refinement iteration, i.e. the point clouds are aligned.
However, if track is lost, the rendered mask will barely overlap with the object of interest, resulting in compromised feature detection that hinders an effective alignment process due to the lack of corresponding points. Rather than predicting the zero-action in the final iteration, we observed a likelihood of the agent predicting larger step-sizes in such cases.
We interpret the predicted misalignment, i.e. the selected action-step, in the last iteration over multiple frames as a metric for the agent's own uncertainty. By averaging the predicted action-steps over multiple frames, we are able to trigger a reinitialization as soon as a certain threshold is exceeded and simultaneously robustify this metric against anomalies in a single frame. However, non-zero action-steps in the last refinement iteration may also be caused by large offsets between two frames, e.g. due to fast motion, or when data during inference differs too much from the training distribution, even though the pose estimate is accurate.
Nevertheless, non-zero action-steps in the final iteration may also be caused by large offsets between two frames, e.g. due to fast motion. Moreover, discrepancies between the

inference- and training data contribute to large action-steps, even though the estimated pose is approximately accurate.

4.8.3 Fusing our uncertainty metrics

To circumvent the weakness of either uncertainty metric, a joint consideration of both seems the most logical to us for balancing the tracking accuracy with the number of carried out reinitializations.
Let us assume, that the *geometry-guided mask filtering* fails, then considering the action-steps actively suppresses a reinitialization and makes our TrackAgent less sensitive to false triggers under heavy occlusion. Vice versa, when the action-based uncertainty indicates that track is lost, we rely on our visibility mask to be robust towards large offsets, such as caused by fast motion.
In fact, we leverage either uncertainty metric as a safety check by only initiating a reinitialization when both indicate deterioration in tracking, ultimately making our joint uncertainty metric less vulnerable to the weaknesses of either metric in isolation. We consider a failure of both uncertainty metrics as an inevitable reinitialization.

CHAPTER 5

Experiments

In this chapter, we reiterate the objectives of this thesis and provide an overview of our experimental setup. Our method is evaluated using the BOP Toolkit [HMB+18] and compared against other dedicated RGB-D object tracking methods, depth-only trackers, and traditional registration methods. An extensive ablation study identifies the impact of our proposed components. Moreover, we discuss our autonomous reinitialization heuristic and compare it against reinitializations after fixed time intervals. Furthermore, we provide a runtime analysis for the tight time constraints of the tracking task. At the end of this chapter, qualitative tracking sequences are shown and failure cases are discussed.

5.1 Objectives

Within this thesis, our objectives for the evaluation are set by our proposed research questions. Most importantly, our experiments provide insights on how our RL-based, depth-only object tracking approach performs in comparison to other SotA methods.
Moreover, an ablation study identifies the most suitable configuration of our proposed network architectures from Sec. 4.5, to eventually answer how to best combine *frame-to-frame registration* and *frame-to-model refinement* in a single neural network. In addition, we investigate the achieved accuracy of our fusion variants compared to carrying out each subtask independently.
To address the presence of outliers, we evaluate our method both with and without our segmentation techniques presented in Sec. 4.7 to demonstrate how our outlier-rejection affects tracking.
With special attention to real-world applications, we showcase the efficacy of our uncertainty metrics for the tracking task. These metrics enable our method to autonomously detect deterioration in tracking and consequently balance the number of reinitializations with the achieved tracking accuracy.

To summarize, the evaluation of our method answers the following Research Questions (RQ).

- **RQ1**: How to approach object tracking as a reinforcement-learning problem? And how does this compare to SotA-approaches?
- **RQ2**: How to combine *frame-to-frame registration* with (*frame-to-model refinement*)? What impact does this combined approach have on the tracking accuracy over multiple frames as compared to either approach alone?
- **RQ3**: How robust is the object-tracker to observational noise, e.g. occlusion by other objects or hands and imperfect segmentation?
- **RQ4**: How to exploit the agent's behavior to balance the number of reinitializations with the tracking accuracy, when no annotated data is available?

5.2 Experimental Setup

This section serves as an insight into our experimental setup. We explain the used datasets and metrics, along with implementation details for our object tracking method. All of our results are conducted using the BOP Toolkit [HMB+18], as it is a standardized tool for the evaluation of pose estimation methods. For our experiments, we utilize a machine with an Intel Core i7-8700K @ 3.70GHz processor and an NVIDIA GeForce RTX 2080 Ti graphics card.

5.2.1 Datasets

YCB-V:

Introduced in 2018, the YCB-V dataset [XSNF18] has emerged as a widely used dataset to benchmark object tracking methods. In total, it contains 92 video sequences with 21 distinct objects. Each of these sequences captures up to six static objects on a table, with a camera circulating around the scene. The official data split holds out 12 scenes for testing. As depth-based features can be learned efficiently from limited data, we subsample the real training split to contain only every 7^{th} frame, contributing to reduced training time. All of our quantitative experiments are conducted on the keyframes of the evaluation split. We do not use synthetic training images.

HO-3D-r:

HO-3D-r [PPL+21] is a refactored version of the HO-3D [HROL20] dataset for object and hand pose estimation provided in the BOP format. The refactored version provides 10 scenes for training and 44 for testing, each manipulating a single object. A total of 10 objects has been selected from the YCB-V dataset [XSNF18].
In the context of 6D object tracking, we consider the data split to be unsuitable for

robustly learning object trajectories due to the limited training data. Solely for the purpose of this thesis, we define a custom data split by withholding 1-2 scenes per object category, while the remaining scenes are utilized for training. We provide qualitative results of our object tracking method on this dataset.

DexYCB:

The DexYCB [CYX+21] dataset is a large-scale dataset focusing on hand manipulation and utilizing a subset of objects from the YCB-V dataset [XSNF18]. For this dataset, 1000 RGB-D video scenes each manipulating a single object have been captured from eight different, but static viewpoints. The average video length is around 70 frames leading to a total of 582K RGB-D frames. We provide qualitative results on this dataset, however, our method did not achieve satisfactory results. We explain this by the large fraction of each scene, where the object is not manipulated. Consequently, our agent is unable to generalize well to diverse object motion.

5.2.2 Metrics

In the field of 6D pose estimation and 6D object tracking, related work utilizes the Average Distance to Model Points (ADD) and ADD with indistinguishable views (ADI) [HLI+12] to evaluate the accuracy of the estimated poses. The ADD is given by measuring the average distance between corresponding model points $m \in M$ under an estimated pose $\hat{T}$ and the ground-truth pose T.
The ADI is especially important for objects with symmetry axis, leading to multiple, indistinguishable views. For example, a bowl can be rotated around its center axis and remains aligned but the ADD score is changing due to the rotational offset. For that particular reason, the ADI metric takes the geometrical ambiguities into account by calculating the mean distance to the nearest neighbor.
Mathematically speaking, the metrics are formulated as follows:

$$\text{ADD} = \frac{1}{|M|} * \sum_{m \in M} ||\hat{T}m - Tm|| \tag{5.1}$$

$$\text{ADI} = \frac{1}{|M|} * \sum_{m_1 \in M} \min_{m_2 \in M} ||\hat{T}m_1 - Tm_2|| \tag{5.2}$$

For all of our experiments, we report the *Area under the Precision-Recall Curve* (AUC) score. A recall score is defined as the percentage of object poses within a certain error-threshold. The AUC score is defined as the average recall across a range of uniformly spaced error-thresholds.
Noteworthy, we consider the ADI AUC score as the more meaningful metric for our experiments, as our depth-only approach is unable to differentiate indistinguishable views by explicitly not considering textural information.

5.2.3 Implementation Details

Agent:

Following the definition in [BPV22], our agent aligns point clouds in the normalized model space such that the distance to the farthest point is at most 1. This constrains the potential inputs and thereby making our approach less sensitive to varying object scales. Furthermore, points outside the unit sphere formed by the normalized model space are implicitly labeled as either misaligned or outliers. A strong geometrical cue in addition to the raw point coordinates are surface normals, thereby we adapt prior work [BPV22] and extend our inputs to also incorporate surface normals.
For each of the axes, our agent manipulates point clouds by predicting an action-distribution, formed by an action-step vector [0.00066, 0.002, 0.006, 0.018, 0.054] in positive and negative direction along with a zero-action, when alignment is reached. Using an exponential series enables a wide range of action-steps, allowing to refine large offsets as well as fine-grained adjustments. For translation, the action-steps refer to units in the normalized model space, where the longest distance equals one. On the rotation axis, we interpret the action-steps as radians.
Per frame, TrackAgent predicts action-steps over ten refinement iterations to estimate the relative transformation necessary for realigning the pose. For the sake of comparison, we always initialize the trajectories with the ground-truth pose during training and evaluation.

