OpenDR —
Open Deep Learning Toolkit for Robotics

Project Start Date: 01.01.2020
Duration: 48 months
Lead contractor: Aristotle University of Thessaloniki

Deliverable D5.3: Third report on deep robot action and decision making

Date of delivery: 31 December 2022

Contributing Partners: TUD, ALU-FR, AU, TAU, AUTH
Version: v1.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449.

Title	**D5.3: Third report on deep robot action and decision making**
Project	**OpenDR** (ICT-10-2019-2020 RIA)
Nature	Report
Dissemination Level:	**PU**
Authors	Bas van der Heijden (TUD), Jelle Luijkx (TUD), Laura Ferranti (TUD), Jens Kober (TUD), Robert Babuška (TUD), Anastasios Tefas (AUTH), Nikolaos Nikolaidis (AUTH), Nikolaos Passalis (AUTH), Halil Ibrahim Ugurlu (AU), Lukas Hedegaard Morsing (AU), Erdal Kayacan (AU), Daniel Honerkamp (ALU-FR), Nikolai Dorka (ALU-FR), Roel Pieters (TAU), Akif Ekrekli (TAU)
Lead Beneficiary	TUD (Technische Universiteit Delft)
WP	5
Doc ID:	OPENDR_D5.3.pdf

Document History

Version	**Date**	**Reason of change**
v0.1	1/9/2022	Deliverable structure template ready
v0.2	2/12/2022	Deliverable ready for internal review
v1.0	13/12/2022	Deliverable ready for submission

Contents

Executive Summary

This document presents the status of the work performed between M24 and M36 for **WP5–Deep robot action and decision making**. WP5 consists of four main tasks, that are *Task 5.1–Deep Planning*, *Task 5.2–Deep Navigation*, *Task 5.3–Deep Action and Control*, and *Task 5.4–Human Robot Interaction*.

After a general introduction that provides an overview of the individual chapters with a link to the main objectives of the project, the document dedicates a chapter to each tasks. Each chapter *(i)* provides an overview on the state of the art for the individual topics and existing toolboxes, *(ii)* details the partners' current work in each task with initial performance results, and *(iii)* describes the next steps for the individual tasks. Finally, a conclusion chapter provides a final overview of the work and the planned future work for each individual task.

1 Introduction

This document describes the work done during the third year of the project in the four major research areas of WP5 namely deep planning, deep navigation, deep action and control, and human-robot interactions.

The next sections (Sections 1.1-1.5) provide a summary of the work done so far on these three main topics and the link with the project objectives. The rest of the document is structured as follows. Chapter 2 details our work on deep planning. Chapter 3 describes our work on deep navigation. Chapter 4 presents our work on deep action and control. Chapter 5 presents our work on human robot interaction. Finally, Chapter 6 concludes this deliverable.

More details related to the implementations of the proposed methods in the OpenDR Toolkit can be found in D7.3 (WP7). Details related to evaluation and benchmarking of the proposed methods can be found in D8.2 and D8.3 (WP8).

1.1 Deep Planning (T5.1)

1.1.1 Objectives

Conventional robot motion planning is based on solving individual sub-problems such as perception, planning, and control. On the other hand, end-to-end motion planning methods intend to solve the problem in one shot with less computational cost. Deep learning enables us to learn such end-to-end policies, particularly integrated with Reinforcement learning. AU introduces end-to-end motion planning methods for UAV navigation trained with Deep reinforcement learning.

1.1.2 Innovations and achieved results

AU proposed a novel end-to-end path planning algorithm based on deep Reinforcement learning for aerial robots deployed in dense environments. The learning agent is finding an obstacle-free way around the provided rough global path by only depending on the observations from a forward-facing depth camera. A novel deep reinforcement learning framework is proposed to train the end-to-end policy with the capability of safely avoiding obstacles. Webots open-source robot simulator is utilized for training the policy, introducing highly randomized environmental configurations for better generalization. The training is performed without dynamics calculations through randomized position updates to minimize the amount of data processed. The trained policy is first comprehensively evaluated in simulations involving physical dynamics and software-in-the-loop flight control. The proposed method is proven to have 38% and 50% higher success rate compared to both deep reinforcement learning-based and artificial potential field-based baselines, respectively. The generalization capability of the method is verified in simulation-to-real transfer without further training. Real-time experiments are conducted with several trials in two different scenarios, showing a 50% higher success rate of the proposed method compared to the deep reinforcement learning-based baseline.

1.1.3 Ongoing and future work

There are two possible future directions to improve the presented end-to-end planner. Firstly, AU plans to study the theoretical properties of deep planners, such as stability or convergence.

Secondly, AU intends to combine imagination and curiosity with the current deep planning framework for smoother and shorter path plans.

1.2 Deep Navigation (T5.2)

1.2.1 Objectives

Learning based approaches have shown to be well suited to solve navigation tasks across diverse environments and platforms, including autonomous vehicles, video games and robotics. Particularly deep learning and reinforcement learning approaches have shown to work well with the complex, high-dimensional inputs of real-world environments. Navigation tasks involve both long-horizon goals that require long-term planning as well as local, short-term decision making such as traversing unknown terrain or avoiding static and dynamic obstacles. As a result both the decomposition of the problem into different components and levels of abstraction as well as the combination of traditional optimization and planning approaches with learned modules are very promising approaches.

1.2.2 Innovations and achieved results

Within this year, ALU-FR has extend its previous approach to mobile navigation and manipulation to complex obstacle environments. By generalizing the objective function, extending the control of the reinforcement learning agent and incorporating obstacle sensing into the observation space, the approach becomes applicable to a much wider range of tasks and environments. The approach is demonstrated to achieve high success rates on two real world robots across a large range of unseen and demanding tasks. This work has furthermore won the best paper award at the 2022 IROS Workshop on Mobile Manipulation and Embodied Intelligence. ALU-FR has proposed a novel multi-object search approach. The proposed method unifies short- and long-term reasoning within a single model and time scale. We demonstrate its effectiveness on an HSR robot in our office buildings. The approach is currently being integrated into the OpenDR toolkit. Lastly, ALU-FR has a developed an approach for active localization. The method combines differentiable particle filters with reinforement learning and a hard attention mechanism to actively localize itself in photorealistic environments. AUTH has also developed a data-efficient deep reinforcement learning approach that can improve inertial-based UAV localization, enabling more effective low-cost navigation (Section 3.4).

1.2.3 Ongoing and future work

We are currently extending our work on mobile navigation and manipulation towards task-level reasoning and learned end-effector motions. At the same time, we are extending our work on exploration and object search to more complex environments, in which the agent has to interact with doors or drawers to explore its environment. Simultaneously, we are working on integrating the developed approaches into the OpenDR toolkit.

1.3 Deep Action and Control (T5.3)

1.3.1 Objectives

OpenAI Gym [9] is the standard for training and evaluation of reinforcement learning algorithms. However, developing Gym environments for robot control tasks is an inefficient process that requires expertise, since it is all but trivial to synchronise actions and observations while having asynchronous input and output streams for sensors and actuators that run at different frequencies. In order to facilitate this process TUD to developed EAGERx, a framework that bridges the gap between OpenAI gym and robots, both in simulation and reality. The EAGERx framework aims to separate everything that is engine-specific from everything that is engine-agnostic, such that that environments can be reused for different simulators and even when switching from simulated to real robots.

In order to close the sim2real gap, the challenges of asynchronous control must be addressed. The TUD proposes an accurate delay simulation framework that supports simulating stochastic delays in a deterministic manner (i.e. seeded) that is invariant to the speed of simulation, unlike existing frameworks [3, 29, 40].

Learning-based grasping models typically require a vast amount of training data and training time to train an effective grasp pose detector. Alternatively, small non-generic grasp models have been proposed that are tailored to specific objects by, for example, directly predicting the object's location in 2/3D space, and determining suitable grasp poses by post processing. In both cases, data generation is a bottleneck, as it has to be separately collected for each individual object. Moreover, some approaches require CAD models to generate data and train a model, which are not always available. In this work, TAU has developed a light-weight grasping pipeline that can generate a grasp detection model and execute a grasping action, based in few input images. Object grasp annotation is requested from the operator, and data augmentation is then automatically generated.

Off-policy reinforcement learning is an important research direction as the reuse of old experience promises to make these methods more sample efficient than their on-policy counterparts. This is an important property for many applications such as robotics where interactions with the environment are very time- and cost-intensive. Many successful off-policy methods make use of a learned Q-value function. Accurate estimates of the Q-values are of crucial importance. Unfortunately, learning the Q-function off-policy can lead to bias. To overcome these problems ALU-FR proposes Adaptively Calibrated Critics (ACC) that uses the most recent high variance but unbiased on-policy rollouts to alleviate the bias of the low variance temporal difference targets.

In model-based reinforcement learning (RL) the agent learns a predictive world model to derive the policy for the given task through interaction with its environment. Learning a world model is in principle a supervised learning problem. However, in contrast to the standard supervised learning setting, in model-based RL the dataset is not fixed and given at the beginning of training but is gathered over time through the interaction with the environment which raises additional challenges. A typical problem in supervised learning is overfitting on a limited amount of data. This is well studied and besides several kinds of regularizations a common solution is to track model performance on a validation set that is not used for training. For neural networks a typical behavior is that too few updates lead to underfitting while too many updates lead to overfitting. For learning a world model on a dynamic dataset there unfortunately is no established method to determine if the model is under- or overfitting the training data available at the given point in time. ALU-FR proposed a method that dynamically adjusts the update-to-data (UTD) ratio to balance under- and overfitting based on the performance on an held-out validation set.

1.3.2 Innovations and achieved results

TUD continued the development of the EAGERx framework. The framework now has a consistent interface with an interactive GUI, unit tests with code coverage $> 95\%$, and is accompanied by extensive documentation including a set of 10 interactive tutorials to make it easy for new users to get started. Our documentation, tutorials, and source-code are available at `https://github.com/eager-dev/eagerx`. Furthermore, support for various robots was added and several dissemination activities were undertaken, such as parcipating in two tutorials (ICRA, CCTA) and a summer school.

Also, TUD presented a way of modelling delays in a graph of nodes such that delays can be accurately simulated. This model circumvented the false trade-off between simulation speed that was showed experimentally.

TAU proposed an approach that can utilize state of the art RGB object detection models to develop a real-time robot object grasping pipeline. From few images, annotated by a person, a full dataset is generated that can detect object grasps. Evaluation considers different objects and object detectors with respect to the industrial task of Diesel engine assembly. A collaborative robot (Franka Emika) and standard gripper were utilized for hardware.

ALU-FR developed ACC, a new general algorithm that reduces the bias of value estimates in a principled fashion with the help of the most recent unbiased on-policy rollouts. As a practical implementation, ACC is applied to learn a bias-controlling hyperparameter of the TQC algorithm and the resulting algorithm achieves state of the art results on the OpenAI continuous control benchmark suite as well as on several robotics tasks.

Further, ALU-FR proposed the Dynamic Update-to-Data ratio (DUTD) algorithm that can be applied to most model-based RL algorithms. DUTD detects under- and overfitting of the world model online by evaluating it on hold-out. With this information the UTD ratio is adjusted dynamically to optimize world model performance. DUTD makes tuning the UTD hyperparameter obsolete. Exemplarily, DUTD is applied to a state-of-the art model-based RL method and the results show that it leads to an improved overall performance and higher robustness.

1.3.3 Ongoing and future work

As future work, TUD will make the EAGERx framework compatible with Jax [8] such that EAGERx environments can be run on acceleration hardware. This will greatly improve the speed with which users can train RL agents. Also, we will add a ROS 2 communication backend. Finally, we are preparing a publication that we expect to submit in the beginning of next year.

Also, the TUD will continue the development of a novel phase-synchronized communication protocol that uses the delay model to render delays within the graph to be deterministic (i.e. zero-jitter). We expect this protocol to reduce the effect of jitter on the sim2real performance in RL tasks.

The develop grasping approach by TAU demonstrates that new objects can be grasped by training a light-weight grasp detection model. The current approach utilizes RGB images and depth is assumed to be known. Future work will tackle this by introducing depth sensing of the area around a detected object.

The ACC algorithm was evaluated on simulated environments. A promising direction for future work is to apply it on robots in the real-world.

DUTD in its current form can be applied to model-based RL algorithms. To further allow its application to model-free RL algorithms future work will explore non-supervised objectives that can be used for evaluation on the validation set.

1.4 Human Robot Interaction (T5.4)

1.4.1 Objectives

The interaction between human and robot can serve several functions. When considering a collaborative scenario, the robot can assist a person by taking heavy, dirty or repetitive tasks, relieving the person to a more supervisory or coordinating role. In addition, robots can learn to interact with humans but also from interactions with humans. In both cases understanding human intentions and behavior is crucial, which can by enabled by motion capturing and other ways of instrumenting the human partner. Typically, the human state and intentions are very ambiguous and uncertain (e.g., forces that are crucial for successful completion cannot be estimated from video) and simulations of the human partner and its behavior are unlikely to be very realistic, compounding the uncertainties. Fusing information from multiple modalities will allow the robot to disambiguate. For DRL of human-robot interaction tasks require approaches that can deal with large uncertainties by making optimal use of fused, multimodal perception. Learning from demonstrations, for example, often relies on kinesthetic teach-in, videos, VR, or textual instructions. The objectives of this task are therefore to utilize the OpenDR toolkit towards human-robot interaction and enable efficient and effective collaboration.

1.4.2 Innovations and achieved results

TUD has studied human-robot interaction in the context of language-conditioned interactive imitation learning. We have developed the PARTNR algorithm that identifies ambiguities in the action outputs of vision-based pick and place models. By identifying these ambiguities, we can reduce the number of demonstrations needed from the human, since demonstrations are only asked in case the output is too ambiguous. Furthermore, we improve the trustworthiness of the system by allowing users to specify a desired sensitivity level. In this way, the user can balance between unnecessary queries and execution failures. TAU has studied human-robot collaboration in the context of manufacturing, where a small collaborative robot assists an operator in assembly tasks. Several DL-based vidual perception modules were selected, as suitable for effective interaction and for scene understanding. These included human detection, human action recognition and object detection/pose recognition. For human detection and action recognition pretrained models were utilized which achieves suitable performance in experiments. For object detection, a custom dataset was generated, which included eight object and target classes, as part of the industrial assembly task. All perception was integrated in a robot control framework based on ROS and provided in the OpenDR toolkit for replication (and extension) of our work. Results demonstrate that DL-based perception models can be easily trained and deployed to robotic environments and achieve reliable detection and recognition results. Results also demonstrated that multiple perception models can be utilized simultaneously, enabling the fusion of different sensors or utilizing different detection modules in parallel.

1.4.3 Ongoing and future work

As ongoing work, TUD is developing the language-conditioned interactive imitation learning algorithm PARTNR. Initial results look promising and will be extended to more evaluation environments and evaluated as future work. This includes multiple simulated benchmark tasks as well as a real-world participant study. Furthermore, we are currently integrating a tool related to the PARTNR algorithm in the OpenDR toolkit.

As ongoing work from TAU, the human-robot collaboration scenario has utilized the tools developed in WP3 (human detection and human action recognition) and WP4 (object detection and pose estimation). Initial results with individual modules look promising. Future work will include the fusion of different perception modules to achieve more robust and different recognition. For example, multiple sensor measurements and modalities (human detection plus action recognition) can be utilized to recognize the state of a person collaborating with a robot, such the recognition action relevant to the industrial task at hand.

1.5 Connection to Project Objectives

The work performed within WP5, as summarized in the previous subsections, perfectly aligns with the project objectives. More specifically, the conducted work progressed the state-of-the-art towards meeting following objectives of the project:

O2.c *To provide lightweight deep learning methods for deep robot action and decision making, namely:*

O2.c.i Deep reinforcement learning (RL) and related control methods.

- The zero-jitter communication protocol proposed by TUD in Chapter 4 will reduce the effect of timing and delays on the sim2real performance.
- The EAGERx toolkit presented by TUD in Chapter 4 enables users to use a single pipeline for both the real and simulated environment. Hence, this reduces the chance for mismatches between the two implementations. Therefore, EAGERx facilitates the application of deep RL methods in practice.
- ALU-FR contributed to this objective as described in Chapter 3. ALU-FR developed several reinforcement learning approaches that can remain lightweight and can run directly on robot's onboard CPUs. Further, the Adaptively Calibrated Critic estimates algorithm presented by ALU-FR allows for improved value estimates which is a fundamental problem in model-free RL. Lastly, ALU-FR proposed a method that improves the overall data efficiency of model-based RL algorithms by dynamically adjusting the update-to-data ratio. This makes such approaches more applicable to robots in the real-world.

O2.c.ii Deep planning and navigation methods that can be trained in end-to-end fashion.

- AU contributed to this objective as described in Chapter 2. An end-to-end planner for a quadrotor UAV is provided for collision avoidance in dense environments. The DRL agent, informed by the global trajectory, generates actions as local position plans based on depth images. The efficiency and efficacy of the planner are evaluated over several cluttered field scenarios.

- ∗ AUTH contributed to this objective as described in Chapter 3. AUTH developed an end-to-end trainable model that can increase the precision of interial-based localization and navigation for UAVs.
- ∗ ALU-FR contributed to this objective as described in Chapter 3. ALU-FR extended and generalized its mobile navigation and manipulation tool, making it applicable to a very wide range of tasks in complex and human-centered obstacle environments. ALU-FR furthermore developed an object-search approach that unifies short- and long-term reasoning in a single model and an active localization approach that scales to continuous action spaces and large maps.

O2.c.iii Enable robots to decide on actions based on observations in WP3, as well as to learn from observations.

- ∗ TUD has contributed to this objective as described in Chapter 4 with the graph structure of environments in the EAGERx toolkit. This functionality allows the user to use the perception algorithms in WP3 as nodes.
- ∗ TUD has contributed to this objective as described in Chapter 4. Perception methods are known to be computationally expensive. Hence, they may introduce unwanted communication delays that may affect the performance of control methods that rely on the output of these methods. The zero-jitter communication protocol discussed by TUD mitigates the effect of delays on the sim2real performance of reinforcement learning policies, hence enabling robots to decide on action based on the outputs of methods in WP3.
- ∗ TAU has contributed to this objective as described in Chapter 4. An object grasping model has been developed, extended and evaluated that takes as input few images of an object, annotations by a person and generates the required training data to train a object grasping model. Several grasp detection methods are evaluated for grasping object relevant in the agile production use case.
- ∗ ALU-FR contributed to this objective as described in Chapter 3. ALU-FR extended its mobile navigation and manipulation approach to incorporate sensed obstacle maps of the environment. For object-search, ALU-FR proposed an approach that learns based on semantic maps of the environment, enabling data-efficient learning. Lastly, for active localization ALU-FR proposes an approach that can handle a wide variety of sensors as inputs for its decision making.

O2.c.iv Enable efficient and effective human robot interaction

- ∗ TUD has contributed to this objective as described in Chapter 5, by developing the PARTNR algorithm that identifies ambiguities and thereby improves the data efficiency of vision-based imitation learning methods.
- ∗ TAU has contributed to this objective as described in Chapter 5, where human detection and human action recognition tools from WP3 are utilized to enable collaboration between human and robot, in agile production scenario. In addition, object detection tools from WP4 are utilized to detect eight objects and targets, for robot pick-and-place actions and robot-to-human hand-over tasks. All tools are integrated into a single experimental scenario enabling that sensor information can be fused.

2 Deep Planning

2.1 Safe End-to-end Path Planning of Aerial Robots in Dense Environments

2.1.1 Introduction and objectives

Autonomous aerial robots are increasingly deployed in applications that require safe path planning in dense environments, such as a greenhouse covered with dense plants, search & rescue operation in an unstructured collapsed building, or navigation in a forest. Traditionally, autonomous navigation is solved under separate problems such as state estimation, perception, planning, and control [32]. This approach may lead to higher latency combining individual blocks and system integration issues. On the other hand, recent developments in machine learning, particularly in reinforcement learning (RL) and deep reinforcement learning (DRL), enable an agent to learn various navigation tasks end-to-end with only a single neural network policy that generates required robot actions directly from sensory input. These methods are promising to solve navigation problems computationally faster since they do not deal with the integration of subsystems that are tuned for their particular goals.

This study attempts to address the end-to-end planning problem of a quadrotor UAV in dense indoor environments. The quadrotor deployed with a depth camera is required to find its way around the global trajectory. We propose a DRL-based safe navigation methodology for quadrotor flight. The learned DRL policy, utilizing the depth images and the knowledge of a global trajectory, generates safe waypoints for the quadrotor. We develop a Webots-based simulation environment where the DRL agent is trained with obstacle tracks where the obstacle locations, shapes, and sparsity are randomized for every episode of policy training for better generalization. Furthermore, we introduce safety boundaries to be considered during training besides collision checks. The safety boundaries enable the agent to prevent risky situations that make the method more robust to uncertainties.

The contributions of this study are fourfold:

- A novel DRL simulation framework is proposed for training an end-to-end planner for quadrotor flight, including a faster training strategy using non-dynamic state updates and highly randomized simulation environments.
- The impact of continuous/discrete actions and proposed safety boundaries in RL training are investigated.
- We open-source the Webots-based DRL framework, including all training and evaluation scripts.
- The method is evaluated with extensive experiments in Webots-based simulation environments and multiple real-world scenarios, transferring the network from simulation to real without further training.

2.1.2 Description of work performed so far

The details of this work are found in the corresponding publication that is listed below, and can be found in Appendix AU-Appendix:

- H. I. Ugurlu, H. X. Pham and E. Kayacan "*Sim-to-Real Deep Reinforcement Learning for Safe End-to-End Planning of Aerial Robots.*" Robotics 2022, 11, 109.

In this study, a novel end-to-end path planning algorithm based on deep Reinforcement learning is proposed for aerial robots deployed in dense environments. The learning agent is finding an obstacle-free way around the provided rough global path by only depending on the observations from a forward-facing depth camera. A novel deep reinforcement learning framework is proposed to train the end-to-end policy with the capability of safely avoiding obstacles. Webots open-source robot simulator is utilized for training the policy, introducing highly randomized environmental configurations for better generalization. The training is performed without dynamics calculations through randomized position updates to minimize the amount of data processed. The trained policy is first comprehensively evaluated in simulations involving physical dynamics and software-in-the-loop flight control. The proposed method is proven to have 38% and 50% higher success rate compared to both deep reinforcement learning-based and artificial potential field-based baselines, respectively. The generalization capability of the method is verified in simulation-to-real transfer without further training. Real-time experiments are conducted with several trials in two different scenarios, showing a 50% higher success rate of the proposed method compared to the deep reinforcement learning-based baseline.

2.1.3 Future work

One possible future research direction is on the theoretical properties of deep planners, such as stability or convergence. The deep learning-based methods are criticized for not having theoretical guarantees compared to conventional methods. Hence, the research of theoretical methods to prove the capabilities of deep planners is gaining more attention. Another future direction is combining the strengths of model-based (such as MPC) and model-free (such as DRL) methods to obtain an advantage of both. Further, it is also possible to work on model-learning based approaches. Besides that, One issue in the convergence of the neural network learning based approach could be converging to local-maxima in term of reward-maximization, however, using methods such as curiosity (as an intrinsic reward) it is possible to find global-maxima by enhancing the state exploration.

3 Deep Navigation

3.1 N^{2}M^2: Learning Navigation for Arbitrary Mobile Manipulation Motions in Unseen and Dynamic Environments

3.1.1 Introduction and objectives

We have previously introduced our approach for mobile navigation and manipulation [22], developed within the first years of Open. In this work, we extend our method to incorporate obstacle avoidance, extend the control of the reinforcement learning agent to the torso and velocity of the target motions and generalize the reward function. This results in a powerful approach that is applicable to a much wider range of tasks in complex environments and across a wide variety of kinematically diverse robots.

3.1.2 Description of work performed so far

Details of this work can be found in the publication listed below, which is also provided in Appendix A. The work furthermore won the best paper award of the IROS Workshop on Mobile Manipulation and Embodied Intelligence, 2022.

- [23] D. Honerkamp, T. Welschehold and A. Valada, "*N2M2: Learning Navigation for Arbitrary Mobile Manipulation Motions in Unseen and Dynamic Environments*", in IROS Workshop on Mobile Manipulation and Embodied Intelligence, Oct. 2022, *DOI: 10.48550/ARXIV.2206.08737.*

We extend and generalize our previous work as follows: First, we extend the observations of reinforcement learning agent to its surroundings by incorporating the obstacle maps into its observation space. This modality promises to generalize well to unseen scenes due to its geometric nature and can be constructed from a variety of sensors, making it applicable across robots. Secondly, we extend the agent's control to the torso lift joints and to the norm of the end-effector velocities. This increases its flexibility to navigate complex obstacle maps and to slow down the target end-effector motions whenever the base of the robot has to maneouver more extensively. Lastly, we devise a training scheme in a procedurally generated obstacle map.

In extensive experiments on three simulated and two real-world robotic platforms we demonstrate that the resulting approach and solve a wide variety of complex tasks involving articulated objects in cluttered environments. In this, the approach generalizes to unseen tasks, motions and environments, demonstrating its robustness and flexibility.

3.1.3 Future work

In current work, we are extending this work to jointly or iteratively learn the desired end-effector motions as well as to incorporate 3D obstacle avoidance. This will further scale its capabilities and reduce human inputs to a bare minimum.

3.2 Learning Long-Horizon Robot Exploration Strategies for Multi-Object Search in Continuous Action Spaces

3.2.1 Introduction and objectives

Recent advances in vision-based navigation and exploration have shown impressive capabilities in photo realistic indoor environments. However, these methods still struggle with long-horizon tasks and require large amounts of data to generalize to unseen environments. In this work, we present a novel reinforcement learning approach for multi-object search that combines short-term and long-term reasoning in a single model while avoiding the complexities arising from hierarchical structures. In contrast to existing multi-object search methods that act in granular discrete action spaces, our approach achieves exceptional performance in continuous action spaces.

3.2.2 Description of work performed so far

Details of this work can be found in the publication listed below, which is also provided in Appendix B:

- [35] F. Schamlstieg, D. Honerkamp, T. Welschehold and A. Valada, "*Learning Long-Horizon Robot Exploration Strategies for Multi-Object Search in Continuous Action Spaces*", in Proceedings of the International Symposium on Robotics Research (ISRR), Sept. 2022, *DOI: 10.48550/ARXIV.2205.11384*.

Our approach learns to predict the direction of the path towards the closest target object. It then learns a policy that observes this prediction, enabling it to express long-term intentions while taking short-term actions based on the full context. As a consequence, the policy can incorporate expected inaccuracies and uncertainties in the predictions and balance strictly following its intentions with short-term exploration and collision avoidance. We perform extensive experiments and show that it generalizes to unseen apartment environments with limited data. Furthermore, we demonstrate zero-shot transfer of the learned policies to an office environment in real world experiments.

3.2.3 Future work

In the future, we aim to further exploit the ability of the approach to learn different actions at a high control frequency, such as controlling the camera of the robot. Furthermore, we are currently extending this work to a hierarchiecal approach which extends to more complex tasks in which the robot has to interact with objects such as door, drawers or cabinets to move around the environment and find the target objects.

3.3 Active Particle Filter Networks: Efficient Active Localization in Continuous Action Spaces and Large Maps

3.3.1 Introduction and objectives

Accurate localization is a critical requirement for most robotic tasks. The main body of existing work is focused on passive localization in which the motions of the robot are assumed given, abstracting from their influence on sampling informative observations. While recent work has shown the benefits of learning motions to disambiguate the robot's poses, these methods are restricted to granular discrete actions and directly depend on the size of the global map. We propose Active Particle Filter Networks (APFN), an approach that only relies on local information for both the likelihood evaluation as well as the decision making. To do so, we couple differentiable particle filters with a reinforcement learning agent that attends to the most relevant parts of the map. The resulting approach inherits the computational benefits of particle filters and can directly act in continuous action spaces while remaining fully differentiable and thereby end-to-end optimizable as well as agnostic to the input modality.

3.3.2 Description of work performed so far

Details of this work can be found in the publication listed below, which is also provided in Appendix C:

- [21] D. Honerkamp, S. Guttikonda and A. Valada, "*Active Particle Filter Networks: Efficient Active Localization in Continuous Action Spaces and Large Maps*", in IROS 2022 Workshop Probabilistic Robotics in the Age of Deep Learning, Oct. 2022, *DOI: 10.48550/ARXIV.2209.09646*.

We present an approach that couples probabilistic and learning-based methods through learned particle filters and deep reinforcement learning (RL) to generalize to continuous action spaces and arbitrary sensor modalities independent of map size. Particle filters enable efficient representation of multi-modal beliefs over large maps. These mechanisms can be made fully differentiable enabling us to learn the components of a particle filter end-to-end, thereby extending it to abstract observations such as pixels or depth maps. Importantly, these networks only need to process local information for each particle. We then train a reinforcement learning agent that selects actions to minimize the overall localization error, following the same principle of processing only local information over the most likely hypotheses through a hard attention mechanism. In contrast to previous work, this enables us to process hypotheses over continuous poses while at the same time breaking the dependency on processing the full map with a neural network. We demonstrate the benefits of our approach with extensive experiments in photorealistic indoor environments built from real-world 3D scanned apartments.

3.3.3 Future work

In future work, we aim to extend the approach to simultaneously control sensors such as actuated cameras, which promises to benefit even more from active perception. Another promising avenue is the extension of learning-based localization and attention mechanisms to dynamic environments and noisy, partial or incorrect maps in which it becomes important to selectively filter out uncertain or incorrect observations. Lastly, the trade-off between active localization and other task objectives is an exciting direction for future work.

3.4 Improving Inertial-based UAV Localization using Data-efficient Deep Reinforcement Learning

3.4.1 Introduction and objectives

Unmanned Aerial Vehicles (UAVs) are increasingly used in various applications, ranging from precision agriculture [33] and search and rescue missions [5] to indoor surveillance [7]. A common point between these applications, along with virtually every UAV-based application, is the need for precise UAV localization. UAV localization is critical both for mission control purposes, i.e., some tasks are related to the location of a UAV, as well as for safety purposes, i.e., avoid flights over restricted areas. Several different approaches have been developed for UAV localization, with each one relying on different sensors and providing a different level of accuracy.

Perhaps among the most well known localization approaches is using satellite-based radionavigation systems, such as the Global Positioning System (GPS) [37, 18]. Despite its low cost the accuracy of1.2.4 GPS and related systems is usually low. Indeed, according to the official GPS documentation, GPS-enabled devices are normally accurate to within a 4.9 meters (16 feet), which is unacceptable for many applications. At the same time, there are several locations where there is no GPS coverage [1], while such approaches cannot be used indoors. The use of real-time kinematic positioning can further reduce the errors introduced in satellite-based radio navigation [19], yet it typically requires the use of extra base stations, which increases the cost and reduces the flexibility of UAVs. Light detection and ranging approaches [25, 15], also known as LIDAR, can be also used to provide accurate localization, especially when coupled with simultaneous localization and mapping (SLAM) approaches [39]. However, such approaches involve the use of very expensive sensors and they have greater computational and energy demands.

On the other hand, the use of Inertial Measurement Units (IMUs) [4], which is a combination of accelerometers, gyroscopes, and magnetometers can provide very low-cost solutions that also do not rely on any kind of external hardware or communication (e.g., satellites, base stations, etc.). The localization is accomplished by utilizing IMU data for dead reckoning, called Inertial Navigation System (INS) [6]. The recent demand for smaller sensors that can be integrated into cutting-edge technologies, has prompted engineers to build a Micro Electro-Mechanical System (MEMS) which can provide low-cost and low-footprint sensors that can be very easily integrated with virtually any UAV and provide real-time measurements. Despite the cost and flexibility benefits of such systems, they also come with accuracy limitations. IMU sensors monitor the linear acceleration and rotational velocity of the body with just a very small degree of inaccuracy every time. However, over long periods these errors can accumulate leading to significant position drifts that can comprise their application, especially when used as a sole localization sensor in mission critical applications.

These limitations have fueled research on methods for improving inertial-based navigation for UAVs [10, 12, 20]. Many recent approaches built upon Deep Learning (DL)-based models that allow for significantly improving the localization process. However, despite these improvements, these approaches suffer from a significant drawback. They mostly rely on supervised learning (either regression-based or classification-based), which in turn requires a large number of samples to be collected and annotated to train the corresponding methods. At the same time, such approaches are typically linked to the hardware used for data collection and their performance deteriorates when deployed on different hardware, requiring collecting data again and re-training the models. Furthermore, even when using the same hardware, manufacturing tolerances might lead to sensors that have different noise characteristics, which make the application of supervised learning approaches challenging.

Deep Reinforcement Learning (DRL) can overcome these limitations [11], since it enables autonomous agents to learn just by interacting with the environment. Indeed, DRL methods have shown to achieve remarkable results in a variety of tasks in recent years, often outperforming humans [31, 36]. However, directly applying DRL for improving inertial-based navigation for UAVs is not directly feasible since: a) a feedback signal is still required in order to measure the quality of the learned policy and b) a large number of episodes are typically required for learning. Even though the first limitation can be easily addressed, e.g., by using visual cues to provide a feedback signal, the low-data efficiency of DRL approaches still pose a significant limitation that prohibits such approaches from being deployed in practice.