Hyperparameters:

For blending IL and RL, prior work [BPV22] scales the RL-loss term by the factor $\alpha = 0.2$, identified to be the best configuration for YCB-V. We define our reward according to the notion in Eq. 4.1, denoted as $r = (0.5^{+}, -0.1^{0}, -0.6^{-})$. This definition is used for concurrently optimizing the policy for single-frame refinement steps *as well as* multi-frame object trajectories.
Parameters for the segmentation branch are adapted from prior work [BPV22] that scales the loss term by $\beta = 7$.

Training:

Our agent learns object trajectories from scratch by estimating the relative transformation between frames across a sequence of ten frames. During training, two replay buffers of size 128 collect samples of 1) single-frame refinement trajectories and 2) multi-frame object trajectories to eventually be utilized by the policy update.
We artificially generate error-afflicted point clouds with a 20% outlier ratio by adapting the augmentation strategy from [BPV22]. We increase the variety during training by randomly sampling trajectories from our already subsampled dataset such that the overall number of frames per epoch only accounts for $1/5^{th}$ of the available training frames. While this significantly reduces the training time, random sampling of the trajectories also limits overfitting and enables our agent to better generalize. We train our agent on frame-

AUC [%] ↑	se(3)‡ [WMRB20]		ICG‡ [SST22]		POT† [WPK+13]		RGF† [IWC+16]		ICP§ [BM92, ZPK18]		Reg§ [BPV21]		Ref§ [BPV22]		**ours**§		**ours**	
Reinitialization every n^{th} frame	None		None		None		None		30		30		30		30		Autonomous	
# Reinitializations	0		0		0		0		137		137		137		137		118	
Metric	ADD	ADI	ADD	ADI	ADD	ADI	ADD	ADI	ADD	ADI	ADD	ADI	ADD	ADI	ADD	ADI	ADD	ADI
master chef can	**93.9**	**96.3**	66.4	89.7	55.6	90.7	46.2	90.2	78.7	*94.1*	*84.2*	92.2	80.2	91.6	76.8	92.0	75.0	91.1
cracker box	**96.5**	**97.2**	82.4	92.1	*96.4*	***97.2***	57.0	72.3	78.3	88.4	84.9	91.9	88.6	92.8	93.3	95.7	92.3	95.1
sugar box	**97.6**	98.1	96.1	**98.4**	*97.1*	*97.9*	50.4	72.7	81.0	92.2	89.8	94.9	95.7	97.4	95.6	97.0	95.6	97.0
tomato soup can	**95.0**	97.2	73.2	**97.3**	64.7	89.5	72.4	91.6	75.3	91.4	*92.8*	*95.8*	90.8	94.4	91.0	95.7	87.8	92.6
mustard bottle	95.8	97.4	96.2	**98.4**	***97.1***	98.0	87.7	*98.2*	88.0	94.4	94.3	96.4	96.6	97.7	96.0	97.5	96.1	97.6
tuna fish can	86.5	91.1	73.2	**95.8**	69.1	*93.3*	28.7	52.9	76.4	88.1	***90.4***	*93.3*	60.0	81.9	64.5	82.6	73.6	91.7
pudding box	**97.9**	**98.4**	73.8	88.9	*96.8*	*97.9*	12.7	18.0	81.5	91.4	93.6	96.1	89.2	94.0	94.4	96.5	94.0	96.4
gelatin box	**97.8**	98.4	97.2	**98.8**	*97.5*	*98.4*	49.1	70.7	81.3	92.0	95.7	97.1	87.6	92.9	96.9	97.7	96.9	97.7
potted meat can	77.8	84.2	**93.3**	**97.3**	*83.7*	86.7	44.1	45.6	81.6	*89.9*	75.5	80.7	77.5	80.1	78.9	82.5	83.4	88.8
banana	94.9	97.2	**95.6**	**98.4**	86.3	96.1	93.3	*97.7*	71.9	87.2	88.8	92.8	*94.8*	97.0	93.6	96.6	93.8	96.7
pitcher base	96.8	97.5	97.0	**98.8**	97.3	97.7	***97.9***	*98.2*	90.3	96.1	92.9	95.9	94.5	96.9	95.3	97.1	95.4	97.2
bleach cleanser	**95.9**	97.2	92.6	**97.5**	95.2	97.2	***95.9***	*97.3*	71.7	89.2	82.0	91.9	93.2	96.4	91.4	95.5	91.3	95.5
bowl	**80.9**	94.5	74.4	**98.4**	30.4	*97.2*	24.2	82.4	*78.1*	93.1	72.9	85.7	64.1	87.3	48.7	78.6	63.1	88.1
mug	91.5	96.9	**95.6**	**98.5**	83.2	93.3	60.0	71.2	81.8	93.3	84.0	89.6	93.9	96.6	*94.4*	*96.9*	94.3	*96.9*
power drill	96.4	97.4	96.7	**98.5**	97.1	97.8	***97.9***	*98.3*	78.1	90.3	90.6	94.1	94.1	96.2	92.4	95.0	94.3	96.5
wood block	95.2	96.7	93.5	**97.2**	***95.5***	*96.9*	45.7	62.5	71.9	90.5	83.6	91.6	89.4	95.6	91.3	95.6	91.4	95.6
scissors	**95.7**	**97.5**	93.5	97.3	35.6	4.2	16.2	20.9	38.6	39.9	73.5	57.3	69.1	70.2	79.5	72.3	*80.1*	*90.0*
large marker	**92.2**	96.0	88.5	**97.8**	35.6	53.0	12.2	18.9	51.1	70.5	50.5	66.4	45.6	64.6	54.9	68.4	*67.0*	*94.2*
large clamp	**94.7**	**96.9**	91.8	**96.9**	61.2	72.3	62.8	80.1	63.6	77.6	81.0	87.5	*82.1*	88.6	*82.1*	89.0	81.9	*89.7*
extra large clamp	91.7	95.8	85.9	94.3	***93.7***	***96.6***	67.5	69.7	60.5	80.2	80.7	88.5	78.7	88.2	89.1	94.9	90.8	95.7
foam brick	93.7	96.7	96.2	**98.5**	***96.8***	*98.1*	70.0	86.5	69.6	83.8	94.7	96.4	85.2	90.9	95.3	96.9	95.1	96.9
All Frames	**93.0**	95.7	86.4	**96.5**	78.0	90.2	59.2	74.3	74.9	88.6	84.6	90.6	83.6	90.7	84.9	91.6	*86.8*	*93.6*

Table 5.1: **Tracking Accuracy on YCB-V [XSNF18]**: We report the average recall for 50 uniformly distributed thresholds in the range $[0, 10cm]$, i.e. the AUC score. The results are taken from [SST22] for the RGB-D (‡) and two of the depth-only methods (†). Each method (re-)initializes tracking with the ground-truth pose. Best overall AUC per metric is highlighted in **bold**, best depth-based AUC per metric in *italics*.

batches of size 8, leading to a total of approximately 32 objects to track simultaneously, given an average of four objects per frame on the YCB-V dataset. Our agent is trained for 60 epochs using Adam [KB14] as an optimizer, starting with a learning rate of 0.003 and halving it every 20 epochs. On the NVIDIA GeForce RTX 2080 Ti graphics card, the training is finished after 24 hours.

5.3 Tracking Accuracy on YCB-V

We list the achieved tracking accuracy of our hybrid approach, along with other model-based 6D object tracking baselines in Tab. 5.1. We compare our method against SotA RGB-D object tracking methods [WMRB20, SST22], three depth-based approaches [WPK+13, IWC+16, BM92, ZPK18] as well as *frame-to-frame registration* [BPV21] and *frame-to-model refinement* [BPV22] both as an individual task.

Our experiments emphasize the benefit of jointly solving *frame-to-frame registration* and *frame-to-model refinement*, contributing to an increased tracking ADD/ADI AUC score as compared to each task in isolation. Interestingly, *frame-to-frame registration* [BPV21] and *frame-to-model refinement* [BPV22] both achieve a comparable overall accuracy during tracking. However, the observed differences in accuracy at the object-category level of the individual subtasks further motivate our hybrid approach to merge the

strengths of *frame-to-frame registration* and *frame-to-model refinement*. By leveraging our uncertainty metrics, we reduced the number of reinitializations and simultaneously improved the tracking accuracy, as compared to fixed reinitializations after a certain interval.
All of the RL methods (*frame-to-model refinement* [BPV22], *frame-to-frame registration* [BPV21], ours) outperform ICP and the other depth-based baselines [WPK+13, IWC+16]. Noteworthy, reinitializations are disabled for POT [WPK+13] and RGF [IWC+16]. Our depth-only TrackAgent clearly closes the gap towards dedicated RGB-D methods [WMRB20, SST22]. For some objects, our method demonstrates comparable results to se(3)-TrackNet [WMRB20] and we outperform ICG [SST22] across multiple object categories. We want to emphasize, that se(3)-TrackNet leverages large amounts of synthetic training data, while we are only using a subsampled version of the training data, and ICG, as a handcrafted method, entirely avoids training.