3.4.2 Description of work performed so far

Based on the aforementioned observations, in this work we propose a pipeline that can allow for easing these limitations, enabling data-efficient DRL on UAVs for inertial-based navigation. The proposed method employs a two-stage pipeline. In the first stage, a backbone is trained using supervised learning in a simulator. Acquiring ground truth annotations in a simulator is easy and cheap, so this approach can enabled us to train a backbone that can capture the dynamics of the behavior of IMUs without targeting a specific sensor. Then, the employed DL model is fine-tuned using DRL on a real UAV. Since this can be an especially data-intensive process, we further propose: a) a data augmentation method that can generate multiple simulated episode trajectories just from one real episode and b) a regularizer than can provide additional feedback when fine-tuning the learned policy based on the sign of the measured reward signal. For acquiring a reward signal, we propose a simple, yet efficient visual landmark-based approach that can be used even with low-resolution cameras. As we demonstrated through extensive experiments on regressing the 2D position of a UAV, the proposed method can indeed lead to significant performance improvements over the employed baseline approaches.

The technical report, along with the full results, are provided in Appendix E:

E D. Tsiakmakis, N. Passalis, and A. Tefas. "*Improving Inertial-based UAV Localization using Data-efficient Deep Reinforcement Learning*", Technical Report (AUTH), 2022.

3.4.3 Future work

This work has demonstrated that the proposed method can indeed allow for improving inertial-based navigation, focusing on cases where the IMUs used in UAVs can have different characteristics requiring UAV-specific fine-tuning using a very small number of real episodes. These results highlight the potential of such methods for other navigation tasks as well. At the same time, using more accurate approaches for calculating the reward signal, is expected to further increase the accuracy of the developed method. Finally, significant improvements in data efficiency were obtained by employing data augmentation approaches, hinting into another interesting research direction.

4 Deep action and control

4.1 EAGERx

4.1.1 Introduction and objectives

Reinforcement learning (RL) methods have received much attention due to impressive results. Despite significant interest in RL in recent years, many works with RL in robotics are done in simulation only. While RL promises learning-based control of near-optimal behaviors in theory, successful real-world learning can elude practitioners due to various implementation challenges

Prior works on real-world robot learning used a variety of environmental instrumentation in an effort to obtain state information, define reward functions, and define manually designed reset routines between episodes [17, 28]. Furthermore, real-world learning is not only prohibitively expensive, but it can also introduce issues related to safety, such as damages to the equipment or human operators. Therefore, learning a control policy may also be done in simulation. Simulators void the need for environmental instrumentation because they can provide perfect state information and allow for arbitrary state resets. Moreover, learning in simulation can be faster than real-time, inexpensive, and safer.

To this end, we developed a novel robotics framework called EAGERx (Engine Agnostic Graph Environments for Robotics). EAGERx can be used to built complex distributed robotic systems as graph structures of interconnected nodes similar to ROS. We circumvent the false trade-off between simulation speed and accuracy with a novel algorithm that synchronizes inter-node communication. EAGERx further narrows the sim2real gap with native support for essential features such as domain randomization and delay simulation. Moreover, its graph structure allows users to easily extend the simulator with custom models for physical phenomena that are inaccurately modeled (e.g. motor dynamics, friction models). The framework promotes code reuse and enables users to easily define new tasks, switch from one sensor to another, and switch from simulation to reality by being agnostic to the physics-engine. A single RL pipeline that works with both simulated and real robots eliminates the chance for unintended mismatches when two separate software implementations are used. EAGERx is compatible with established RL packages such as Stable Baselines [34], because it follows the OpenAI Gym API.

4.1.2 Description of work performed so far

Last year, we redeveloped EAGERx to be reactive for synchronized communication. Also, we performed initial experiments for a simple task (manipulator reference tracking) in order to validate the core functionalities of the toolkit. Finally, we presented a first rudimentary version of an interactive GUI that could visualize the constructed graph.

This year, we focused a lot on improving the user-friendliness and maintainability of EAGERx. The framework now has a consistent interface with an interactive GUI 1, unit tests with code coverage $>95\%$, and is accompanied by extensive documentation (>80 pages) including a set of interactive tutorials to make it easy for new users to get started. Our documentation, tutorials, and source-code are available at `https://github.com/eager-dev/eagerx`.

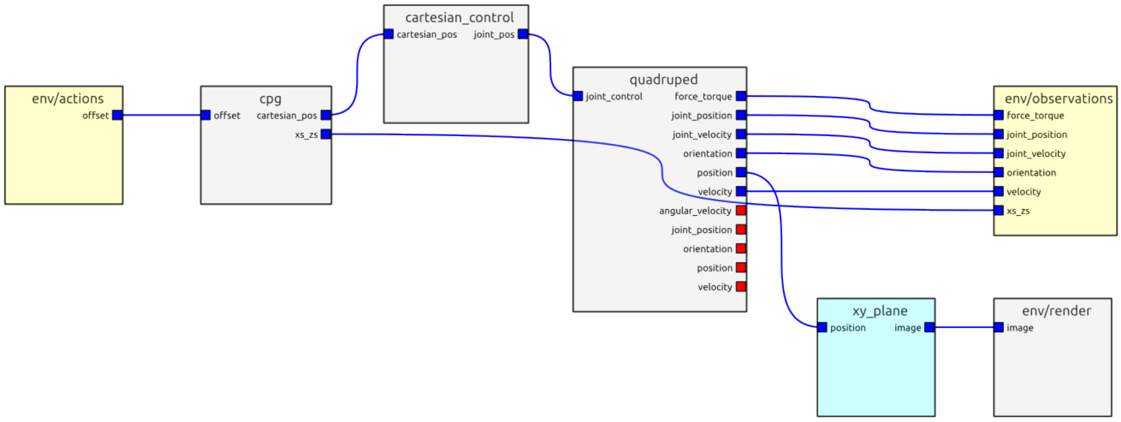


Figure 1: The graphical user interface allows users to easily inspect the environment they created. The graphical user interface is based on the PyQtGraph library [2].

We continued developing EAGERx such that different communication backends could be used. This allows users to install EAGERx via a simple "pip install eagerx" command, and alleviates the need for a full installation of ROS. This also clears the way for ROS2 support. This also allowed the 10 tutorials to be set up as interactive python notebooks, which was previously difficult due to dependency on ROS.

Last year, we have disseminated EAGERx in tutorial sessions (ICRA, CCTA with over 800 attendees) and presented the framework at the summer school "Continuous Engineering and Deep Learning for Trustworthy Autonomous Systems" at the Aristotle University of Thessaloniki.

Finally, we added support for several robots in Pybullet (quadrupeds, manipulators, pendulum) and the real-world (manipulators, pendulum). In the experimental setup shown in Figure

2, we trained a box-pushing policy in Pybullet, that successfully transferred to the real-world. This demonstrates the effectiveness of EAGERx in solving complex robotic learning tasks.

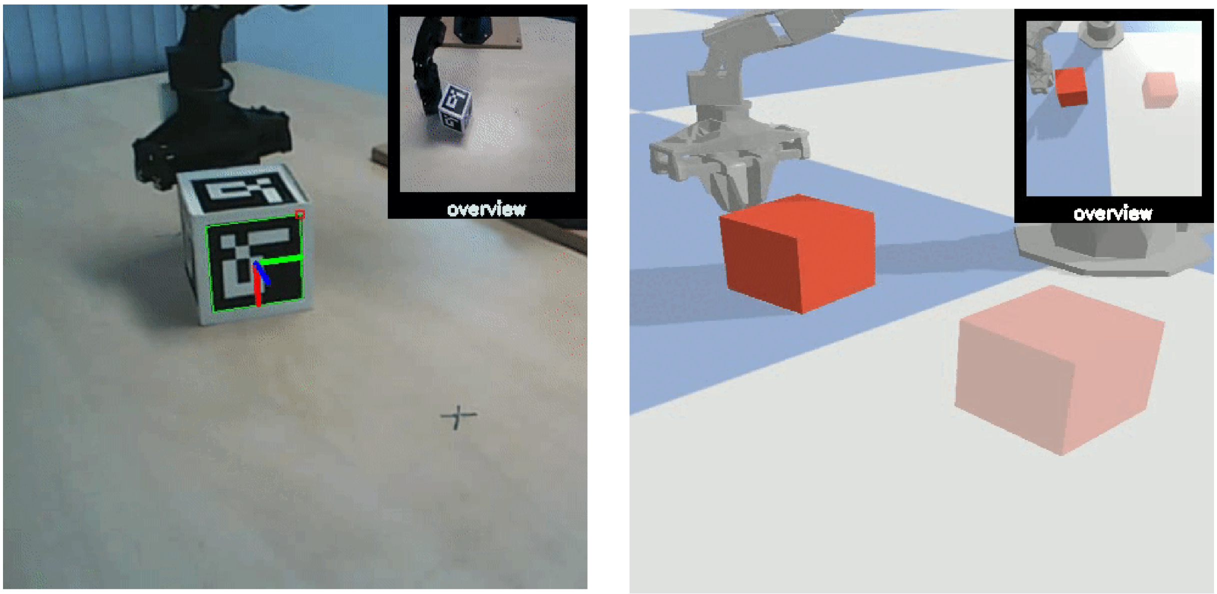


Figure 2: Policies trained in simulation and zero-shot evaluated on real systems using EAGERx. Here we trained a box-pushing policy in pybullet (right), that successfully transferred to the real-world (left).

4.1.3 Future work

As future work, we will make the EAGERx toolkit compatible with Jax such that EAGERx environments can be run on acceleration hardware. This will greatly improve the speed with which users can train RL agents. Also, we will add a ROS 2 communication backend. Finally, we are preparing a publication that we expect to submit in the beginning of next year.

4.2 Zero-Jitter Communication Protocol for Improved Sim2Real Policy Transfer

4.2.1 Introduction and objectives

The transfer of simulated robot learning to the real world is termed “sim2real” and recent works have shown promising results on real-world problems [26, 38]. Nevertheless, a gap in performance is often observed between simulation and reality due to unaccounted differences between the simulator and reality. These differences may arise due to inaccurate modeling of physical phenomena, such as friction, collision, and deformation and are collectively called the “sim2real gap”. RL methods are notorious for exploiting (or even overfitting on) these differences to maximize the simulated rewards. Prior work [38] has shown that a randomization of the dynamic properties of a simulator during training can mitigate the negative effect of simulator inaccuracies. This technique is also referred to as domain randomization.

A subtle but important challenge in sim2real is that of asynchronous control [24]. The Markov decision process (MDP) formulation in RL assumes synchronous execution: the observed state remains unchanged until the action is applied. While robotic systems are typically simulated synchronously, sensing and acting happen simultaneously and asynchronously in the real-world. Consequently, the sim2real performance may drop dramatically in the presence of asynchronous control, because the agent may always use the current observations in simulation, while observations turn out to be systematically outdated in the real-world due to computation and communication delays.

Computation and communication delays can be naturally included by simulating the robotic system asynchronously. However, such delays are usually inherent to the system and independent from the simulation speed. Consequently, the effect of delays on the simulated behavior may unintentionally increase ten fold when the simulation runs ten times faster than real-time. Hence, asynchronous simulation introduces a false trade-off between speed and accuracy if no form of input/output synchronization between asynchronous components is ensured in the simulated robotic system.

Existing robotic learning frameworks [3, 29, 40] are often built on-top of ROS and its scalable publisher-subscriber software architecture that allows easy switching between simulation and reality. While these frameworks use most of the components that comprise the real-world robotic system to close the sim2real gap, they unintentionally instigate the previously mentioned trade-off between simulation speed accuracy when asynchronous components are also included.

To this end, we present a way of modelling delays in a graph of nodes such that delays can be accurately simulated that circumvents the false trade-off between simulation speed. We also propose a novel communication protocol that minimizes the effect of jitter (varation in delays) on the sim2real performance by leveraging the delay model.

4.2.2 Description of work performed so far

We were able to show the effect of asynchronous simulation on the simulated accuracy in a simple simulation experiment of a pendulum. Figures 3 and 4 show that the synchronized simulation is fully deterministic irrespective of the real-time factor, in contrast to the variability in the asynchronous case that increases with the real-time factor.

By modelling the delays within a graph of nodes as an inter-connection of computation and communication delays we are able to infer the phase shift for every node. This is shown in Figure 5. This allows us to propagate delays through the network, irrespective of the real-time factor.

4.2.3 Future work

As future work, we will use the modeled delays in a novel phase-synchronized communication protocol that renders all delays to be deterministic (i.e. zero-jitter). We expect this protocol to reduce the effect of jitter on the sim2real performance in RL tasks.

4.3 Single demonstration grasping

4.3.1 Introduction and objectives

Collaborative robots have gained popularity in industry as they are designed to be safe, particularly where human and robot share the workspace. Accompanied by intuitive programming interfaces, robot tasks can be programmed efficiently. Despite the benefits, the application of cobots in industrial settings are mainly limited to offline tasks where the actions and targets

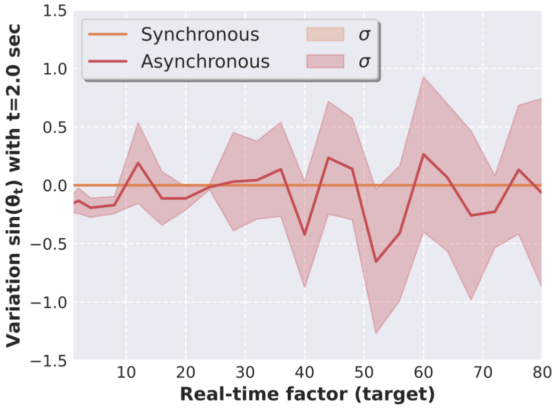


Figure 3: The variation in angle $\sin(\theta)$ of a simulated pendulum as a function of the real-time factor. The pendulum is simulated at 15 Hz and driven by a voltage sequence that is identical over episodes. Commands and angle measurements are send and received asynchronously in *async* mode. We measure the variation in angle $\sin(\theta)$ difference with respect to a synchronized simulation at $t = 2.0$ seconds over 5 episodes. The synchronized simulation is fully deterministic irrespective of the real-time factor, in contrast to the variability in the asynchronous case that increases with the real-time factor.

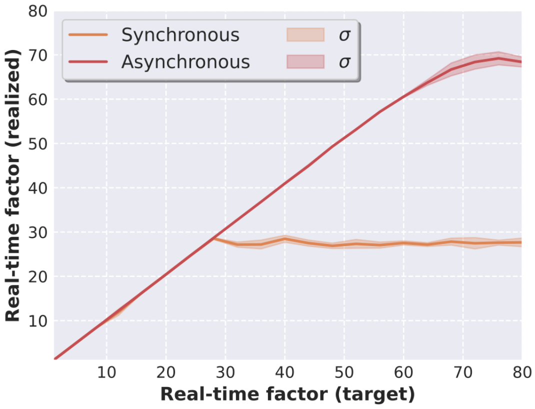


Figure 4: The real-time factor of the simulation. The realized real-time factor for synchronous simulations naturally plateaus in order to stay synchronized. The realized real-time factor for the asynchronous simulations is higher, but this apparent gain in speed is deceiving. Figure 3 reveals that the increased speed comes at a cost of higher variability. The asynchronous components are increasingly out of sync even though all asynchronous components may realize their target real-time factor.

are defined to the system beforehand. For example, in the majority of pick and place tasks, object poses are fixed, and the robotic arm should reach a predefined grasp pose. Although there is great interest in the generation of object grasp models from visual data, limitations still exist, for example, in terms of object type coverage, grasp success, training complexity, model inference time, etc. In particular, while grasp models have reported high success rate, this typically only holds for the task at hand, i.e., bin picking with generic household items. Evaluating such grasping model on objects that exhibit different properties (e.g., industrial parts) might re-

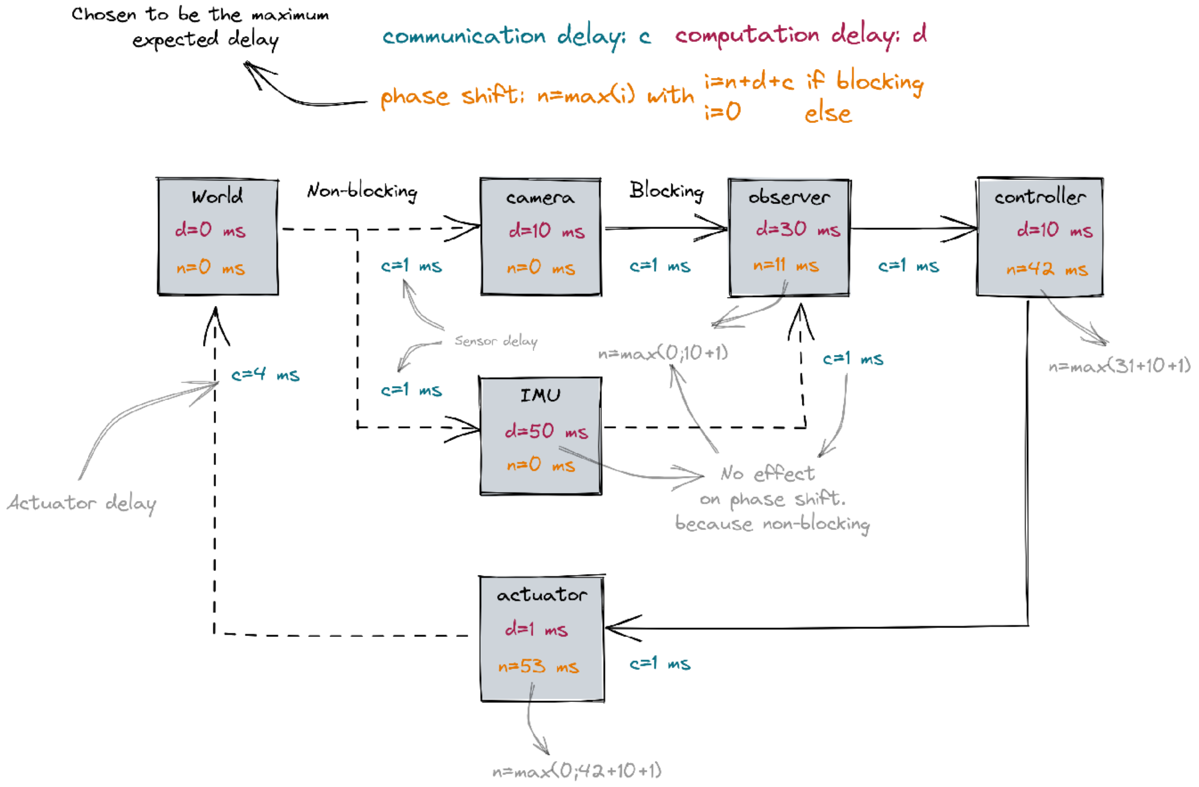


Figure 5: Here is a schematic overview of how delays can be modeled within a graph of interconnected nodes.

sult in unsuccessful grasp attempts and an overall lower accuracy. In addition, grasp modelling requires vast amounts of training data and considerable training time on high-performance computing clusters. Consequently, state of the art grasping models can be large in size and slow to execute.

4.3.2 Description of work performed so far

The details of this work can found in the publication listed below, and can be found in Appendix F:

- A. Mehman Sefat, A. Angleraud, E. Rahtu and R. Pieters “*SingleDemoGrasp: Learning to Grasp From a Single Image Demonstration*”, in IEEE Conference on Automation Science and Engineering (CASE), 2022, pp. 390-396,
  DOI: 10.1109/CASE49997.2022.9926463.

In this work, we aimed to tackle this issue by investigating visual learning-based approaches for object grasp detection, with human annotation of a desired object grasp pose. For this, different variants of the R-CNN architecture from Detectron2 are evaluated for the fast generation of a grasping model. Single or multiple image demonstrations with human annotations of an object grasp are collected and utilized to generate an augmented object training dataset, from which a detection model is trained. Object grasp detection results (object grasp position and orientation on a plane) are transformed to a 3D grasp pose and given as input for robot motion planning. Four different networks are developed and evaluated in simulation (Webots) with eight different objects. The grasp detection model with best performance was then implemented and evaluated in real robot experiments (Franka robot with standard gripper).

4.3.3 Future work

The presented work demonstrates that the grasping pipeline can generate suitable data in order to train a light-weight model and successfully grasp objects with a robot. Even though the objects used are relevant for the agile production use case, a relatively small set of objects was evaluated. In addition, the depth information for grasping was assumed to be known. Future work will include depth sensing to enable the grasping of objects that are located on surfaces with uknown height.

4.4 Adaptively Calibrated Critic Estimates for Deep Reinforcement Learning

4.4.1 Introduction and objectives

Accurate value estimates are important for off-policy reinforcement learning. Algorithms based on temporal difference learning typically are prone to an over- or underestimation bias building up over time. Especially when a nonlinear function approximator is used to model the Q-function, there are many potential sources of bias. Different heuristics were proposed for their mitigation, such as the double estimator in the case of discrete action spaces or taking the minimum of two estimates in the case of continuous actions. While these methods successfully prevent extreme overestimation, due to their coarse nature, they can still induce under- or overestimation bias to a varying degree depending on the environment.

4.4.2 Description of work performed so far

Details of this work can be found in the publication listed below, which is also provided in Appendix G.

- [13] Dorka, N., Welschehold, T. Boedecker, J., & Burgard, W., "*Adaptively Calibrated Critic Estimates for Deep Reinforcement Learning*", arXiv preprint arXiv:2111.12673 (2022).

We propose a general method called Adaptively Calibrated Critics (ACC) that uses the most recent high variance but unbiased on-policy rollouts to alleviate the bias of the low variance temporal difference targets. We apply ACC to Truncated Quantile Critics [27], which is an algorithm for continuous control that allows regulation of the bias with a hyperparameter tuned per environment. The resulting algorithm adaptively adjusts the parameter during training rendering hyperparameter search unnecessary and sets a new state of the art on the OpenAI gym continuous control benchmark among all algorithms that do not tune hyperparameters for each environment. ACC further achieves improved results on different tasks from the Meta-World robot benchmark. Additionally, we demonstrate the generality of ACC by applying it to TD3 [16] and showing an improved performance also in this setting.

4.4.3 Future work

In future we plan to evaluate the effectiveness of ACC applied to algorithms that work with discrete action spaces and when learning on a real robot where tuning of hyperparameters is very costly.

4.5 Dynamic Update-to-Data Ratio: Minimizing World Model Overfitting

4.5.1 Introduction and objectives

Early stopping based on the validation set performance is a popular approach to find the right balance between under- and overfitting in the context of supervised learning. However, in reinforcement learning, even for supervised sub-problems such as world model learning, early stopping is not applicable as the dataset is continually evolving. For learning a world model on a dynamic dataset there unfortunately is no established method to determine if the model is under- or overfitting the training data available at the given point in time. Additionally, in model-based RL a poorly fit model can have a dramatic effect onto the learning result as from it the agent derives the policy, which influences the future collected experience which again influences the learning of the world model. So far, in model-based RL this is commonly addressed with some form of regularization and by setting an update-to-data (UTD) ratio that specifies how many update steps the model does per newly collected experience, similar to selecting the total number of parameter updates in supervised learning. Analogously to supervised learning, a higher UTD ratio is more prone to overfit the data and a lower one to underfit it. State-of-the-art methods set the UTD ratio at the beginning of the training and do not base the selection on a dynamic performance metric. Unfortunately, tuning this parameter is very costly as the complete training process has to be traversed several times. Furthermore, a fixed UTD ratio is often sub-optimal because different values for this parameter might be preferable at different stages of the training process.

4.5.2 Description of work performed so far

Details of this work can be found in the publication listed below, which is also provided in Appendix H.

- [14] Dorka, N., Welschehold, T. & Burgard, W., "*Dynamic Update-to-Data Ratio: Minimizing World Model Overfitting*", In Decision Awareness in Reinforcement Learning Workshop at ICML 2022.

We propose a new general method that dynamically adjusts the UTD ratio during training based on underand overfitting detection on a small subset of the continuously collected experience not used for training. We apply our method to DreamerV2, a state-of-the-art model-based reinforcement learning algorithm, and evaluate it on the DeepMind Control Suite and the Atari 100k benchmark. The results demonstrate that one can better balance under- and overestimation by adjusting the UTD ratio with our approach compared to the default setting in DreamerV2 and that it is competitive with an extensive hyperparameter search which is not feasible for many applications. Our method eliminates the need to set the UTD hyperparameter by hand and even leads to a higher robustness with regard to other learning-related hyperparameters further reducing the amount of necessary tuning.

4.5.3 Future work

In future work we plant to explore non-supervised objectives for model-free RL algorithms that can be used for evaluation on the validation set. This would allow the usage of DUTD to adjust the UTD ratio of such algorithms.

5 Human robot interaction

5.1 PARTNR

5.1.1 Introduction and objectives

Several recent works show impressive results in mapping language-based human commands and image scene observations to direct robot executable policies (e.g., pick and place poses). However, these approaches do not consider the uncertainty of the trained policy and simply always execute actions suggested by the current policy as the most probable ones. This makes them vulnerable to domain shift and inefficient in the number of required demonstrations. We extend previous works and present the PARTNR (Pick and place Ambiguity Resolving by Trustworthy iNteractive leaRning) algorithm that can detect ambiguities in the trained policy by analyzing multiple modalities in the pick and place poses using topological analysis. PARTNR employs an adaptive, sensitivity-based, gating function that decides if additional user demonstrations are required. User demonstrations are aggregated to the dataset and used for subsequent training. In this way, the policy can adapt promptly to domain shift and it can minimize the number of required demonstrations for a well-trained policy. The adaptive threshold enables to achieve the user-acceptable level of ambiguity to execute the policy autonomously and in turn, increase the trustworthiness of our system. We demonstrate the performance of PARTNR in a table-top pick and place task.

5.1.2 Description of work performed so far

Our submission for the Robot Learning Workshop at NeurIPS 2022 was accepted and can be found in Appendix J [30]. Additional material is available at partnr-learn.github.io.

5.1.3 Future work

In the future, we plan to evaluate PARTNR with the original CLIPort baseline as well and to further address the epistemic uncertainty of the model, e.g., through an ensemble approach. Also, we wish to extend the method with sequence prediction and feedback control. Finally, we plan to monitor the human cognitive load in a real-world participant study. Also, we will finalize the integration of this tool within the OpenDR toolkit.

5.2 Sensor-based Human-Robot Collaboration

5.2.1 Introduction and objectives

Collaborative robots (co-bots) can improve the safety, work efficiency and productivity of industrial processes by acting as flexible and reconfigurable tool to human operators. Within

Industry 4.0, co-bots have a core role to contribute to the transition from traditional manufacturing to digital manufacturing. Co-bots can be easily programmed and reconfigured, and are safe for interaction, due to their small form-factor and incorporated sensor systems that can detect collisions. Co-bots are also to be found in high-payload form, where protective covering can be complemented by sensor-based safety features. Human-robot collaboration is typically possible in two ways: 1. Off-line programming of robot tasks by demonstration (also known as hand-guiding or kinesthetic teaching), and 2. On-line interaction between human and cobot, enabled by external sensor systems. While off-line programming is an established method of collaboration, on-line interaction still typically requires great efforts in development and its success depends highly on the sensor system. That is, if the external sensor system is not robust or has high latency, this reflects negatively on the performance of the collaboration. The practical requirements and tools needed, however, are often underestimated or given little attention, resulting in great interest from industry and SMEs, but not many practical implementations. To be realistic, successful integration of perception tools in human-robot collaboration requires considerable effort towards the selection of suitable detection tools, the preparation of suitable data for training, and the actual training of a detection model.

5.2.2 Description of work performed so far

The details of this work can found in the draft publication listed below, and can be found in Appendix I:

- A. Ekrekli, A. Angleraud, K. Samarawickrama, G. Sharma, R. Pieters "*Sensor-based Human-Robot Collaboration for Industrial Tasks*", in preparation, 2023.

In this work we address the current limitations in perception models and situational awareness for industrial human-robot collaboration. Perception and situational awareness of robot systems can be enhanced, such that fluent and responsive collaboration between human and robot is possible. We believe that perception models, based on deep learning, are ideal for this, as they can be accurate, reliable and fast to execute. These can then provide the required sensory input for interaction, such as the human body and its pose, human actions or gestures, and the pose of objects and targets in the scene. Developing and integrating such models for robotics in industry are hard tasks, often requiring expertise from many different areas. Therefore, we additionally provide a general HRC software framework, based on ROS, which can be utilized to replicate our developments. The framework is build around OpenDR and has the perception tools integrated for a practical and industrially relevant use case in agile production. The visual perception tools are human skeleton detection, human action recognition and the detection and pose estimation of objects and targets in the scene.

5.2.3 Future work

In future work we will aim to extend the approach to sensor fusion of multiple perception modalities, to increase robustness and include redundancy for perception. This should lead to more reliable perception of the human state, as well as provide robust situational awareness.

6 Conclusions

This document presented the work performed on WP5. After a short introduction on the work done on the individual tasks, the document provided a detailed overview of the individual tasks, as detailed below.

Chapter 2 presented the status of the work performed for Task 5.1–Deep Planning. AU presented an end-to-end planner trained with DRL for safe navigation in cluttered obstacle environments. The end-to-end planning algorithm is trained and tested in comprehensive simulations developed in Webots. While the training of the policy network is handled without dynamics and control to save time, it is successfully sim-to-real transferred for physical evaluations. Moreover, safety boundaries for training are introduced, which successfully prevents the quadrotor from being in hazardous situations. The method is also deployed in real-world indoor environments successfully. The end-to-end planner outperforms a baseline implementation based on the artificial potential field method, which has a lower success rate, especially in cluttered obstacle settings. This shows that SCDP has learned to make better long-term decisions. The real-world experiments demonstrate that the proposed UAV planner trained solely with simulation can directly work in a real environment.

Chapter 3 detailed the status of the work performed for Task 5.2–Deep Navigation. ALU-FR introduced a more general version of its approach for mobile manipulation, which largely extends the tasks and environments this approach can solve. The resulting approach has been recognized with the best paper award of the IROS Workshop on Mobile Manipulation and Embodied Intelligence, 2022. ALU-FR furthermore developed a novel multi-object search approach that unifies short- and long-term reasoning in a single model. This approach is currently being integrated into the OpenDR toolkit. Lastly, ALU-FR has developed an active localization methods that combines differentiable particle filters with reinforcement learning to scale to large maps and continuous action spaces.

Chapter 4 highlighted the work performed for Task 5.3–Deep Action and Control. First, TUD introduced the progress made on the EAGERx toolkit. The framework now has a consistent interface with an interactive GUI, unit tests with code coverage $> 95\%$, and is accompanied by extensive documentation including a set of 10 interactive tutorials to make it easy for new users to get started. Our documentation, tutorials, and source-code are available at `https://github.com/eager-dev/eagerx`. Second, TUD also presented a delay simulation framework that allows delays to be accurately simulated in simulators that run faster than real-time. Furthermore, a novel communication protocol was proposed that reduces the effect of jitter on the sim2real performance. TAU has made improvements to the SingleDemoGrasp tool, which can generate the required training data from a single or object image demonstrations. The general code quality is improved and made consistent with the toolkit. In addition, the tool now includes data annotation and augmentation functionalities as part of the toolkit, and the functionality to utilize different visual detection modules from the Detection_2D tool.

Finally, Chapter 5 highlighted the work performed for Task 5.4–Human Robot Interaction. The chapter covered the following topics. TUD developed an interactive imitation learning algorithm named PARTNR that exploits an ambiguity measure to improve data-efficiency and trustworthiness. This work was accepted for the 5th Robot Learning Workshop at NeurIPS 2022. We are currently working on the integration of a tool in the OpenDR toolkit related to this work. Also, we plan to perform more extensive evaluation both in simulation as on the real system. TAU has developed a collaborative scenario between human and robot, which is inspired by the Agile Production use case. Perception tools from the OpenDR toolkit are utilized to enable the robot to act as assistant to the human, by functionalities such as automated pick-and-place and robot to human hand-overs. The visual perception tools utilized are human skeleton detection, human action recognition and the detection and pose estimation of objects and targets in the scene. Performance of the tools is tested for different input image sizes. As developing and integrating such models for robotics in industry are hard tasks, often requiring expertise from many different areas, we additionally provide basic HRC software templates, which can be utilized to replicate our developments.

References

[1] GPS Accuracy. https://www.gps.gov/systems/gps/performance/accuracy/. Accessed: 2022-07-21.

[2] PyQtGraph: Scientific Graphics and GUI Library for Python. https://www.pyqtgraph.org/. Accessed: 2021-11-29.

[3] robo-gym – an open source toolkit for distributed deep reinforcement learning on real and simulated robots. *IEEE International Conference on Intelligent Robots and Systems*, pages 5364–5371, 7 2020.

[4] N. Ahmad, R. A. R. Ghazilla, N. M. Khairi, and V. Kasi. Reviews on various inertial measurement unit (imu) sensor applications. *International Journal of Signal Processing Systems*, 1(2):256–262, 2013.

[5] E. T. Alotaibi, S. S. Alqefari, and A. Koubaa. Lsar: Multi-uav collaboration for search and rescue missions. *IEEE Access*, 7:55817–55832, 2019.

[6] B. Barshan and H. F. Durrant-Whyte. Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation*, 11(3):328–342, 1995.

[7] N. Boonyathanmig, S. Gongmanee, P. Kayunyeam, P. Wutticho, and S. Prongnuch. Design and implementation of mini-uav for indoor surveillance. In *Proceedings of the 9th International Electrical Engineering Congress (iEECON)*, pages 305–308, 2021.