5.4 Ablation Study

For our ablation study, we compare our proposed architectures that fuse *frame-to-frame registration* and *frame-to-model refinement* 1) at the input side (*early-fusion*), 2) after the encoding stage (*mid-fusion*), and 3) after the policy embeddings (*late-fusion*). We evaluate our methods with fixed reinitializations across varying time horizons.
Moreover, we investigate the sensitivity of our approach to outliers by letting the baselines compete against variants incorporating 1) our handcrafted *geometry-guided mask filtering*, 2) the segmentation branch for learning-based outlier-rejection, or 3) both in combination. The results of all of our variants are listed in Tab. 5.2.
Dedicated to real-world applications, we employ our reinitialization heuristic to demonstrate the efficacy of our uncertainty metrics for balancing the computational cost of carrying out reinitializations with the desired tracking accuracy.
Addressing the time restrictions of the tracking objective, we showcase and compare the runtimes of our proposed variants to conclude our ablation study.

5.4.1 Fusion-Variants vs. Independent Subtasks

Following the results in Tab. 5.2, we conclude that the early-fusion variant as our simplest approach fails to leverage the benefits of our joint consideration. While being outperformed by the more complex fusion variants, the early-fusion variant is also surpassed by the independent subtasks, labeling the joint encoding of this variant as unsuitable for fusing *frame-to-frame registration* and *frame-to-model refinement* despite it's simplicity.
The late-fusion variant as our most complex variant emerges as a more accurate choice by achieving superior accuracy compared to the standalone subtasks. However, when compared with the mid-fusion variant, no advantage of the duplicated policy networks in the late-fusion variant is observed.
Consequently, our results demonstrate the mid-fusion to be the best tradeoff between network complexity and its achieved accuracy. The mid-fusion variant outperforms

Reinitialize every nth frame	30		90		120	
AUC [%] ↑	ADD	ADI	ADD	ADI	ADD	ADI
Registration	*84.6*	*90.6*	***80.5***	*86.6*	*78.7*	*84.8*
+ mask filtering	80.1	88.0	71.0	80.0	68.5	77.7
+ segmentation branch	81.3	88.3	74.3	81.9	73.5	80.3
+ both	77.2	86.6	70.2	81.0	67.9	78.0
Refinement	81.9	90.2	72.3	81.9	72.8	80.5
+ mask filtering	77.0	87.4	68.2	78.4	66.0	76.3
+ segmentation branch	*83.6*	*90.7*	*78.0*	*85.0*	*77.3*	*85.1*
+ both	82.6	90.2	73.3	82.6	71.2	80.0
Early-Fusion	*82.2*	*89.8*	*78.1*	*85.7*	*75.8*	*84.2*
+ mask filtering	78.3	87.2	71.1	79.9	68.6	76.4
+ segmentation branch	74.0	86.0	66.5	77.1	63.0	73.4
+ both	71.1	83.7	60.1	69.7	55.7	66.0
Mid-Fusion	82.1	90.0	77.3	85.5	74.0	82.3
+ mask filtering	79.2	88.4	72.0	80.7	69.2	78.4
+ segmentation branch	*84.9*	***91.6***	*80.0*	***87.3***	*79.3*	***86.6***
+ both	79.9	88.3	72.0	86.7	69.1	77.8
Late-Fusion	***85.1***	*91.3*	*80.3*	*86.5*	***79.5***	*85.4*
+ mask filtering	81.3	89.4	72.0	80.6	71.8	79.9
+ segmentation branch	83.7	90.2	77.2	84.3	76.4	83.3
+ both	78.1	86.8	70.9	79.5	68.0	78.2

Table 5.2: **Ablation Study on YCB-V [XSNF18]:** We present results for our fusion-variants along with the independent tasks. Moreover, we incorporate different segmentation strategies to identify and reject outliers. For our ablation study, different time intervals have been used to reinitialize tracking. Best overall AUC per metric is highlighted in **bold**, best method-specific AUC per metric in *italics*.

not only the independent subtasks but especially all other fusion variants across all reinitialization intervals. We conclude the distinct encoder branches to be essential for our joint consideration, enabling the finegrained optimization to either subtask. Processing the combined state vector by one unified policy network is shown to be the optimal fusion-variant for leveraging the strengths of either subtask from the common state vector.

5.4.2 Outlier-Rejection

To our surprise, none of our methods improved the achieved AUC score by incorporating our handcrafted *geometry-guided mask filtering*. This issue may be explained by the disparity of filtered point clouds during inference and unfiltered point clouds during training, leading to confusion for the agent. Moreover, solely computing the depth differences might prune masks too aggressively under error-afflicted poses, resulting in an insufficient number of data points and consequently further contributing to the deterioration in tracking.
The impact of utilizing the segmentation branch to learn the masks varies across our

architectures. For the *frame-to-model refinement* subtask, the network's ability to classify and reject outliers significantly increases accuracy, whereas *frame-to-frame registration* performs best by processing raw, unfiltered point clouds. We expect such a behavior as defined by either subtask's nature. Aligning a noisy source to a perfect object model may be disrupted by the presence of outliers. Vice versa, point clouds from consecutive observations exhibit similar imperfections and outliers. In the best case, these shared defects serve as an additional cue for the *frame-to-frame registration* objective to achieve alignment.
In our joint task, neither the early- nor late-fusion variant took advantage from learning-based outlier-rejection. Only our mid-fusion variant leveraged the efficacy of the segmentation branch, contributing to the best achieved ADI AUC scores. Especially with less frequent reinitializations, incorporating the segmentation branch boosts the achieved recall score by up to 4.3% for the mid-fusion variant.
To summarize, we aimed to enable our methods to detect and eventually reject outliers to reduce their impact. The efficacy of our employed outlier-rejection strategies varies across our proposed architectures. We observe that *frame-to-model refinement* as a standalone method is the most susceptible to the influence of outliers, with the segmentation branch having the most significant impact on this variant's recall score. Conversely, our fusion-variants are in general more robust to outliers due to integrating the *frame-to-frame registration* objective. For example, the late-fusion variant achieves slightly lower scores without any filtering of the point clouds as compared to the mid-fusion variant incorporating the segmentation branch. We explain this observation with a certain sensitivity of the mid-fusion variant towards inaccurate segmentation, however, this sensitivity is mitigated by the learned outlier-rejection, contributing to the best recall score of our fusion variants.

5.4.3 Reinitializations

When object tracking methods are deployed in a real-world scenario, accurately identifying deterioration in tracking becomes a more challenging problem without ground-truth data. For our previously listed results, we employed a straightforward strategy of reinitializing tracking after fixed time intervals. However, such a procedure may introduce additional computational effort by reinitializing accurately tracked object instances, resulting in a reduced frame processing rate.
Conversely, our uncertainty metrics introduced in Sec. 4.8 exploit information already present in our pipeline as an autonomous reinitialization trigger. Visualized in Fig. 5.1, an extensive study demonstrates computing the object visibility in interaction combined with an action-based uncertainty as an effective reinitialization trigger when tracking is unable to recover.
In contrast to fixed reinitializations, our autonomous reinitialization heuristic reduces the overall quantity and consequently the computational effort, while simultaneously improving the tracking accuracy. On the one hand, our experiments indicate a higher visibility threshold to correspond to high recall values, consequently with frequent reinitializations. On the other hand, setting higher stepsize thresholds actively suppresses

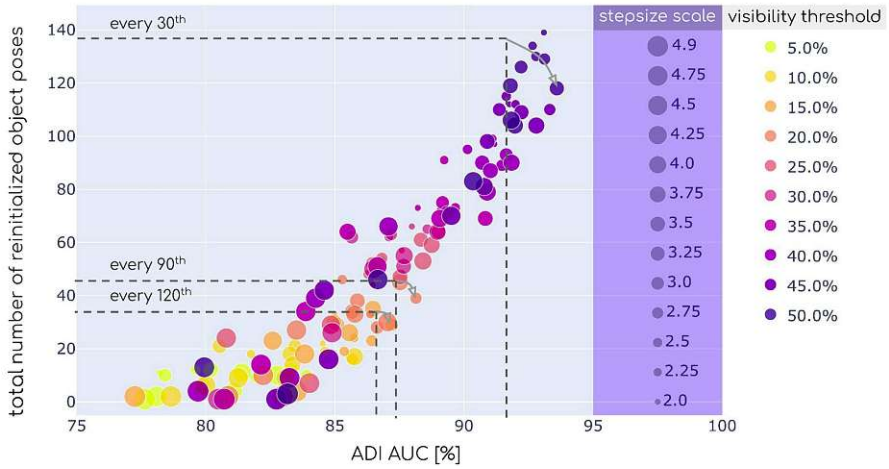

Figure 5.1: **Employing our Uncertainty-Metrics as Reinitialization Heuristic:** By adjusting the thresholds of our reinitialization heuristic, we control the interplay of the executed reinitializations with the achieved recall score. We observe higher visibility thresholds to result in frequent reinitializations. Conversely, larger stepsizes actively minimize reinitializations, albeit at the cost of a limited recall score. For our experiments, we count reinitializations per object instance. The results of fixed reinitializations are shown in dashed lines, along with arrows pointing to the recall gain achieved by our autonomous triggering.

reinitializations up to a bare minimum. However, this is paid for by a reduced tracking accuracy.
To conclude, finetuning the thresholds of our uncertainty metrics allows to control the balance between the carried out reinitializations and the resulting tracking accuracy. This enables users to execute our TrackAgent in an optimal, application-dependent configuration, e.g. to prioritize high accuracy or minimize costly reinitializations.

5.4.4 Runtime

We finish our ablation study by investigating and comparing the runtimes of our methods, providing insights into the temporal constraints of our tracking approach.
Fig. 5.2 visualizes how the runtime of our TrackAgent pipeline scales with the amount of tracked objects. For this experiment, we utilized our best performing model, namely the mid-fusion variant with incorporated segmentation branch. When tracking a single object, we observe computing the refinement transformation, i.e. tracking itself, to be the most time-intensive step. Nevertheless, with increasing object count, the runtime of the sequential preprocessing and rendering steps grows proportionally. In contrast, the runtime of tracking itself scales more efficiently by harnessing the GPU for parallel processing. When tracking at least four objects, preprocessing emerges as the slowest component of our pipeline. Offering the most potential for improvement, this step may

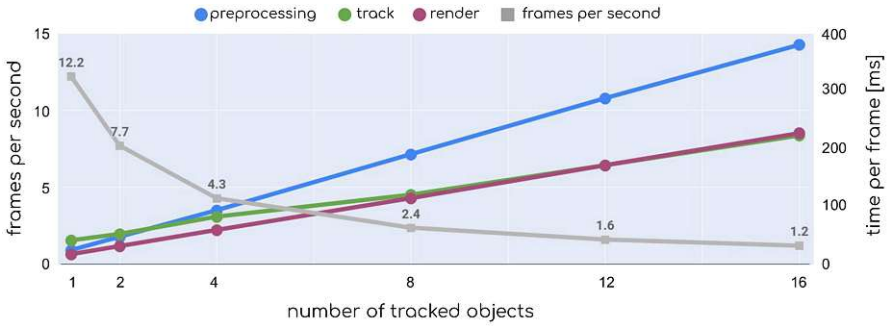

Figure 5.2: **Runtime Analysis for Varying Object Counts:** For tracking a single object, the inference time of the model is the most time-consuming step. However, the time spent for the sequential preprocessing step scales linearly with the object count whereas tracking itself leverages parallel processing for a more efficient scaling of the runtime. Consequently, we see the most potential for improvement in the preprocessing step, e.g. by preparing the point clouds in parallel.

be optimized by extracting the point clouds in parallel.
Moreover, illustrating the frame processing rate in relation to the achieved tracking accuracy in Fig. 5.3 serves as an insight to the influence of utilizing more complex network architectures. The achieved frame rates of our baselines, i.e. without any of our proposed outlier-rejection techniques, are all within 13-15 fps. Integrating the segmentation branch scatters the resulting runtimes further apart, reducing the throughput by at least 0.5 fps. Nevertheless, the runtimes match our expectations, where increased network complexity contributes to higher latency of inference.
However, this thesis specifically focuses on how to fuse the strengths of *frame-to-frame*

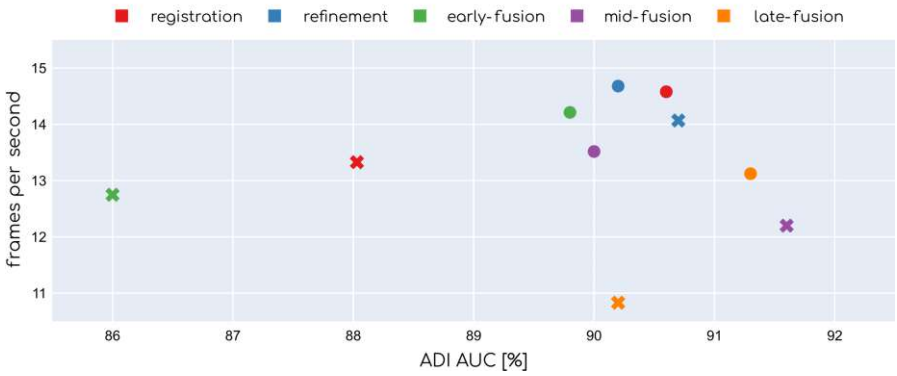

Figure 5.3: **Accuracy and Runtime Comparison:** We illustrate the runtimes of our proposed fusion variants and the isolated subtasks along with the achieved accuracy. For each method, we list results of the baseline (●) along with additionally incorporating the segmentation branch (✖). All runtimes have been collected using fixed reinitializations after 30 frames.

registration and *frame-to-model refinement* in a novel network architecture. Besides that, we are considering the resulting processing rates as sufficient for a real-life application. Consequently, we are prioritizing gains in accuracy over reduced runtime and therefore consider the mid-fusion variant with an integrated segmentation branch as our most efficient fusion-variant.

5.5 Qualitative Results

In addition to the quantitative analysis, this section provides qualitative results of our tracked sequences for the datasets YCB-V in Fig. 5.4, HO-3D-r in Fig. 5.5, and DexYCB in Fig. 5.6. Noteworthy, we visualize all of our predicted poses in the RGB frame for better visibility. Our visualization illustrates the ground-truth pose (blue), the estimated pose (red), and the overlapping fractions (green) along with the outlines (pink) of our visibility mask.

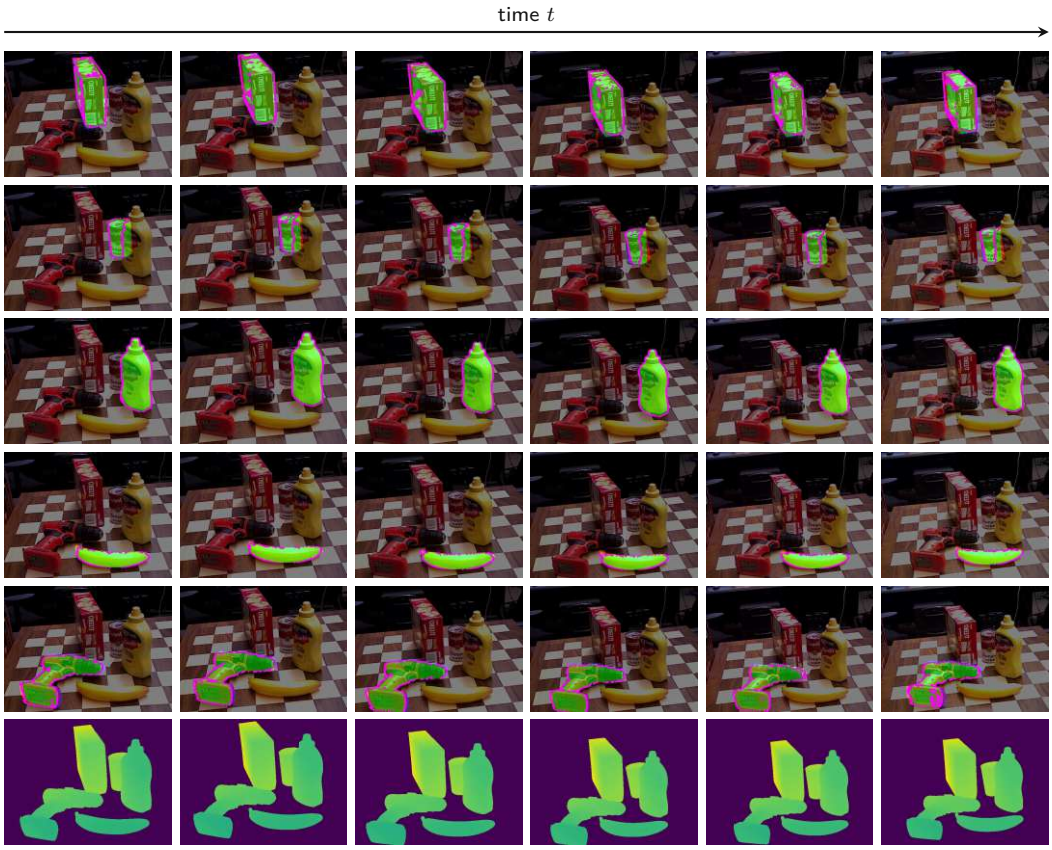

Figure 5.4: **Qualitative Results on YCB-V:** In addition to the visualization of our predictions, the bottom row provides a virtual scene representation by utilizing the rendered depth per object.