[8] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[9] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[10] M. Brossard, A. Barrau, and S. Bonnabel. Rins-w: Robust inertial navigation system on wheels. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2068–2075, 2019.

[11] J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. *Proceedings of the Advances in Neural Information Processing Systems*, 31, 2018.

[12] S. Cortés, A. Solin, and J. Kannala. Deep learning based speed estimation for constraining strapdown inertial navigation on smartphones. In *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2018.

[13] N. Dorka, J. Boedecker, and W. Burgard. Adaptively calibrated critic estimates for deep reinforcement learning. *arXiv preprint arXiv:2111.12673*, 2021.

[14] N. Dorka, T. Welschehold, and W. Burgard. Dynamic update-to-data ratio: Minimizing world model overfitting. In *Decision Awareness in Reinforcement Learning Workshop at ICML 2022*.

[15] F. G. Fernald. Analysis of atmospheric lidar observations: some comments. *Applied Optics*, 23(5):652–653, 1984.

[16] S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[18] S. Han and J. Wang. Integrated gps/ins navigation system with dual-rate kalman filter. *GPS Solutions*, 16(3):389–404, 2012.

[19] P. Henkel, U. Mittmann, and M. Iafrancesco. Real-time kinematic positioning with gps and glonass. In *Proceedings of the 24th European Signal Processing Conference (EUSIPCO)*, pages 1063–1067, 2016.

[20] S. Herath, H. Yan, and Y. Furukawa. Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, & new methods. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3146–3152, 2020.

[21] D. Honerkamp, S. Guttikonda, and A. Valada. Active particle filter networks: Efficient active localization in continuous action spaces and large maps. *IROS 2022 Workshop Probabilistic Robotics in the Age of Deep Learning*, 2022.

[22] D. Honerkamp, T. Welschehold, and A. Valada. Learning kinematic feasibility for mobile manipulation through deep reinforcement learning. *IEEE Robotics and Automation Letters*, 6(4):6289–6296, 2021.

[23] D. Honerkamp, T. Welschehold, and A. Valada. N^{2}m^2: Learning navigation for arbitrary mobile manipulation motions in unseen and dynamic environments. *IROS 2022 Workshop on Mobile Manipulation and Embodied Intelligence*, 2022.

[24] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.

[25] M. Jaboyedoff, T. Oppikofer, A. Abellán, M.-H. Derron, A. Loye, R. Metzger, and A. Pedrazzini. Use of lidar in landslide investigations: a review. *Natural Hazards*, 61(1):5–28, 2012.

[26] J. E. Kooi and R. Babuška. Inclined quadrotor landing using deep reinforcement learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2361–2368. IEEE.

[27] A. Kuznetsov, P. Shvechikov, A. Grishin, and D. Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International Conference on Machine Learning*, pages 5556–5566. PMLR, 2020.

[28] A. X. Lee, A. Nagabandi, P. Abbeel, and S. Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33:741–752, 2020.

[29] N. G. Lopez, Y. Leire, E. Nuin, E. B. Moral, L. Usategui, S. Juan, A. S. Rueda, M. Vilches, R. Kojcev, and A. Robotics. gym-gazebo2, a toolkit for reinforcement learning using ros 2 and gazebo. 3 2019.

[30] J. Luijkx, Z. Ajanović, L. Ferranti, and J. Kober. PARTNR: Pick and place Ambiguity Resolving by Trustworthy iNteractive leaRning. In *5th NeurIPS Robot Learning Workshop: Trustworthy Robotics*, 2022.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[32] H. X. Pham, H. I. Ugurlu, J. Le Fevre, D. Bardakci, and E. Kayacan. Deep learning for vision-based navigation in autonomous drone racing. In *Deep Learning for Robot Perception and Cognition*, pages 371–406. Elsevier, 2022.

[33] P. Radoglou-Grammatikis, P. Sarigiannidis, T. Lagkas, and I. Moscholios. A compilation of uav applications for precision agriculture. *Computer Networks*, 172:107148, 2020.

[34] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 2021.

[35] F. Schmalstieg, D. Honerkamp, T. Welschehold, and A. Valada. Learning long-horizon robot exploration strategies for multi-object search in continuous action spaces. *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2022.

[36] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[37] S. Sukkarieh, E. M. Nebot, and H. F. Durrant-Whyte. A high integrity imu/gps navigation loop for autonomous land vehicle applications. *IEEE Transactions on Robotics and Automation*, 15(3):572–578, 1999.

[38] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

[39] D. Van Nam and K. Gon-Woo. Solid-state lidar based-slam: A concise review and application. In *Proceedings of th IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 302–305, 2021.

[40] I. Zamora, N. G. Lopez, V. M. Vilches, and A. H. Cordero. Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo. *arXiv preprint arXiv:1608.05742*, 2016.

A

N^{2}M^2: Learning Navigation for Arbitrary Mobile Manipulation Motions in Unseen and Dynamic Environments

Daniel Honerkamp, Tim Welschehold, and Abhinav Valada

***Abstract*—Despite its importance in both industrial and service robotics, mobile manipulation remains a significant challenge as it requires a seamless integration of end-effector trajectory generation with navigation skills as well as reasoning over long-horizons. Existing methods struggle to control the large configuration space, and to navigate dynamic and unknown environments. In previous work, we proposed to decompose mobile manipulation tasks into a simplified motion generator for the end-effector in task space and a trained reinforcement learning agent for the mobile base to account for kinematic feasibility of the motion. In this work, we introduce Neural Navigation for Mobile Manipulation (N^{2}M^2) which extends this decomposition to complex obstacle environments and enables it to tackle a broad range of tasks in real world settings. The resulting approach can perform unseen, long-horizon tasks in unexplored environments while instantly reacting to dynamic obstacles and environmental changes. At the same time, it provides a simple way to define new mobile manipulation tasks. We demonstrate the capabilities of our proposed approach in extensive simulation and real-world experiments on multiple kinematically diverse mobile manipulators. Code and videos are publicly available at http://mobile-rl.cs.uni-freiburg.de.**

***Index Terms*—Mobile Manipulation, Robot Learning, Embodied AI, Reinforcement Learning**

I. Introduction

While recent progress in control and perception has propelled the capabilities of robotic platforms to autonomously operate in unknown and unstructured environments [1]–[4], this has largely focused on pure navigation tasks [5], [6]. In this work, we focus on autonomous mobile manipulation which combines the difficulties of navigating unstructured, human-centered environments with the complexity of jointly controlling the base and arm. Mobile Manipulation is commonly reduced to sequential base navigation followed by static arm manipulation at the goal location. This simplification is restrictive as many tasks such as door opening require the joint use of the arm and base and is inefficient as it dismisses simultaneous movement and requires frequent repositioning.

Mobile manipulation requires a range of capabilities including collision-aware navigation, object interactions, manipulation, exploration of unknown environments, and long-term reasoning. As a result, approaches from a variety of areas such as planning, optimal control, and learning have been proposed. Inverse reachability map (IRM) approaches seek good base positioning for the robot base given the task constraints [7], [8] but often require expert knowledge and can be overly restrictive. Planning approaches [9]–[11] come with asymptotic optimality guarantees, but scale unfavorably with the size of the configuration space, can have long planning times, and often require frequent re-planning in dynamic environments. Model predictive control (MPC) formulations explicitly define and optimize over a range of collision and desirability constraints, and recently achieved strong results in mobile manipulation [12], [13]. However, they are computationally expensive, often do not optimize past a limited horizon, and can struggle when objectives oppose each other. Learning-based methods efficiently learn directly from high-dimensional observations and are well suited to handle unexplored environments [14]–[16]. Nevertheless, they either restrict the action space [14], [17] or rely on expert demonstrations [15] to cope with the high-dimensional action space and long-horizon nature of mobile manipulation. Furthermore, the learned behavior is often task-specific, requiring re-training for each novel task.

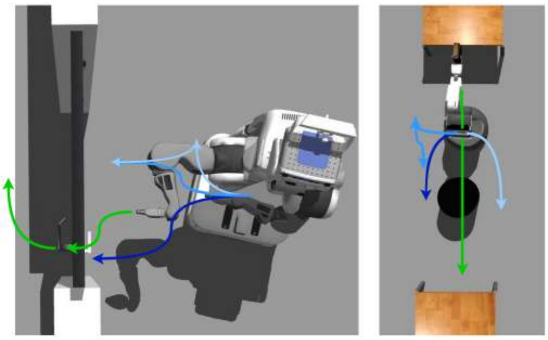


Fig. 1. Mobile manipulation tasks in unstructured environments typically require the simultaneous use of the robotic arm and the mobile base. While it is comparably simple to find end-effector motions to complete a task (green), defining base motions (blue) that conform to both the robot's and the environment's constraints is highly challenging. We propose an effective approach that learns feasible base motions for arbitrary end-effector motions. The resulting model is flexible, dynamic and generalizes to unseen motions and tasks.

In this work, we formulate mobile manipulation as a goal conditional reinforcement learning (RL) problem in which the RL agent observes the end-effector motion and goal, and aims to ensure that these motions remain kinematically feasible. Fig. 1 depicts a high-level overview of our approach. We extend our formulation introduced in [18] to unstructured

All authors are with the Department of Computer Science, University of Freiburg, 79110 Germany (e-mail: honerkamp@cs.uni-freiburg.de, twelsche@cs.uni-freiburg.de valada@cs.uni-freiburg.de).

obstacle environments and increase the agent's freedom to control the velocity of the end-effector motions and torso joints. This provides a very simple and yet effective way to define new tasks, as the RL agent takes care of all the complexities regarding collision-free navigation and kinematic feasibility. The resulting approach, which we term N^2M^2 (Neural Navigation for Mobile Manipulation), efficiently learns to solve long-horizon tasks, lasting thousands of steps, generalizes to unseen tasks and environments in a zero-shot manner, and reacts instantaneously to changes in the environment without any planning times. Lastly, the approach is directly applicable to a wide range of kinematics including both holonomic and non-holonomic robotic bases. We show that with appropriate action regularization, our hybrid approach can be trained without a complex simulator and directly transfers to the real world. We demonstrate these capabilities in both extensive simulation and real-world experiments on multiple mobile manipulators, across a wide range of tasks and environments.

This paper makes the following main contributions:

1) We formulate the fulfillment of kinematic feasibility constraints for mobile manipulation tasks in the presence of obstacles as a reinforcement learning problem.
2) We propose a reactive approach to learn complementary mobile manipulator base motions for arbitrary end-effector motions on unstructured obstacle maps.
3) We develop a procedurally generated training task and an approach to generate end-effector motions that can be used across a variety of mobile manipulators with largely varying kinematics and driving models.
4) We demonstrate the capabilities of our approach in extensive simulated and real-world experiments on unseen environments, obstacles, and tasks.
5) We make the code publicly available at http://mobile-rl.cs.uni-freiburg.de.

The remainder of the paper is organized as follows: Sec. II discusses existing literature and approaches for mobile manipulation. Sec. III describes the technical details of our method, Sec. IV then evaluates its capabilities and compares it to previous approaches. Lastly, Sec. V discusses limitations and future work and concludes.

II. Related Work

Mobile manipulation tasks require the composition of a vast range of capabilities spanning perception, control and exploration. In the following, we discuss previous methods from planning, optimal control, and learning to tackle these challenges.

Sequential navigation and manipulation: Due to the difficulties of planning in the conjoint space of the mobile manipulator base and arm, many existing approaches restrict themselves to sequential movements of the base followed by static manipulations with the arm. This decomposition has been popular across approaches based on reachability [8], planning [1], [19], [20], impedance control [21], and reinforcement learning [14], [22].

Planning: To ensure kinematic feasibility in mobile manipulation tasks, planning-based approaches plan trajectories for the robot in joint space and as such only explore kinematically feasible paths [11], [23]. Sampling-based approaches such as rapidly exploring random trees (RRT) and their variants have shown to perform well in high-dimensional spaces and come with certain optimality guarantees. However, increasing their configuration spaces and environments can result in long planning times or far from optimal solutions. Operation in unobserved or dynamic environments can trigger costly re-planning if the environment changes [24], [25]. While certain constraints such as fixed orientations or task space regions [26] can be incorporated well to plan motions such as opening a door [20], the incorporation of arbitrary (end-effector) constraints is difficult and often requires expert knowledge and task-specific adaptations. In contrast, our approach can near-instantly react to dynamic changes, offers a natural way to incorporate unexplored environments, and can be directly applied to arbitrary end-effector motions.

Inverse Reachability maps [7] can be used to seek good positioning for the robot base given the task constraints [8], subsequently the task is solved stationary. Combinations with planning methods exist [27], but it remains a hard problem to integrate kinematic feasibility constraints in task space mobile motion planning. Welschehold *et al.* [28] treat the kinematic feasibility of arbitrary gripper trajectories as an obstacle avoidance problem based on a geometric description of the inverse reachability. They analytically modulate the base velocity such that the base stays within feasible regions and orientations relative to the end-effector.

Optimal Control approaches have demonstrated promising results on complex manipulation tasks that require conjoint movements of arm and base. Model predictive control (MPC) based approaches have demonstrated strong performance on whole-body control tasks such as door opening [13], [29], obstacle avoidance [12], and articulated objects [30]. Constraints are explicitly incorporated into the objective function and optimized over a (usually fixed) rollout horizon. While efficient implementations can incorporate horizons of up to several seconds in near real-time, these approaches still require significant compute. Moreover, they also require simplified collision objects to achieve this and often do not take into account consequences beyond the rollout horizon. In contrast, our approach learns a value function reflective of the full episode horizon and can perform fast inference with a single forward pass, without reliance on highly optimized implementations. Furthermore, it does not rely on explicit representations of the environment and offers direct extensions to arbitrary input modalities and partially observed environments.

Haviland *et al.* [31] propose a reactive controller for both holonomic and non-holonomic bases, however, they do not demonstrate any collision avoidance. Redundancy resolution methods specify desirable aspects of different solutions through additional constraints or parameters [32]. In contrast, our approach directly learns to resolve redundancies with regard to long-term optimality, removing the need to specify explicit desirability of different choices.

Task and motion planning (TAMP) combines low-level motion planning and high-level task reasoning and has resulted in approaches that generalize to many robots and environments [33]–[35]. At the same time, it can be computationally demanding, depends on the specification of symbolic actions, and does not generally incorporate uncertainties or partial observability as it requires full information about the 3D structure of the environment. In contrast, we assume that we can generate the desired end-effector motions, but may act in unexplored or dynamically changing environments.

Obstacle Avoidance: Learning-based methods have shown to be effective for learning obstacle avoidance from high dimensional sensor inputs such as color images [36], [37], LiDAR [38], [39], depth images [40] or a combination thereof [41]. These approaches have demonstrated the ability to avoid dynamic obstacles such as pedestrians [39], [41], [42]. Sensors such as LiDAR and depth sensors have also shown good transfer from simulation into the real world [40], [42]. Several approaches have explored the use of 2D obstacle maps to learn a local planner component to avoid pedestrians based on a 2D LiDAR [39] or to predict a collision map from a 2D obstacle map based on a laser scan and binary collision events [38]. Optimal control approaches have successfully used voxel map representations [43] and signed distance fields [12], [30], [44].

Reinforcement Learning: Recently mobile manipulation has also been tackled as a reinforcement learning task. Relmogen [14] learns subgoals for arm and base, but relies on sequential execution of each and pre-specified pushing motions. Kindle *et al.* [17] learn to directly control both arm and base, however, they restrict the arm movements to a plane. Jauhri *et al.* [22] learn a base positioning and a discrete decision on whether to use the arm. While these approaches learn effective policies for specific tasks, they cannot easily be applied to novel tasks. In contrast, we address the problem of enabling arbitrary end-effector trajectories, providing a straightforward technique to define novel tasks with arbitrary motion constraints in task space and the capability for zero-shot generalization to such unseen tasks at test time.

Recently several approaches and benchmarks for complex long-horizon tasks have been proposed. Most similar to our tasks is MoMaRT, which uses imitation learning to control a Fetch robot in simulation and in mostly seen environments [15]. In contrast, we focus on generalization to completely unseen environments as well as the real-world. ManipulaTHOR focus on the higher-level task aspects, using a strongly simplified robot arm and magic grasping action [45]. Rearrangement-style benchmarks define tasks as moving the environment from an initial to a goal state [46]. BEHAVIOR [47] and Habitat-2.0 [48] instantiate a large number of tasks in simulation. These approaches often abstract from the actual kinematics with actions such as ”magical” grasping actions that immediately succeed [45], [47], [48] or explicitly provided lower-level motion primitives [47], [49]. In contrast, we assume that the agent is aware of the high-level task goals in form of access to an end-effector motion, but has to learn to achieve them with the actual robotic kinematics. Secondly, these approaches achieve limited success even when training separate agents specific to each task. We provide an approach that generalizes to unseen tasks without any re-training or finetuning. A very interesting aspect will be to incorporate and combine our work with the higher-level task focus of these benchmarks.