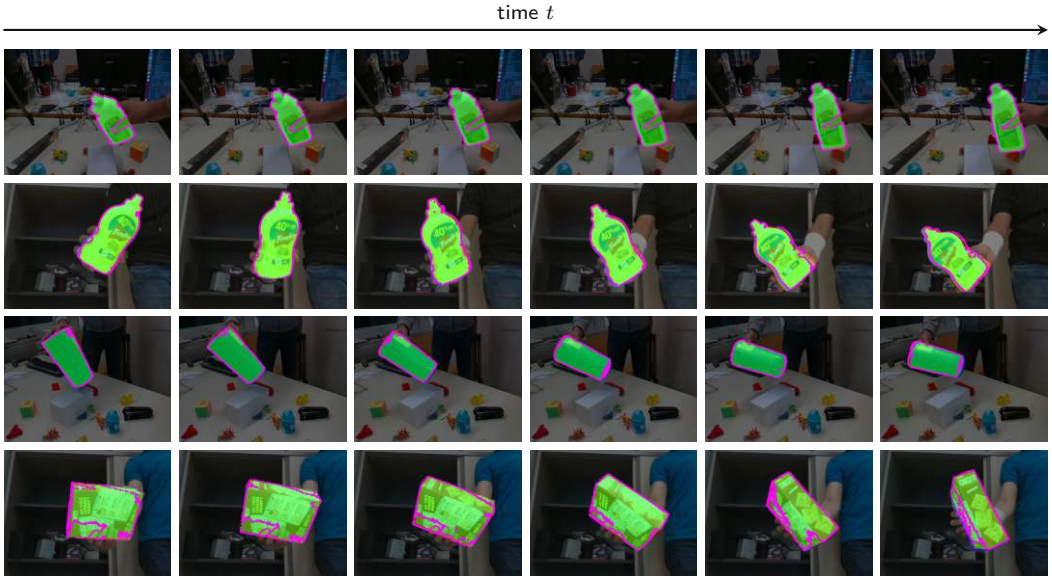

Figure 5.5: **Qualitative Results on HO-3D-r:** Tracking sequences evaluated on our custom data split.

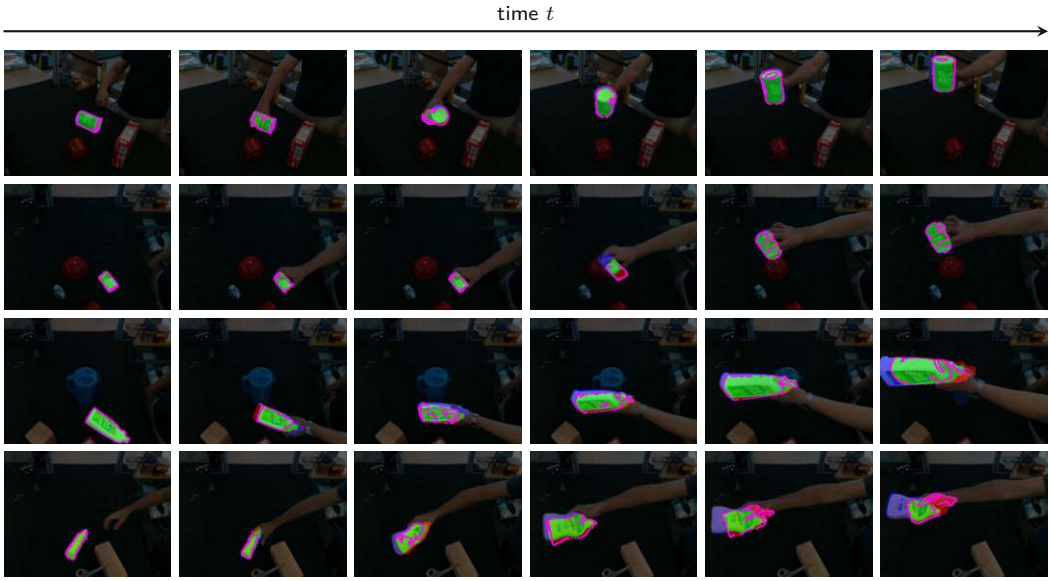

Figure 5.6: **Qualitative Results on DexYCB:** Our method was unable to generalize well to the tracking objective when utilizing this dataset. We explain this by the large fraction of frames with little or no object motion, making the zero-action very likely and therefore reducing the efficacy in tracking.

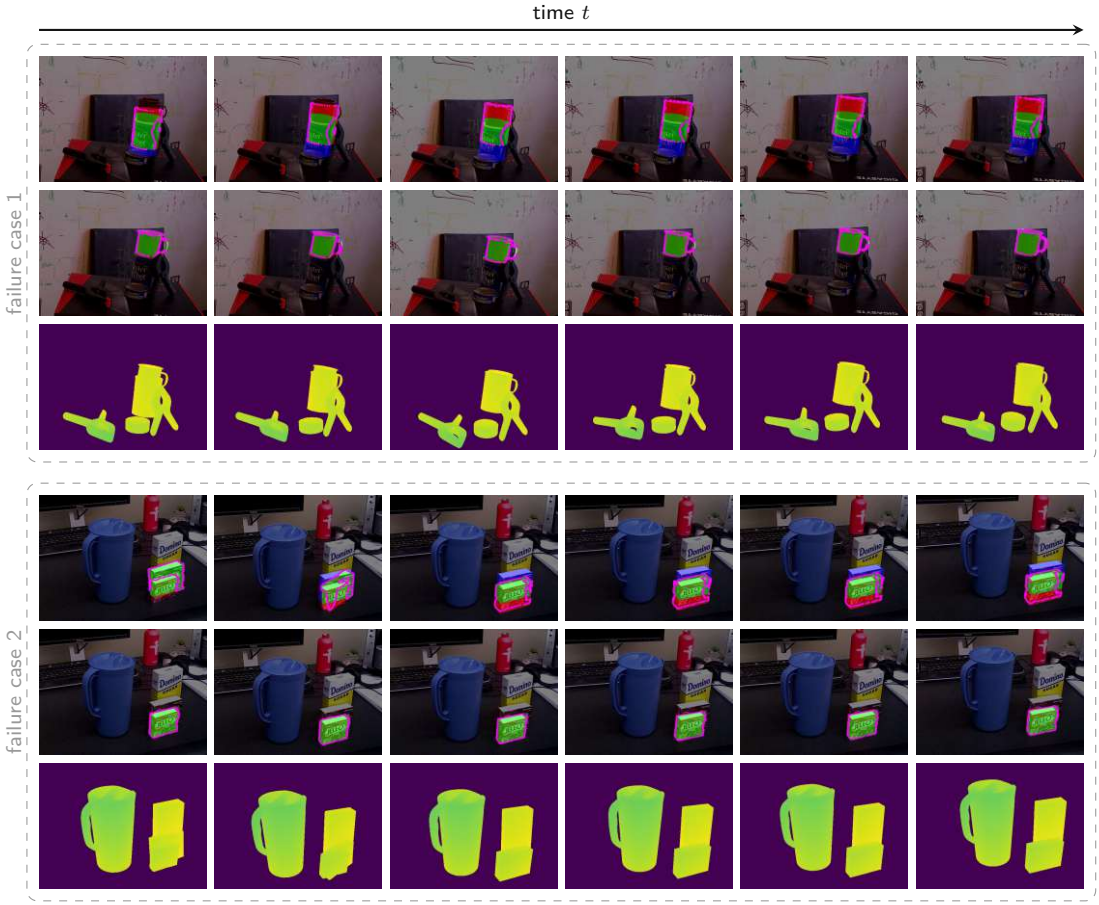

Figure 5.7: **Failure Cases on YCB-V:** We observe a likelihood of our agent mismatching similar-shaped objects. The top failure case features the *master chief can* being matched onto the *mug*. In the bottom failure case, our agent is unable to distinguish the two *jello* objects and, even though correctly initialized, fits both poses onto the object in front.

5.6 Analysis of Failure Cases

During our experiments, we observed our agent to be sensitive to similarly shaped objects, resulting in mismatches between distinct object categories. For example, our agent repeatedly matches the *master-chef-can* object onto the *mug* object, highlighting the mug handle as an insufficient geometrical cue. Similarly, our agent is unable to distinguish the two cubic *jello* objects, even though the object poses are initialized correctly. These failure cases may be effectively addressed by integrating physical constraints, particularly by enforcing non-intersecting objects, as proposed in [BPV22].
Regarding the DexYCB dataset [CYX+21], our method did not generalize well. We explain this by the data organization, where the object remains static over a significant

fraction of frames per scene. Consequently, the zero-action becomes very likely as compared to other action-steps, therefore hindering the learning process.
As a general remark for our framework, we exploit the previous pose to render a mask for the current frame. This postulates limited motion in between frames, otherwise, the accuracy of our mask decreases and consequently, the efficacy of our method is challenged.

CHAPTER 6

TrackAgent in a Dynamic Handover Scenario

In this chapter, we demonstrate the practical ability of our TrackAgent in a dynamic handover scenario, showcasing how our depth-based tracking method may be embedded in robotic applications using *pose-based visual servoing* (PBVS). Mounting a camera on the robot's end effector and tracking the object's pose by utilizing the observed depth frame allows the robot to adjust its position accordingly, keeping the object centralized in the image frame even when manipulated. Moreover, we simulate a handover scenario during our demonstration, where the robot approaches the object to potentially execute a grasp. However, as grasping is a complex topic going beyond the scope of this thesis, our focus is solely on showcasing the feasibility of such applications.

6.1 Demonstration Setup

By employing our TrackAgent in a real-world scenario, we are aiming to highlight the relevance of object tracking in dynamic environments and especially robotic manipulation tasks. Simultaneously, we want to showcase the practicability of our approach, in contrast to the evaluation on datasets, in a real-world setting.
PBVS controls a robot's pose by utilizing visual feedback from an RGB-D camera, enabling to adjust the robot's movements to specific observations in the environment. By mounting an RGB-D camera on the robot's end effector, we leverage our TrackAgent to enable object tracking from the robot perspective, allowing to control the robot's pose relative to the object's pose using PBVS [CH06]. In our real-world experiment, we utilize a KUKA LBR iiwa 14 R820 robot with seven degrees of freedom, specifically suitable for human-robot interactions. We structure our demonstration by defining two phases:

- During the *tracking phase*, as illustrated in Fig. 6.1, the robot keeps the object centralized in the camera view with a distance towards the object of 50cm, while a human dexterously manipulates the object's pose. In addition, the object's orientation in the image frame controls the robot's orientation to maintain the initial object orientation from the camera's perspective.

- An *approaching phase* serves as a demonstration for simulating a potential grasp in our dynamic handover scenario, as visualized in Fig. 6.2. The object is approached by the robot, maintaining close distance towards the object for a specific time interval to theoretically execute a grasp until the robot returns to the tracking phase. Noteworthy, the object is still tracked during the approaching phase, however, significant manipulations during this phase may cause a failure in a real grasping scenario.

6.2 Real-World Results

We present the results of our dynamic handover scenario from four different views. Most importantly, we provide the color- and depth-frame from the robot's camera view, along with the object's outlines under the estimated pose visualized in magenta. For a better understanding of the robot's movements, we additionally provide a front- and side-view, where the latter is especially important for demonstrating the approaching phase for the simulated handover. Sequences of our TrackAgent in our dynamic handover scenario are shown in Fig. 6.1 and Fig. 6.2.
During our experiments, we observed that TrackAgent consistently achieves accurate pose estimates, therefore effectively guiding the robot's movements for successful object following. We want to emphasize the usage of a model trained on YCB-V [XSNF18], where only the camera views are manipulated while objects remain static. In contrast, our demonstration involves simultaneous manipulation of both the camera view and the object, showcasing strong generalization of our method.

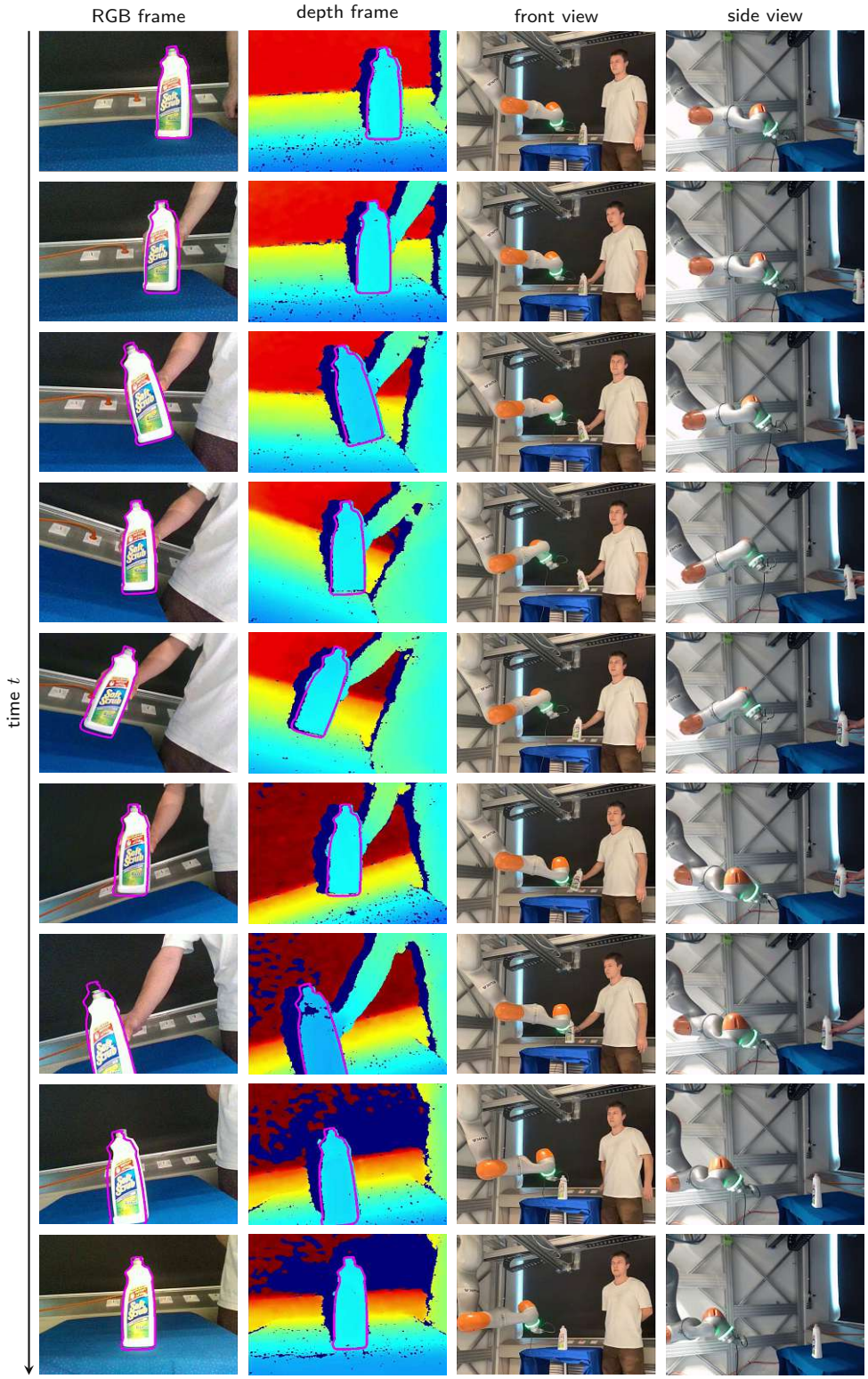

Figure 6.1: **TrackAgent in the Wild during the Tracking Phase:** PBVS enables the robot to adjust its position and orientation relative to the object, ensuring the object remains centralized in the image frame. Moreover, the robot rotates accordingly with the object to preserve the initial orientation of the object from the camera view. Best viewed digitally.