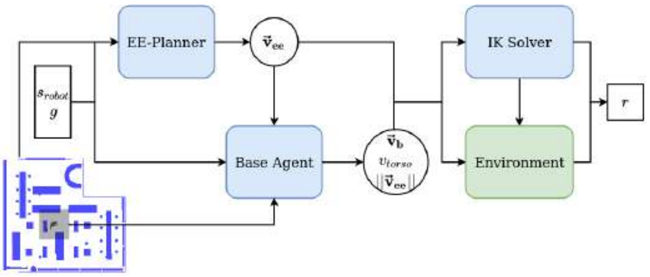


Fig. 2. N^{2}M^2: We decompose mobile manipulation tasks into two components: an end-effector (EE) motion generation and a conditional RL agent that controls the base, the torso lift joint and the velocity of the end-effector motions. The agent receives a local map of the environment (shown in grey) together with the robot state and the next end-effector motions. Given the agent's actions, an inverse kinematics solver then solves for the remaining arm joints.

We formulate our approach as goal conditional RL problem [50], [51] which conditions its policy on a goal state to arrive at this goal. Hierarchical methods [52]–[54] abstract complex long-horizon tasks into a high-level policy proposing subgoals and goal-conditional low-level policies. While adapting these methods for mobile manipulation can improve sample efficiency [55], it still has to deal with the complexity and non-stationarity of learning hierarchical policies.

Articulated Objects have been a particular focus for mobile manipulation. Mittal *et al.* [30] estimate the articulation parameters of unseen objects, then use this to generate keyframes for the end-effector. Kineverse constructs articulation models for complex kinematics and demonstrates how these can be used to generate motions [56]. These works provide simple ways to generate end-effector motions for a wide variety of objects, complementing our approach and making it even more broadly applicable.

III. Neural Navigation for Mobile Manipulation

Mobile manipulation in real-world environments is a complex long-horizon task that combines navigation and control in a high-dimensional action space while respecting constraints imposed by the task, the hardware, and the environment. At the core, we decompose the mobile manipulation problem into an end-effector motion and base velocities learned by a reinforcement learning agent with the goal to ensure that these concurrent end-effector and base motions are kinematically feasible. We extend the formulation introduced by [18] to enable the reinforcement learning agent to simultaneously avoid collisions with the environment and further expand its control to the velocity of the end-effector motions and the height adjustment of the robot torso. We then generalize the objective and training scheme to learn policies that avoid configuration jumps and generalize to unseen environments while remaining applicable to a diverse set of mobile manipulation platforms. The resulting approach enables it to solve a very broad range of tasks in unstructured real-world environments. An overview of the proposed approach is depicted in Fig. 2.

A. End-effector Motions

Defining tasks such that they can be achieved by algorithms while remaining generally applicable yet simple to specify for humans is a hard problem. Much of the recent work has focused on addressing this challenge through imitation learning or definition of task-specific goal states. In the domain of mobile manipulation, outcomes often depend on specific end-effector operations. As such we define tasks by end-effector motions. This is on one hand very expressive: a wide variety of tasks can be expressed this way. On the other hand, it provides a simple interface for humans to define and generate novel tasks quickly and unambiguously, as it requires only to reason about motions in simple cartesian space, instead of a whole-body control problem. We provide a wide variety of example tasks in Sec. IV. Further, this definition is not overly restrictive in terms of methodology as various types of motion generators for the end-effector can be employed, such as dynamic system based imitation learning, planning based systems, or even a reinforcement learning system.

In particular, we only assume access to an arbitrary end-effector motion generator f_{ee} that, given a current (partial or fully known) map m_{global} of the environment, the current end-effector pose ee_t and velocities $v_{ee,t}$, and the current end-effector goal g, outputs feasible next-step velocities $v_{ee,t+1}$ for the end-effector. We define feasible as any motion that does not violate the constraints of the robot's hardware or the (collision) constraints resulting from obstacles in the world. We can define this as a function

$$v_{ee,t+1} = f_{ee}(ee_t, v_{ee,t}, m_{global}, g). \tag{1}$$

B. Learning Base Control for Kinematic Feasibility

We formulate mobile manipulation tasks as a goal conditional reinforcement learning problem in which the RL agent observes end-effector motion and goal, and aims to ensure that these motions remain kinematically feasible [18]. The agent controlling the robot's base is operating in a Partially Observable Markov Decision Process (POMDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T(s'|s,a), P(o|s), r(s,a))$ where $\mathcal{S}, \mathcal{O}$ and $\mathcal{A}$ are the state, observation and action spaces, T and P describe the transition and observation probabilities, and r and γ are the reward and discount factor. The agent's objective is to learn a policy $\pi(a|o)$ that maximises the expected return $\mathbb{E}_\pi[\sum_{t=1}^T \gamma^t r(s_t, a_t)]$. In the goal conditional setting [50], the agent then learns a policy $\pi(a|o,g)$ that maximises the expected return $r(s,a,g)$ under a goal distribution $g \sim \mathcal{G}$ as $\mathbb{E}_{\pi,\mathcal{G}}[\sum_{t=1}^T \gamma^t r(s_t, a_t, g)]$. At each step, an arbitrary end-effector motion generator produces the next step velocities v_{ee} for the end-effector. The reinforcement learning agent receives an observation o consisting of these velocities v_{ee}, the resulting desired EE pose, and an end-effector goal pose g in the robot's base frame together with the current robot state consisting of the joint configurations and a binary local obstacle map m_{local} centered and oriented around the base and outputs base velocities $v_b \sim \pi(a|o,g)$. In practice we do not let the agent observe the final end-effector goal which can often be far away, but we repeatedly apply the end-effector motion generator f_{ee} to generate an intermediate goal roughly $1.5\,\mathrm{m}$ ahead of the agent.

The objective of the agent is to ensure that these end-effector motions remain feasible. Given the desired end-effector motion and the agent's base commands, we use inverse kinematics (IK) to solve for the remaining degrees of freedom in the arm. To achieve this objective, we extend our previous work [18] in several dimensions. First, we generalize the kinematic feasibility reward as

$$r_{ik} = -||\hat{ee}_{xyz} - ee_{xyz}||^2 - c_{rot} * d_{rot}(\hat{ee}, ee), \tag{2}$$

where ee and $\hat{ee}$ are the achieved and desired end-effector poses with position and orientation components ee_{xyz} and ee_o, d_{rot} calculates a rotational distance of the quaternions $d_{rot} = 1.0 - \langle \hat{ee}_o, ee_o \rangle^2$ and c_{rot} is a constant scaling both components into a similar range. This allows us to explicitly trade-off precision in the achieved pose with adhering to executable arm motions within the IK solver to address the configuration jumps observed in [18]. We use the flexible BioIK solver and regularise the IK solver with a minimum displacement goal, consisting of the squared distances of each joint, weighted by the reciprocals of the maximum joint velocities [57]. While this restricts the arm joints to stay close to the last time step, for robots with flexible arms there may still be several possible configurations for a given end-effector motion. Moreover, the selected arm configuration can be important to fulfill the remaining motions. To increase the RL agent's control over what configuration the arm goes into, we extend its control to the torso lift joint. This joint is often very slow and as a consequence requires substantial look-ahead to change significantly in height. We extend the agent's action space with a learned torso velocity and use the IK solver only for the remaining arm joints.

An episode terminates early if the gripper deviates too much from the desired pose for more than 20 steps. We optimize this as an infinite horizon task, correcting for the (non-markovian) early terminations [58] and taking into account that robots are expected to act in the real world in which they will have to continue reaching new goals after fulfilling the current task. To do this we continue to bootstrap the value functions in the final episode states. Lastly, we replace the regularization of the agent's actions' norm with a regularisation of the acceleration, incentivizing smooth instead of small actions:

$$r_{acc} = ||a_t - a_{t-1}||^2. \tag{3}$$

To ensure the environment remains Markovian, we extend the observation space with the agent's previous actions. In our experiments, we find this regularisation to be essential for the transfer to the real world.

C. Obstacle Avoidance

To perceive the environment, we equip the agent with a local occupancy map of its surroundings. Occupancy maps provide several advantages: (i) They can be constructed from and integrate several sensors, such as LiDAR, RGB-D, and range sensors, ensuring applicability across different robots with different sensors. (ii) They offer a geometric abstraction over sensors such as RGB images, that we hypothesize is easier to generalize to unseen objects and environments at test time. (iii) They can be constructed in either 2D or as 3D voxel maps. In the following, we rely on a 2D representation which as we will show in the experiments is sufficient for a large range of real-world tasks while keeping computational costs low and during training can be generated without having to rely on a complex simulator.

The agent receives a robot-centered map m_{local} with a side length of three meters. This map is processed by a map encoder before being concatenated with the robot's proprioceptive state and the goal observation. The map encoder receives two versions of the local map: a coarse one at a resolution of $0.1\,\mathrm{m}$ and a fine-grained version with a side length of $0.75\,\mathrm{m}$ at a resolution of $0.025\,\mathrm{m}$, enabling precise navigation. The concatenation of the two maps is passed through three convolutional layers each followed by a maxpool and ReLU activation and then flattened into a vector. We then extend the reward with a collisions penalty of $r_{coll} = -10$.

D. Controlling Motion Velocities

While we can express discretized motions as a time-stamped sequence of poses in SE3, many real-world tasks do not depend on a particular execution speed. Furthermore, in cluttered environments, the robot base has to frequently evade obstacles, potentially requiring significantly longer paths than the end-effector. As a consequence, we propose to also learn to control the norm of the velocity of the end-effector motions. We extend the agent's action space by an additional action a_{ee} that controls the norm of the end-effector velocity after observing the desired next motion by scaling it to $||v_{ee}|| = a_{ee}$. To incentivize fast motions whenever possible, we add the following reward:

$$r_{vel} = -(v_{ee,max} - a_{ee})^2, \tag{4}$$

where $v_{ee,max}$ is the maximum velocity the end-effector is allowed, which we set equal to the maximum base velocity.

To prevent the agent from finding fast movement through difficult poses as a valid strategy to minimize collisions or ik failures, we furthermore transform the penalties for collisions and ik failures from a reward per time step into a reward per distance by multiplying them with the normalization term

$$n_{vel} = \frac{a_{ee}}{v_{ee,max}}. \tag{5}$$

E. Overall Objective

Combining the objectives, the overall reward function is

$$r = n_{vel}(\lambda_{ik} r_{ik} + r_{coll}) + \lambda_{vel} r_{vel} + \lambda_{acc} r_{acc}. \tag{6}$$

We optimize this objective with model-free reinforcement learning, namely soft-actor critic (SAC) which has shown to effectively learn robust policies for continuous control tasks [59].

F. Training Environment

To generalize to unseen motions and obstacles, the agent has to be exposed to a wide variety of motions, poses, and obstacles. We introduce a procedurally generated environment that uses elementary shapes to generate obstacles and a simple yet effective obstacle-avoiding end-effector motion generator. As we will demonstrate in our experiments, the resulting policy generalizes to unseen geometries and motions at test time.

Task: In each episode, we randomly arrange elementary shapes, namely rectangles and ellipses, on an empty map. Each obstacle is placed on a regular grid, offset by a random number drawn from a normal distribution, and placed in a random orientation. The obstacle's width, breadth, and height are drawn from a uniform distribution. An example of such a map is depicted in Fig. 3. Given the obstacle map, we set a goal reaching task by sampling a random start pose, random initial joint values, and a random goal position within a distance of $0.5\,\mathrm{m}$ to $5\,\mathrm{m}$ from the start.

While procedurally generated environments have proven very powerful to learn robust policies, they come with their challenges. In particular, we have to ensure that the generated task is solvable. For this, we follow a simple heuristic and reject any goal for which no valid path can be found in a map with an inflation radius of $0.4\,\mathrm{m}$. While more elaborate methods such as curricula [60] and adversarial environment generation [61] have shown promising results, we found it not just simpler but also more effective to directly randomize and maximize the diversity of the environment and goals. During training, we simply integrate the robot kinematics over time, allowing us to generate data very quickly and without relying on a complex simulator. We will refer to this as *analytical environment* throughout the remainder of the paper.

End-Effector Motions: For training, we propose an instance of a simple obstacle-aware motion generator that remains agnostic to the robot. In Sec. IV, we evaluate how well the behavior learned on these motions generalizes to arbitrary, unseen motions. To ensure that the desired motions are in general feasible, we use two heuristics and develop a refined A*-based planner to implement them as soft constraints:

(i) Avoid end-effector collisions: A minimum distance d_{ee} of the end-effector from tall obstacles. This can be implemented by inflating the obstacle map, as is common practice in robot navigation. We ignore obstacles that are smaller than the maximum height the end-effector can reach (minus a small margin) max_z, meaning the end-effector motion is allowed to pass over obstacles where kinematically possible.

(ii) Ensure the base can follow the end-effector motions: we first plan a path from the initial base pose to the end-effector goal in a map that is inflated by the radius of the robot base. We then define a maximum distance d_{base} that the end-effector is allowed to deviate from this base path. We set this value to the range of the robot's arm. We implement this by inflating the base path by d_{base} and adding the inverse of this to the weights.

Together, this results in the following weights w of the A*-planner, consisting of the inflated end-effector path and the inverse of the inflated base path:

$$\begin{aligned} w = c * \mathbb{1}[&inflate(m_{global,z+}, d_{ee}) \\ &+ (1 - inflate(m_{basepath}, d_{base}))], \end{aligned} \tag{7}$$

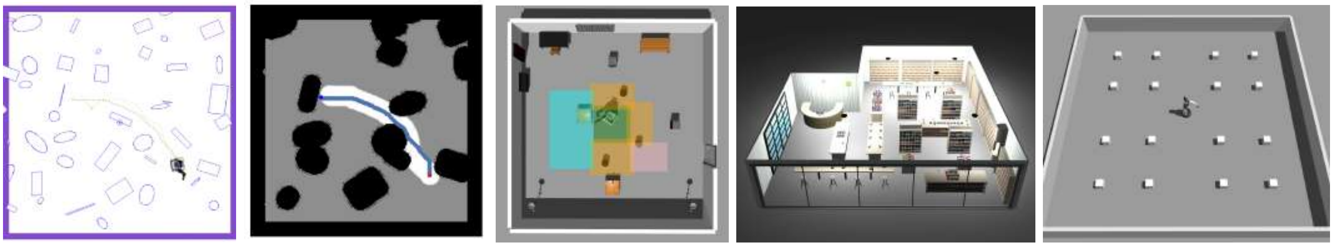


Fig. 3. Depiction of the training and testing environments. Left: procedurally generated training task. Middle left: generated weights and path of the end-effector planner according to Eq. (7). The darker the image, the higher the cost. Center: articulated object environment. The starting the area of the robot is shown in green, the spawn area of the obstacles is shown in orange (except for door and spline tasks), the starting area for the door task is shown in pink. For the p&p task, the robot has to grasp a cereal box on the bottom table and place it onto a table randomly spawned in the cyan area. Middle right: bookstore map. Right: dynamic obstacle task.

where c is a constant, $m_{global,z+}$ is a map with only the obstacles larger than max_z over which the end-effector cannot pass, and $m_{basepath}$ is a map consisting only of the shortest path from the initial base pose to end-effector goal obtained by A*. An example of the resulting weights is shown in Fig. 3.

We then generate dense waypoints with A* and interpolate them with a linear dynamic system to produce the actual end-effector motions. We linearly interpolate the height over the found path from the current height towards $\max(next\_obstacle\_z + height\_margin, goal\_z)$. Rotations are obtained via spherical linear interpolation (slerp) from start to goal orientation. As we demonstrate in the experiments, this results in an effective and yet simple method capable of generating motions for complex, real-world floor plans. Note that while more sophisticated methods could generate easier to achieve motions in crowded maps, our main interest here is to generate diverse motions for the RL agent. The task is deemed successful if the agent gets within $2.5\,\mathrm{cm}$ and a rotational distance of 0.05 of the end-effector goal, has no base collisions or self-collisions, and does not deviate more than $10\,\mathrm{cm}$ or a rotational distance of 0.05 from the desired motion. We early terminate the episode if the agent violates any of these constraints for more than 20 steps.

IV. Experimental Evaluations

We evaluate our approach across different robotic platforms, different environments, and a wide variety of tasks. In these experiments, we aim to answer the following questions:

- Do agents trained with our proposed approach generalize to unseen end-effector motions?
- Does our approach generalize across robotic platforms with different kinematic abilities?
- Does the agent learn robust obstacle avoidance that generalizes to (i) occupancy maps generated by real-world LiDAR scans and (ii) unseen obstacles?
- Does our approach scale to complex, real-world based obstacle environments?
- Can the agent react to dynamic obstacles and changing environments?
- Does our approach transfer to direct execution in the real world?