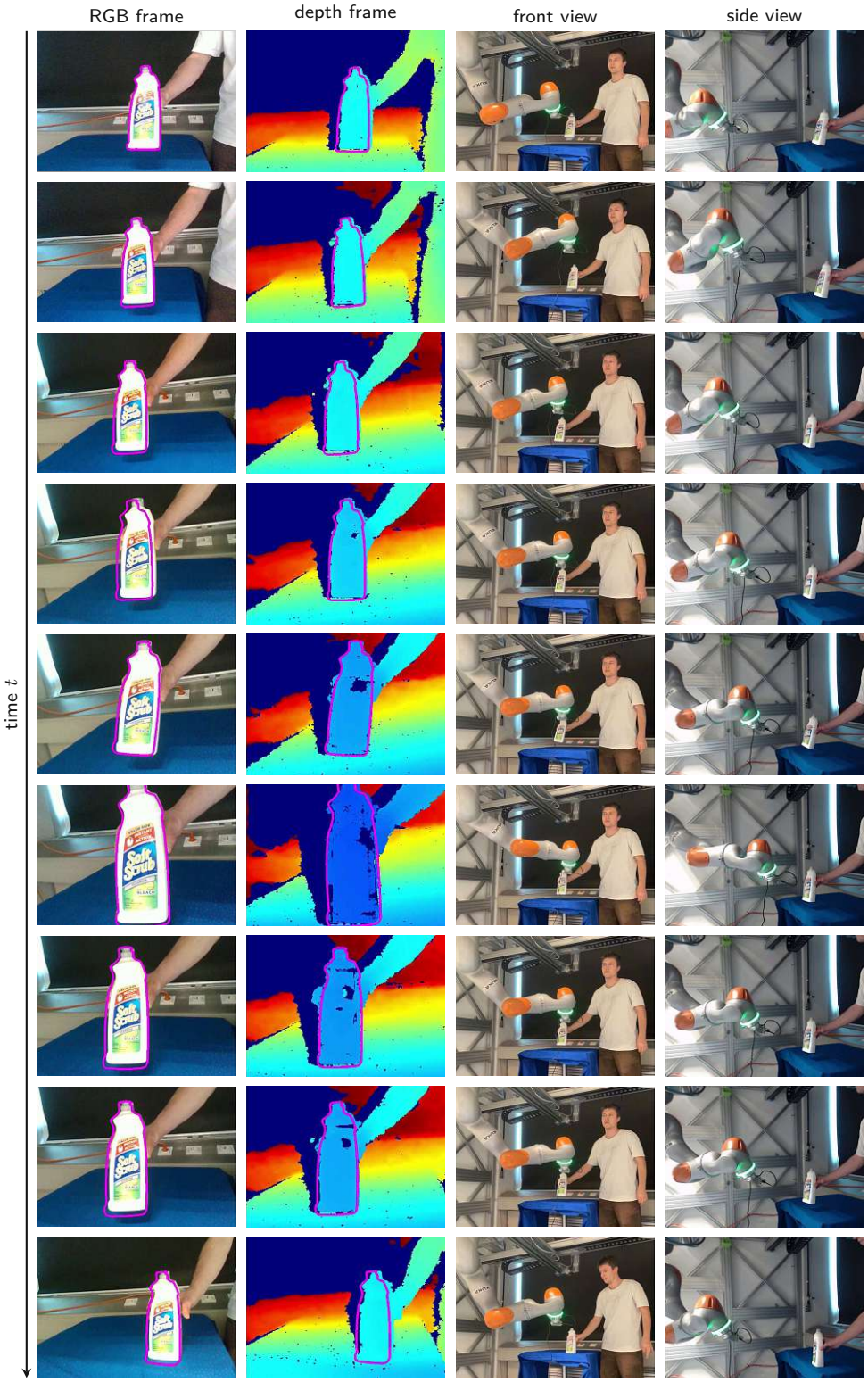

Figure 6.2: **TrackAgent in the Wild during the Grasping Phase:** We simulate the handover of the object by approaching the object with the robot's end effector. The robot remains close to the object's proximity to enable a time window for potentially executing a grasp. The grasping phase is concluded by returning to our predefined safety distance and continuing in the tracking phase. Best viewed digitally.

CHAPTER 7

Conclusion and Future Work

Within this work, we have introduced TrackAgent, an RL approach stemming from a concurrent solving of *frame-to-frame registration* and *frame-to-model refinement* to track 6D object poses as a depth-based alignment problem. Building upon previous work [BPV22], our contributions involve rewarding the agent from a single- and multi-frame perspective, reinforcing closer alignment over both time horizons for improved temporal and spatial coherence. Incorporating learning-based outlier-rejection from prior research [BPV22] enables TrackAgent to distinguish the object region from the background and other objects, making it more robust towards inaccurate segmentation. Dedicated to real-world applications, exploiting information within our pipeline as our uncertainty metrics triggers reinitializations autonomously, thereby balancing the number of reinitializations with the desired accuracy during tracking, enabling the user to strive for highly accurate object poses or minimize computationally expensive reinitializations. Finally, we have demonstrated how our TrackAgent may be employed in a dynamic robotic handover application, enabling the robot to follow the object continuously manipulated by a human and eventually execute a potential object handover.

By evaluating on the YCB-V dataset [XSNF18], our experiments have shown a superiority of TrackAgent compared to all other depth-only baselines. Moreover, we close the gap towards dedicated RGB-D tracking methods. In some object categories, we achieve a higher recall than methods utilizing vast amounts of synthetic data, while ours learns from only a fraction of the YCB-V training data. An extensive ablation study identified early-fusion as too simple, and late-fusion as too complex, pointing to our mid-fusion variant as the optimal architecture for leveraging our joint consideration. Most importantly, fusing the subtasks in our mid-fusion variant also outperforms *frame-to-frame registration* and *frame-to-model refinement* in isolation. Furthermore, the mid-fusion variant's ability to classify inliers and outliers using the segmentation branch proves to be crucial, resulting in the highest ADI AUC score over all other variants. Deploying our reinitialization heuristic contributes to a lower frequency of reinitializations while elevating the resulting

accuracy in tracking, as compared to fixed reinitializations. This suppresses redundant reinitializations of accurately tracked objects, thereby reducing computational effort compared to reinitializing after fixed intervals.

Analysis of the individual subtasks revealed the outlier-rejection to be beneficial for the *frame-to-model refinement* subtask but hinders the efficacy of *frame-to-frame registration*. A promising direction for future work is to investigate the influence of task-specific activation of outlier-rejection, rather than the current approach of universally enabling or disabling it for both encoder branches in the model. TrackAgent's concept of joint solving may guide future RGB methods to improve accuracy by simultaneously registering consecutive color image patches with an RGB rendering as the reference. Heavily inspired by BundleSDF [WTB+23], tracking object poses by (re-)aligning their consecutive depth observations may be exploited as an implicit object reconstruction. Derived from this observation, future researchers may focus on two key aspects - accurately reconstructing objects during the tracking process and exploring the efficacy of replacing the object model with the ongoing reconstruction. Such contributions would eliminate the necessity of an object model prior to tracking, potentially enabling tracking for novel object categories.

List of Figures

List of Tables

List of Algorithms

Bibliography

[AGRL19] Yasuhiro Aoki, Hunter Goforth, Arun Srivatsan Rangaprasad, and Simon Lucey. PointNetLK: Robust & Efficient Point Cloud Registration Using PointNet. In *IEEE/CVF Computer Vision and Pattern Recognition Conference*, pages 7156–7165, 2019.

[AHS22] Mohammad Dawud Ansari, Alwi Husada, and Didier Stricker. Scale Invariant Semantic Segmentation with RGB-D Fusion. *arXiv preprint arXiv:2204.04679*, 2022.

[AMD21] Eduardo Arnold, Sajjad Mozaffari, and Mehrdad Dianati. Fast and Robust Registration of Partially Overlapping Point Clouds. *IEEE Robotics and Automation Letters*, pages 1–8, 2021.

[AS23] Alexander Avery and Andreas Savakis. DeepRM: Deep Recurrent Matching for 6D Pose Refinement. In *IEEE/CVF Computer Vision and Pattern Recognition Conference*, pages 6206–6214, 2023.

[Bau21] Dominik Bauer. *Visually and physically plausible object pose estimation for robot vision.* Dissertation, TU Wien, 2021.

[BJN20] Benjamin Busam, Hyun Jung, and Nassir Navab. I Like to Move It: 6D Pose Estimation as an Action Decision Process. *arXiv preprint arXiv:2009.12678*, 2020.

[BM92] Paul Besl and Neil McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 239–256, 1992.

[BPV20] Dominik Bauer, Timothy Patten, and Markus Vincze. VeREFINE: Integrating Object Pose Verification with Physics-guided Iterative Refinement. *IEEE Robotics and Automation Letters*, pages 4289–4296, 2020.

[BPV21] Dominik Bauer, Timothy Patten, and Markus Vincze. ReAgent: Point Cloud Registration using Imitation and Reinforcement Learning. In *IEEE/CVF Computer Vision and Pattern Recognition Conference*, pages 14586–14594, 2021.

[BPV22] Dominik Bauer, Timothy Patten, and Markus Vincze. Sporeagent: Reinforced Scene-level Plausibility for Object Pose Refinement. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 654–662, 2022.

[BR19] Eric Brachmann and Carsten Rother. Neural- Guided RANSAC: Learning Where to Sample Model Hypotheses. In *International Conference on Computer Vision*, 2019.

[CH06] Francois Chaumette and Seth Hutchinson. Visual Servo Control Part I: Basic Approaches. *IEEE Robotics and Automation Magazine*, pages 82–90, 2006.

[CYX+21] Yu-Wei Chao, Wei Yang, Yu Xiang, Pavlo Molchanov, Ankur Handa, Jonathan Tremblay, Yashraj S. Narang, Karl Van Wyk, Umar Iqbal, Stan Birchfield, Jan Kautz, and Dieter Fox. DexYCB: A Benchmark for Capturing Hand Grasping of Objects. *IEEE/CVF Computer Vision and Pattern Recognition Conference*, pages 9044–9053, 2021.

[DBK+20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv preprint arXiv:2010.11929*, 2020.

[DMX+19] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. PoseRBPF: A Rao-Blackwellized Particle Filter for 6D Object Pose Tracking. In *Robotics: Science and Systems*, 2019.