TABLE I
Velocity and pose constraints.

Parameter	EE-Motion	PR2	TIAGo	HSR
Max. velocity (m/s)	0.2	0.2	0.2	0.2
Max. rotation (rad/s)	0.3	1.0	1.0	1.0
Goal height (m)	-	[0.2, 1.55]	[0.2, 1.5]	[0.2, 1.4]
Restr. height (m)	-	[0.4, 1.0]	[0.4, 1.1]	[0.4, 1.1]

A. Experimental Robotic Platforms

We train agents for three different robotic platforms differing considerably in their kinematic structure and base motion abilities. The *PR2* robot consists of an omnidirectional base, a height-adjustable torso, and a 7-DOF arm, giving it high mobility and kinematic flexibility. But with a base diagonal of $0.91\,\mathrm{m}$ it is comparably large, limiting its maneuverability in narrow spaces. The *Toyota HSR* robot has an omnidirectional base as well, but the arm is limited to 5-DOF including the height-adjustable torso. As a result, it cannot reach all poses in SE3 unless it coordinates movements with its base. The *TIAGo* robot is equipped with a height-adjustable torso similar to the PR2 as well as a flexible 7-DOF arm. But it uses a differential drive, restricting its mobility compared to the PR2. All three robots are equipped with a base LiDAR sensor, covering 270, 240, and 220 degrees on the PR2, HSR, and TIAGo, respectively. The TIAGo additionally has three range sensors pointing backward, each with a field of view of $29\,\mathrm{deg}$ and a range of one meter. In the simulation, we also equip the PR2 and HSR with the same range sensors. The action space for these platforms is continuous, consisting of either one (diff-drive) or two (omni) directional velocities $\mathbf{v}_{\mathbf{b},\{\mathbf{x},\mathbf{y}\}}$, and an angular velocity $\mathbf{v}_{\mathbf{b},\theta}$. This action space is completed with two more actions, one controlling the velocity of the end-effector motion and the other controlling the torso lift joint. Except for the HSR for which we do not find a benefit in learning the torso over solving for it with the IK as it only has limited DoF. Tab. I lists the constraints we set across the different platforms in the analytical environment.

B. Baselines

We compare our proposed approach against a range of methods from planning, optimal control, and machine learning.

MPC: As the main baseline, we compare against a recent model-predictive control approach [30]. The method uses the SLQ algorithm to minimize the deviation to the desired end-effector trajectory with additional soft constraints for self-collision, joint velocity, and joint position limits. To ensure a fair comparison, we change all robot collision meshes to simplistic geometries, reduce the number of self-collision constraints to a bare minimum (avoiding end-effector – torso and end-effector – head collisions), and set all unused joints as fixed. Collision avoidance is incorporated as a hinge-loss function based on a signed distance field. We provide it with a groundtruth 2D signed distance field at a resolution of 2.5 cm from the obstacle map and model the base collisions as a cylinder centered on the robot base. For the rectangular base of the PR2, we define four more small cylinders at the corners of the base and implement the derivatives with regard to position and orientation. We follow the authors and run the algorithm at a control frequency of 30 Hz with a time horizon of four seconds. The agent has to follow the same end-effector motions as our approach. As the original code is not publicly available, we reimplement it based on the same optimization library[1].

E2E: An end-to-end reinforcement learning agent which does not rely on an IK solver but directly learns to control the whole robot by extending the action space to include the joint velocities for the arm. It receives the same desired end-effector motions and reward signal but directly acts in the full configuration space of the robot to achieve these motions.

LKF: The original learning kinematic feasibility approach [18] uses a binary indicator as a reward and does not take into account obstacles. In a range of ablation studies, we extend it until we reach our approach.

RRTConnect: For comparison with classical planning approaches, we evaluate RRTConnect [10]. However, as general motion constraints over time cannot easily be defined in the sampling space, we omit these constraints. The planning algorithm only receives the same start and goal poses as the learning agent and is allowed to freely move to the target pose. As such it has much more freedom to achieve the goal, but at the same time is limited in applicability to goal reaching and pick&place tasks. We use the planner implementations provided by the MoveIt motion planning library [62]. While samplers can in principle support arbitrary constraints such as non-holonomic bases, current ROS1 implementations do not support it. As Tiago is not yet ROS2 compatible, we do not evaluate it on this platform.

Bi2RRT*: A bidirectional RRT* algorithm developed for mobile manipulation with the PR2 which achieves efficient sampling by using a two-tree variant of Informed RRT* [11]. As for RRTConnect, we omit the motion constraints. We use the authors' implementation.

C. Evaluation Setup

As the main focus of this work are the base motions, we provide the EE-planner with a groundtruth map of the environment for all tasks except those involving dynamic environments. However, the RL agent is always restricted to the local map (groundtruth in the analytical environment, based on its sensors in all other environments). The planner baselines receive access to the full 3D groundtruth planning scene. Only for the planner baselines, we additionally simplify all complex collision meshes in the bookstore map with simple geometries to reduce the impact of collision checking on the planning times. We tune the MPC and planner baselines for each robot via grid search on the rnd obstacle task, then evaluate the best set of parameters on all tasks. For our approach we use a single set of parameters across all robots without any further platform-specific tuning. Hyperparameters for all methods are reported in Tab. II. The approaches were evaluated on an AMD Ryzen 9 5900X or AMD EPYC 7452 CPU. As the PR2 Gazebo physics no longer matches the real robot[2], we update it to roughly fit the actual hardware by setting the wheel, and rotation joint efforts to 25 and the wheel controller's proportional gains to 200.

TABLE II
Hyperparameters for the different approaches.

RL			
tau	1e−3	buffer size	10e5
lr	1e−4	lr end	1e−6
gamma	0.99	steps	10e6
ent coef	learned	train intensity	0.1
batch size	256	hidden layers	[512, 512, 256]
action noise	0.04	frame skip	8 (Tiago)
λ_{ik}	50	c_{rot}	2
λ_{vel}	0.1	λ_{acc}	0.05
MPC			
time horizon	{4 sec, 6 sec}	frequency	30 Hz
μ collision	{500, 1000, 5000}	δ collision	{0.001, 0.01, 0.1}
μ joint limit	{0.01, 0.1}	δ joint limit	{0.001, 0.01}
min step	{0.01, 0.03}		
RRTConnect			
max waypoint	{0.1, 0.2, 0.4}	range	{default, 1, 10}
plan attempts	{1, 10}	time limit	200 sec
Bi2RRT*			
extend step	{0.1, 0.2, 0.4}	time limit	200 sec

Notes: For the planner and MPC baselines we perform a grid search on the obstacle task over these values and then report the results of the best parameters on all tasks.

Metrics: A task is deemed successful if the end-effector reaches the goal without any collisions and never deviates more than 10 cm or a rotational distance d_{rot} of 0.05 from the desired motion. In the analytical environment, we also count configuration jumps as a failure, where we define a jump as exceeding any of the robot's maximum joint velocities. In the analytical environment, we only track base collisions, in the Gazebo simulator we check all collisions with the robot body. We evaluate each task over 50 episodes. For learning-based methods, we additionally average over five models trained with different random seeds.

[1] https://github.com/leggedrobotics/ocs2

[2] https://github.com/PR2/pr2_controllers/issues/402

TABLE III
SUCCESS RATES FOR EXPERIMENTS IN THE ANALYTICAL ENVIRONMENT.

		A^*-slerp					Imitation Learning			Spline	Avg
	Agent	rnd obstacle	p&p	bookstore p&p	dynamic obstacle	dynamic p&p	cabinet	drawer	door	spline	
PR2	MPC	54.0	70.0	8.0	64.0	54.0	62.0	76.0	32.0	96.0	57.3
	E2E	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	$\mathbf{N^2M^2}$	86.0	97.6	70.0	84.4	81.6	97.2	97.6	98.4	80.8	**88.2**
HSR	MPC	64.0	72.0	48.0	60.0	60.0	68.0	68.0	66.0	98.0	67.1
	E2E	25.2	38.8	10.4	17.6	13.2	52.4	14.4	43.2	38.0	28.1
	$\mathbf{N^2M^2}$	63.2	92.0	68.0	82.0	88.0	93.2	87.2	88.0	84.0	**82.4**
Tiago	MPC	46.0	68.0	0.8	40.0	30.0	62.0	52.0	32.0	86.0	46.3
	E2E	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	$\mathbf{N^2M^2}$	61.2	91.6	42.0	54.4	50.4	81.6	89.2	62.4	56.0	**65.4**

Notes: Evaluation on unseen tasks from three different motions systems, an A^*-based system, an imitation system learned from human demonstrations and spline interpolation of random waypoints. The last column reports the average across all tasks. We evaluate all models on three different robotic platforms, the PR2, HSR, and TIAGo.

D. Generalisation to Arbitrary Motions

To evaluate the generalization to arbitrary motions, we generate a range of tasks from different motion systems:

A^*-slerp refers to the A*-based motions used during training, as described in Sec. III-F. *A^*-fwd* uses the same end-effector path but generates different desired end-effector orientations: Instead of spherical interpolation, it generates motions in which the end-effector always points in the direction of movement. Only when close to the goal, it uses slerp to achieve the desired final end-effector orientation. From these motions, we construct the training task, labeled *rnd obstacle*, and a pick&place (*p&p*) task in which the agent has to grasp a cereal box from one table and place it down on another table.

An *imitation learning system* learned from a human teacher [63]. The motions are encoded in a dynamic system following a demonstrated hand trajectory to manipulate a certain object. In particular, we use motions to grasp and open a cabinet, grasp and open a drawer and grasp a door handle, push it down and open the door inwards while driving through it. This task is particularly challenging as it requires to avoid both the door frame and the moving door, resulting in a highly constrained and narrow navigation task. With $0.89\,\mathrm{m}$, the door frame is more narrow than the PR2 base-diagonal of $0.91\,\mathrm{m}$. The moving door makes the environment dynamic as it changes the free space.

Spline interpolation draws five random waypoints in SE(3) with a distance of $1\,\mathrm{m}$ to $3\,\mathrm{m}$ from the previous and connects them using cubic splines and spherical interpolation. This results in arbitrary motions over a total distance of $5\,\mathrm{m}$ to $15\,\mathrm{m}$.

With these motions, we extend the suite of mobile manipulation tasks introduced by [18], while also removing all restrictions on the initial start pose. The task objects are situated in a virtual room and the robot spawns in a random pose and configuration within an area in the center of the room. To evaluate the obstacle avoidance capabilities, we place random obstacles in the path of the agent. We allow the end-effector motions to directly pass over these obstacles, leading to challenging poses and decisions on how to position the base. The task setup and start area are shown in Fig. 3.

We first evaluate the agents in the analytical environment, abstracting from low-level execution and sensors. The results are shown in Tab. III. MPC solves a significant amount of episodes across all the robots. In particular, it performs very well on the obstacle-free spline task. But it frequently gets stuck around obstacles, only making progress once time progresses and the desired end-effector pose moves far enough away for the tracking cost term to dominate. This results in a violation of one of the constraints: either large deviations from the desired pose, base collisions, or joint limits as it is unable to trade them off optimally in these situations. Due to the difficulties in balancing these soft constraints, in the door opening task, the MPC controller frequently drives over the door frame with the PR2. The E2E model is able to solve a reasonable number of tasks on the HSR. But as the action space increases with the 7-DoF arms of the PR2 and TIAGo, it is no longer able to succeed in any of the tasks. This matches the fact that recent reinforcement learning based mobile manipulation approaches have to restrict the action space to learn. In contrast, our hybrid approach scales effectively with the configuration space of the robot and is able to use the full flexibility of the PR2, achieving the best results on this platform with far above 90% success on many tasks. We achieve similar success rates on the HSR, demonstrating the agent's ability to use the base to complement its limited arm. The most difficult platform is the TIAGo. While our approach does very well on p&p and the imitation learning tasks, performance is significantly lower on the rnd obstacle and spline tasks. We hypothesize that the differential drive induces a much harder exploration problem. In particular, it is much harder to move from one mode of operation to another: e.g. in front of an obstacle, the base has to decide to either turn left or right. But once it has made a decision, it cannot easily change the path without violating the end-effector motion. As such it becomes much harder to escape local optima in the behaviour policy. To better learn long-horizon policies for this robot, we start training with a lower base control frequency of $0.0125\,\mathrm{Hz}$ and then linearly increase it to $10\,\mathrm{Hz}$ over the course of training. Across all tasks and robots, our approach significantly outperforms the other methods. Also note that, given the procedural generation of the environment, it may not

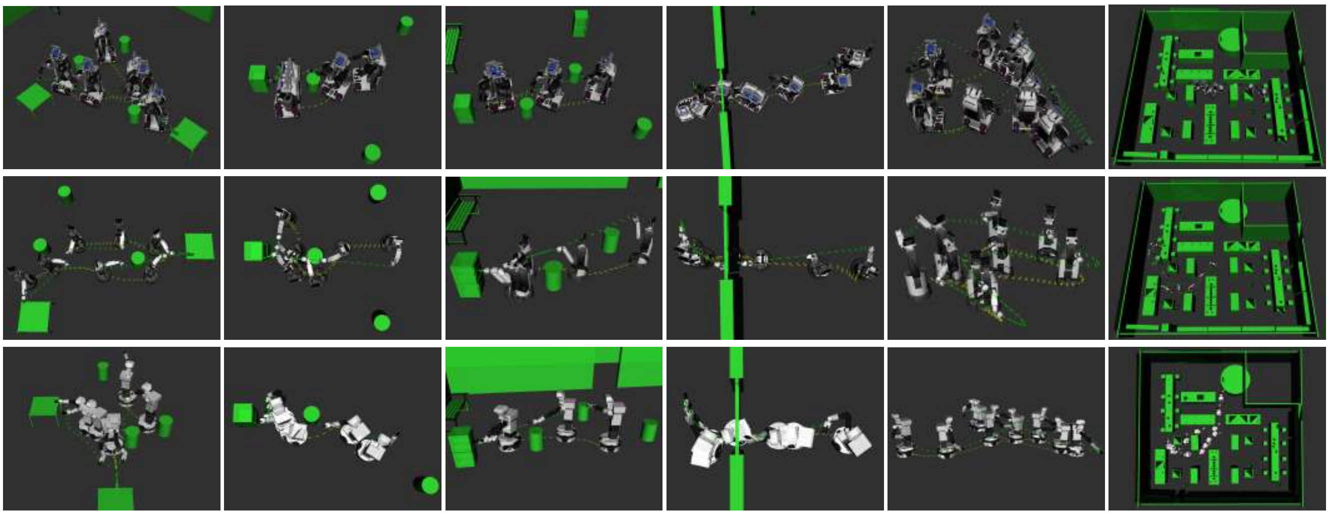


Fig. 4. Example motions produced by N^2M^2on the different tasks and robots. The base and ee-trajectories are marked in yellow and green, respectively. From left to right: p&p, cabinet, drawer, door, spline and bookstore. From top to bottom: PR2, HSR, TIAGo.

be possible to achieve a perfect score on the rnd obstacle task, as some end-effector motions may not be possible to fulfill kinematically for some of the robots.

Qualitatively, we find that the agent produces reasonable behavior such as seeking robust poses in the center of its workspace or moving backward until reaching an open space to turn. Examples are shown in Fig. 4 and in the accompanying video.

E. Ablation: Individual Components

To evaluate the impact of our contributions, we perform extensive comparisons with the original LKF approach. As it cannot avoid obstacles, we train these approaches on a random goal reaching task in an empty map and compare the ability to follow the motions on a subset of the previous tasks *without obstacles*. Note that results are not directly comparable to those reported in [18], as our task definitions are generally more challenging as they remove restrictions on initial poses and include the torso joint of the PR2. We report both the same success rates as before as well as a success rate that does not penalize configuration jumps as used in [18], which we label *w/jumps*.

The results are shown in Tab. IV. Row by row we add our contributions. We first switch to the BioIk solver with minimal displacement regularisation and the generalized L2 kinematic feasibility reward function. This largely increases the success rate by removing most configuration jumps. For the PR2 and TIAGo, we then additionally learn the torso lift velocities, which have a similarly large impact on the success rate. This is because most jumps occurred in the torso lift joints which move much slower than the arm joints. Furthermore, for many goals, it is important to plan ahead and to start moving the torso early on in the episode to achieve large height differences. Next, we also learn to control the velocity of the end-effector motions. This slightly increases success rates, even in the absence of obstacles. We found the impact of this term even more important in the presence of obstacles as the base has to maneuver much more. Lastly, we add the occupancy map and train in the new procedurally generated training task. With this we arrive at our approach, which achieves the best average performance on all robots, highlighting the importance of all components.We hypothesize that this last increase stems from the fact that the new obstacle task largely increases the difficulty and variety of the motions during training.