[DSW+19] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohamad Elbadrawy, Ahsan Lodhi, and Harinandan Katam. BlenderProc. *arXiv preprint arXiv:1911.01911*, 2019.

[ESLG10] Andreas Ess, Konrad Schindler, Bastian Leibe, and Luc Van Gool. Object Detection and Tracking for Autonomous Navigation in Dynamic Environments. *International Journal of Robotics Research*, pages 1707–1725, 2010.

[HJN+20] Matthias Hirschmanner, Ali Jamadi, Bernhard Neuberger, Timothy Patten, and Markus Vincze. Learning Manipulation Tasks from Vision-based Teleoperation. In *Proceedings of the Joint Austrian Computer Vision and Robotics Workshop 2020*, pages 42–47, 2020.

[HLI+12] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes. In *Asian Conference on Computer Vision*, pages 548–562, 2012.

[HMB+18] Tomáš Hodaň, Frank Michel, Eric Brachmann, Wadim Kehl, Anders Glent Buch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, Caner Sahin, Fabian Manhardt, Federico Tombari, Tae-Kyun Kim, Jiří Matas, and Carsten Rother. BOP: Benchmark for 6D Object Pose Estimation. *European Conference on Computer Vision*, 2018.

[HROL20] Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. HOnnotate: A method for 3D Annotation of Hand and Object Poses. In *IEEE/CVF Computer Vision and Pattern Recognition Conference*, pages 3196–3206, 2020.

[ILK+21] Shun Iwase, Xingyu Liu, Rawal Khirodkar, Rio Yokota, and Kris M. Kitani. RePOSE: Fast 6D Object Pose Refinement via Deep Texture Rendering. In *International Conference on Computer Vision*, pages 3303–3312, 2021.

[IWC+16] Jan Issac, Manuel Wüthrich, Cristina Garcia Cifuentes, Jeannette Bohg, Sebastian Trimpe, and Stefan Schaal. Depth-Based Object Tracking Using a Robust Gaussian Filter. *International Conference on Robotics and Automation*, pages 608–615, 2016.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KMI+18] Daniel Kappler, Franziska Meier, Jan Issac, Jim Mainprice, Cristina Garcia Cifuentes, Manuel Wüthrich, Vincent Berenz, Stefan Schaal, Nathan Ratliff, and Jeannette Bohg. Real-Time Perception Meets Reactive Motion Generation. *IEEE Robotics and Automation Letters*, pages 1864–1871, 2018.

[KMT+17] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again. *International Conference on Computer Vision*, pages 1530–1538, 2017.

[LHZA22] Jiayuan Li, Qingwu Hu, Yongjun Zhang, and Mingyao Ai. Robust symmetric iterative closest point. *Journal of Photogrammetry and Remote Sensing*, pages 219–231, 2022.

[LWJ+18] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. DeepIM: Deep Iterative Matching for 6D Pose Estimation. In *European Conference on Computer Vision*, 2018.

[LZZ+22] Xingyu Liu, Ruida Zhang, Chenyangguang Zhang, Bowen Fu, Jiwen Tang, Xiquan Liang, Jingyi Tang, Xiaotian Cheng, Yukang Zhang, Gu Wang, and Xiangyang Ji. GDRNPP, 2022.

[Mar23] Martin Sundermeyer and Tomas Hodan and Yann Labbe and Gu Wang and Eric Brachmann and Bertram Drost and Carsten Rother and Jiri Matas. BOP Challenge 2022 on Detection, Segmentation and Pose Estimation of Specific Rigid Objects. *arXiv preprint arXiv 2302.13075*, 2023.

[MKR+19] Naresh Marturi, Marek Kopicki, Alireza Rastegarpanah, Vijaykumar Rajasekaran, Maxime Adjigble, Rustam Stolkin, Aleš Leonardis, and Yasemin Bekiroglu. Dynamic grasp and trajectory planning for moving objects. *Auton. Robots*, pages 1241–1256, 2019.

[MSL23] Jiageng Mao, Shaoshuai Shi, and Hongsheng Li. 3D Object Detection for Autonomous Driving: A Comprehensive Survey. *International Journal of Computer Vision*, pages 1573–1405, 2023.

[MUS16] Eric Marchand, Hideaki Uchiyama, and Fabien Spindler. Pose Estimation for Augmented Reality: A Hands-On Survey. *IEEE Transactions on Visualization and Compute Graphics*, pages 2633–2651, 2016.

[MVeSTT19] Filipe Monteiro, André Luiz Vieira e Silva, João Marcelo Teixeira, and Veronica Teichrieb. Simulating real robots in virtual environments using NVIDIA's Isaac SDK. pages 47–48, 2019.

[PPL+21] Timothy Patten, Kiru Park, Markus Leitner, Kevin Wolfram, and Markus Vincze. Object Learning for 6D Pose Estimation and Grasping from RGB-D Videos of In-hand Manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4831–4838, 2021.

[QSMG17] Charles Qi, Hao Su, Kaichun Mo, and Leonidas Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *IEEE/CVF Computer Vision and Pattern Recognition Conference*, pages 77–85, 2017.

[Rus19] Szymon Rusinkiewicz. A Symmetric Objective Function for ICP. *ACM Transactions on Graphics*, 2019.

[SSF+22] Yongzhi Su, Mahdi Saleh, Torben Fetzer, Jason Rambach, Nassir Navab, Benjamin Busam, Didier Stricker, and Federico Tombari. ZebraPose: Coarse to Fine Surface Encoding for 6DoF Object Pose Estimation. In *IEEE/CVF Computer Vision and Pattern Recognition Conference*, pages 6728–6738, 2022.

[SST22] Manuel Stoiber, Martin Sundermeyer, and Rudolph Triebel. Iterative Corresponding Geometry: Fusing Region and Depth for Highly Efficient 3D Tracking of Textureless Objects. *IEEE/CVF Computer Vision and Pattern Recognition Conference*, pages 6855–6865, 2022.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[THDT21] Marc Tuscher, Julian Hörz, Danny Driess, and Marc Toussaint. Deep 6-DoF Tracking of Unknown Objects for Reactive Grasping. *International Conference on Robotics and Automation*, pages 14185–14191, 2021.

[VLM18] Joel Vidal, Chyi-Yeu Lin, and Robert Martí. 6D Pose Estimation using an Improved Method based on Point Pair Features. In *International Conference on Control, Automation and Robotics*, pages 405–409, 2018.

[WB21] Bowen Wen and Kostas Bekris. BundleTrack: 6D Pose Tracking for Novel Objects without Instance or Category-Level 3D Models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 8067–8074, 2021.

[WML+21] Gu Wang, Fabian Manhardt, Xingyu Liu, Xiangyang Ji, and Federico Tombari. Occlusion-Aware Self-Supervised Monocular 6D Object Pose Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[WMRB20] Bowen Wen, Chaitanya Mitash, Baozhang Ren, and Kostas E. Bekris. se(3)-TrackNet: Data-driven 6D Pose Tracking by Calibrating Image Residuals in Synthetic Domains. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 10367–10373, 2020.

[WMTJ21] Gu Wang, Fabian Manhardt, Federico Tombari, and Xiangyang Ji. GDR-Net: Geometry-Guided Direct Regression Network for Monocular 6D Object Pose Estimation. In *IEEE/CVF Computer Vision and Pattern Recognition Conference*, pages 16611–16621, 2021.

[WPK+13] Manuel Wüthrich, Peter Pastor, Mrinal Kalakrishnan, Jeannette Bohg, and Stefan Schaal. Probabilistic Object Tracking using a Range Camera. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3195–3202, 2013.

[WS19] Yue Wang and Justin Solomon. Deep Closest Point: Learning Representations for Point Cloud Registration. In *International Conference on Computer Vision*, pages 3522–3531, 2019.

[WTB+23] Bowen Wen, Jonathan Tremblay, Valts Blukis, Stephen Tyree, Thomas Muller, Alex Evans, Dieter Fox, Jan Kautz, and Stan Birchfield. BundleSDF: Neural 6-DoF Tracking and 3D Reconstruction of Unknown Objects. *arXiv preprint arXiv:2303.14158*, 2023.

[XLZ+22] Yan Xu, Kwan-Yee Lin, Guofeng Zhang, Xiaogang Wang, and Hongsheng Li. RNNPose: Recurrent 6-DoF Object Pose Refinement with Robust Correspondence Field Estimation and Pose Optimization. In *IEEE/CVF Computer Vision and Pattern Recognition Conference*, 2022.

[XSNF18] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. In *Robotics: Science and Systems*, 2018.

[YlCJ16] Jiaolong Yang, Hongdong li, Dylan Campbell, and Yunde Jia. Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 2241–2254, 2016.

[ZPK18] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A Modern Library for 3D Data Processing. *arXiv preprint arXiv:1801.09847*, 2018.