F. Scaling to Human-Centered Maps

To further evaluate the generalization to complex objects with unseen geometries and the ability to navigate narrow human-centered environments, we evaluate the agents in a realistic bookstore environment [64] shown in Fig. 3. We define a set of ten feasible object locations across the whole map and draw random pairs of locations to pick up an object from and place it down. The robot starts in the center of the map, again in a random start configuration. The resulting tasks are very long horizon, taking on average 1,300-1,500 steps at a control frequency of $10\,\mathrm{Hz}$ and require passing through passages as narrow as $0.85\,\mathrm{m}$. The results are reported in Tab. III. Our approach is the only method to successfully complete this task on all robots. While success rates are lower than in p&p, both the PR2 and HSR succeed in the majority of episodes. Success rates are somewhat lower on the TIAGo with 42.0%. Difficulties of this task are on one hand the sheer length of each task. On the other hand, the robot has to repeatedly pass through very narrow passages. This is particularly challenging as the agent at the same time has to strictly adhere to the end-effector motions. As our focus is the learned base behavior, we do not adapt the end-effector orientations to the environment, even though the pick & place task does not require specific motions while carrying the object. This highlights a large potential to further improve the success rates by learning or adapting the end-effector motions to the specific task. We leave this for future work.

TABLE IV
SUCCESS RATES FOR EXPERIMENTS IN THE ANALYTICAL ENVIRONMENT *without* OBSTACLES.

	Agent	Linear		A^*-slerp		Imitation				Spline		Avg
		rnd goal		p&p		cabinet		drawer		spline		
		success	w/jumps	success	w/jumps	success	w/jumps	success	w/jumps	success	w/jumps	success
PR2	LKF	0	80	0	46	0	86	0	64	0	42	0.0
	+ bioik, l2, jumps	38	94	8	22	36	82	22	68	0	92	20.8
	+ learn-torso	**92**	92	50	50	72	74	74	74	82	82	74.0
	+ ee-velocity	90	90	72	72	82	82	98	98	66	66	81.6
	+ collision: $\mathbf{N^2M^2}$	88	88	**100**	100	98	98	**100**	100	**72**	72	**91.6**
HSR	LKF	**72**	74	90	92	76	84	80	84	64	80	76.4
	+ bioik, l2, jumps	66	72	94	96	82	92	78	84	82	92	80.4
	+ ee-velocity	**72**	74	90	92	90	92	86	92	80	94	83.6
	+ collision: $\mathbf{N^2M^2}$	**72**	72	**98**	98	**98**	98	**92**	96	**86**	96	**89.2**
Tiago	LKF	14	50	4	76	4	80	0	72	0	4	4.4
	+ bioik, l2, jumps	48	76	42	74	36	92	44	92	20	70	38.0
	+ learn-torso	64	64	**96**	96	80	80	90	90	48	48	75.6
	+ ee-velocity	66	66	86	86	**94**	94	**92**	92	50	50	77.6
	+ collision: $\mathbf{N^2M^2}$	**76**	76	90	90	88	88	86	86	**56**	56	**79.2**

Notes: LKF denotes the previous Learning Kinematic Feasibility approach [18]. Each row adds parts of our contributions to the row above it until we arrive at our proposed model. *w/jumps* denotes success rates ignoring configuration jumps. Evaluated over a single seed.

G. Dynamic Obstacles

To evaluate the reactiveness of the trained agents, we construct two tasks with dynamic obstacles. The obstacles move with a random velocity between $0.1\,\mathrm{m\,s^{-1}}$ and $0.15\,\mathrm{m\,s^{-1}}$. We re-plan the end-effector motion at every time step in MPC-fashion, which easily runs in real-time as all the complexities are shifted to the RL agent. This is again a zero-shot transfer: the agent has never seen moving obstacles during training. In the *dynamic obstacles* task, we uniformly spawn 16 dynamic obstacles in an empty room and draw a random goal for the end-effector, as shown in Fig. 3.

We exclude goals on the very edges of the robot's workspace where its maneuverability is very low, see ”Restr. height” in Tab. I. The *dynamic p&p* task uses the same setup as p&p, but we replace the static obstacles with three dynamic obstacles. The results are included in Tab. III. We find that our agent is able to very quickly react to these obstacles. Success rates for the PR2 and HSR are close to those of static obstacle tasks. The TIAGo has more difficulties: the differential drive does not allow it to easily move away from obstacles that move into the side of its base. As it has never seen dynamic obstacles, it has no incentive to anticipate these situations and to proactively place its base accordingly. Additional fine-tuning on dynamic obstacles might be able to induce it with such a sense of foresight. While the MPC approach is able to react to the dynamic obstacles, its success rate drops further down to 30-64% versus the 50.4-88% of our approach.

H. Comparison to Planners

Planning-based methods are very general and widely used for manipulation tasks. While the specification of arbitrary motion constraints such as defined by the imitation learning motions is difficult as discussed in Sec. II, these approaches are a powerful baseline for a subset of our tasks: goal-reaching and pick&place in static environments. We compare with two sampling-based planners on the rnd obstacle, pick&place, and bookstore tasks. We remove the end-effector motion constraints and instead solely specify the task with three goal poses: in front of the object, the grasp pose, and the location to place it down. Note that this is a significantly simpler task as the end-effector is completely unconstrained in -etween the goals. In the second evaluation, we incorporate an additional simple pose constraint to hold the object upright after grasping it that we label as *(upright)*.

Tab. V compares the success rates, planning time, and the resulting length of the end-effector trajectory relative to our approach, which still has to fulfill all motion constraints. We find that they perform well in the obstacle and pick&place tasks, even outperforming our approach on the HSR. At the same time, success rates decrease on the PR2. This can be attributed to the larger joint space and the larger base, making the spaces narrower. In contrast, our approach performs better on the PR2 and is able to make use of its flexibility, indicating good scaling with robot complexity. Already adding the simple upright constraint decreases performance on p&p by 6ppt for the PR2. The more restricted HSR completely fails to sample successful paths under this constraint. Moving to the complex bookstore map, performance drops significantly below our approach with success rates of 22% to 42% versus 70%. While planning times are quite low on the other tasks, they now increase to almost 2 minutes for the PR2. Qualitatively, the planners can produce somewhat unnatural paths, such as waving the whole arm when going from the goal directly in front of the pick object to the pick object. In contrast, our approach seeks out base poses that enable good reactiveness and that are robust to noise in the base movements.

These results highlight two aspects: on one hand the difficulty of the bookstore map, on the other hand the general applicability of our approach and its beneficial scaling with the size of the configuration space: while having to follow much more restrictive motion constraints, it remains competitive with unrestricted state-of-the-art methods and even outperforms them as both robot and map complexity increase. At the same

TABLE V
EVALUATION OF THE PLANNER BASELINES.

		rnd obstacle			p&p			bookstore p&p		
	Agent	success	planning time	rel ee path	success	planning time	rel ee path	success	planning time	rel ee path
PR2	RRTConnect	74.0	3.1 sec	1.25	78.0	0.9 sec	1.04	22.0	119.6 sec	0.97
	RRTConnect (upright)	–	–	–	72.0	3.3 sec	0.98	22.0	124.6 sec	0.97
	Bi2RRT*	62.5	41.5 sec	1.38	84.0	8.6 sec	1.17	8.0	115.3 sec	1.12
	N^{2}M^2	**86.0**	0.1 sec	1.00	**97.6**	0.1 sec	1.00	**70.0**	0.1 sec	1.00
HSR	RRTConnect	**82.0**	0.5 sec	0.83	**100.0**	1.0 sec	0.99	42.0	12.2 sec	0.89
	RRTConnect (upright)	–	–	–	0.0	n.d.	n.d.	0.0	n.d.	n.d.
	N^{2}M^2	63.2	0.1 sec	1.00	92.0	0.1 sec	1.00	**68.0**	0.1 sec	1.00

Notes: Motions for the planners are unconstrained while N^{2}M^{2}has to follow the full end-effector motion. *(upright)* adds a constraint to keep the object upright after picking it up. rel ee path denotes the length of the ee-path relative to N^{2}M^2. Planning time and rel ee path are only computed over successful episodes.

time, it is directly applicable to dynamic scenes, partially observable environments, and arbitrary end-effector motions as demonstrated in Sec. IV-D.

I. Executability

To evaluate if the produced motions are readily executable, we transfer the learned policies to the Gazebo physics simulator. The agent's actions are sent to the robot's low-level base and arm controllers at an average frequency of around 30 Hz to 70 Hz. We construct an occupancy map by integrating the LiDAR and range sensor data into a binary costmap. For the dynamic obstacle tasks, the end-effector motions are generated based on an additional global costmap built from the robot's sensor data. The MPC baseline relies on velocity controllers for the whole robot. We use default velocity controllers of the PR2 and a pseudo velocity controller of the HSR, which underneath utilizes the position controllers. TIAGo does not provide any velocity controllers for the arm, as such we were unable to evaluate the MPC approach on this robot. While our agent can only rely on its sensors, we still provide the MPC approach with the ground-truth signed distance field to reduce complexity. For the dynamic obstacle tasks and the bookstore map, we inflate the local map passed to the RL agent by 3 cm to allow the agent to more quickly react to the obstacles. For the narrow door opening task, we scale the actions of the PR2 by a factor of 0.5.

The results for all tasks are shown in Tab. VI. In contrast to the analytical environment, the agents have to generalize to unseen physics and low-level controllers, asynchronous execution as well as noisy and partial occupancy maps. Furthermore, any arm collision is now recorded as a failure. Nonetheless, the performance of our approach closely matches the analytical environment with small drops of 9ppt, 0.3ppt, and 2.6ppt in average performance. Looking more closely, the differences stem almost exclusively from the door opening (PR2, TIAGo) and the bookstore (PR2) tasks. In the door opening task we find that the imprecisions induced by the low-level controllers and asynchronous execution cause the PR2 to slightly touch the door frame in a number of episodes. As this is a very high-precision task for its large base, the unseen physics and accelerations can quickly have an impact. For most episodes, success depends on only a few centimeters. Failures on the TIAGo stem largely from collisions between the elbow and the door frame. The robot commonly approaches the door with its arm in one of two configurations. In one of these configurations, the elbow is unable to avoid the door frame, leading to several failures. As the agent does not take into account arm collisions during training, the learned policy has no means yet to avoid this failure mode. We leave full 3D-collision avoidance to future work. Similar to the door task, the drop in performance in the bookstore map for the PR2 can be attributed to a larger number of collisions in narrow pathways due to the differences in physics and accelerations compared to the training environment as well as a small number of arm collisions with bookshelves. Overall, performance remains high across the large majority of tasks. This is in contrast with the MPC approach which is significantly impacted by these factors. The approach struggles with unseen physics and imperfect execution of motions, leading to collisions and large deviations from the desired motions on both robots. Once the error terms become too large, the approach is unable to recover good behavior, resulting in large and abrupt constraint violations.

J. Real-World Experiments

We transfer the agents to the real world and evaluate them on the HSR and PR2 robots. For the PR2, we construct an environment consisting of two rooms connected by a door in the hall it is located in. We directly evaluate the HSR in our office building, a common human-centered environment not adapted to the robot's capabilities. Maps of both environments are shown in Fig. 5. Due to their low success rates in the analytical environment and the gazebo environment, both the MPC and E2E baselines pose an increased risk of damaging the robot if executed in the real world. We therefore refrained from running them in our real world experiments.

We mark goal poses for the end-effector with AR-markers and use the robots' head cameras to detect them. If the marker is initially out of sight, the robot receives an initial guess of the target pose. To easily specify goals, we use a pre-recorded map and Adaptive Monte Carlo Localization (AMCL) for localization. We also give the end-effector planners access to this map, in the form of a static layer in the sensed global costmap. For each episode, we move the robot into a random start pose and draw random initial joint values. We scale the

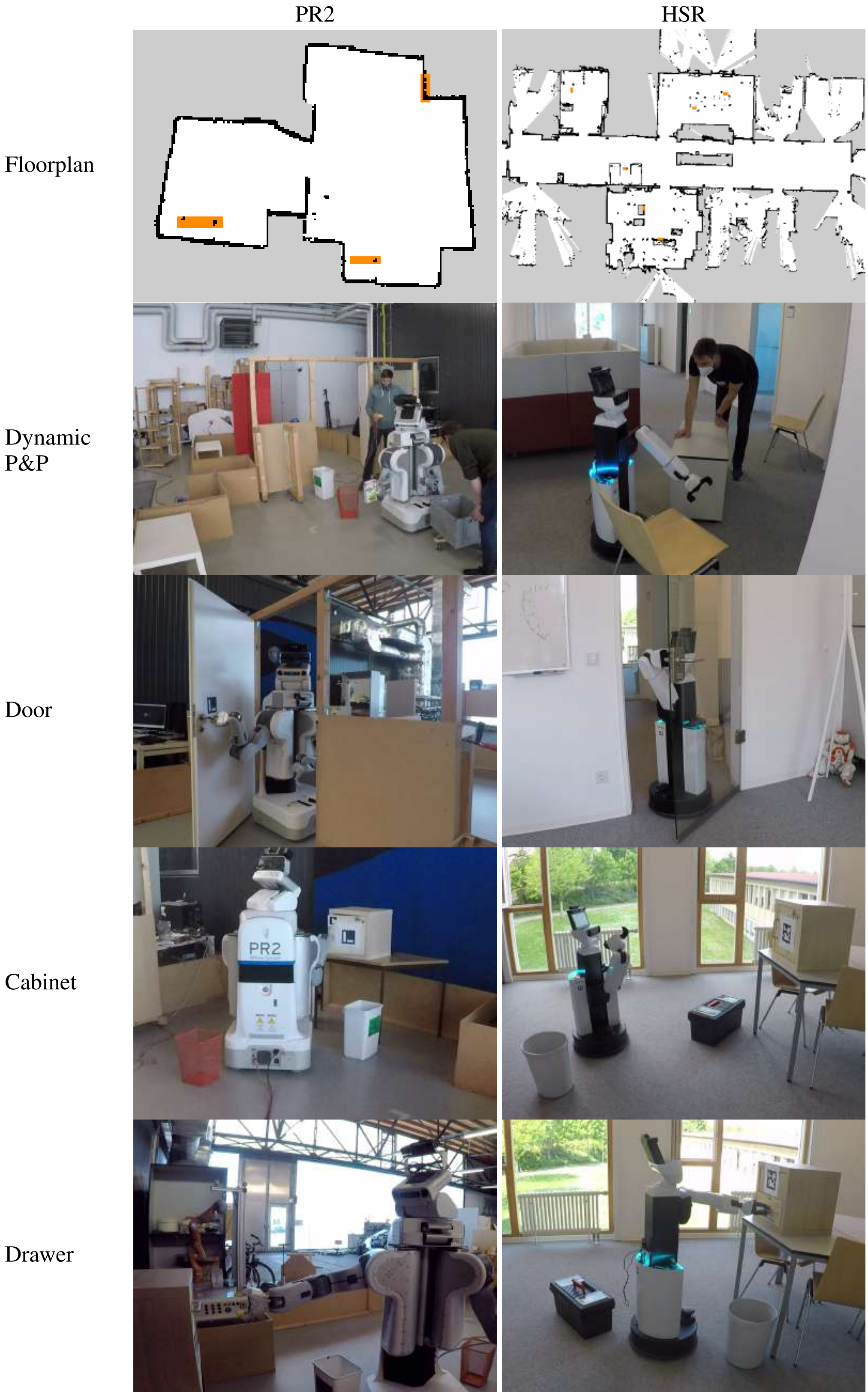


Fig. 5. Real world experiments on the PR2 (left) and HSR (right) robots. From top to bottom: environment map, pick&place while avoiding static and dynamic obstacles (table locations marked in orange), opening and driving through a door, opening a cabinet and opening a drawer with static obstacles constraining the base.

TABLE VI
SUCCESS RATES FOR EXPERIMENTS IN THE GAZEBO ENVIRONMENT.

		A^*-slerp					Imitation Learning			Spline	Avg
	Agent	rnd obstacle	p&p	bookstore p&p	dynamic obstacle	dynamic p&p	cabinet	drawer	door	spline	
PR2	MPC	26.0	12.0	0.0	38.0	2.0	0.0	2.0	0.0	18.0	10.9
	E2E	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	N^{2}M^2	81.6	87.6	48.8	93.2	74.4	94.0	96.8	60.4	75.6	**79.2**
HSR	MPC	12.0	14.0	0.0	30.0	2.0	0.0	4.0	0.0	42.0	11.6
	E2E	29.6	27.2	19.6	27.2	2.8	39.2	8.4	1.2	7.2	18.0
	N^{2}M^2	64.0	92.4	54.4	91.2	78.8	95.6	90.4	85.6	91.6	**82.7**
Tiago	MPC	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
	E2E	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	N^{2}M^2	56.0	85.2	54.8	71.2	51.2	78.8	82.4	30.0	56.0	**62.8**

Notes: Evaluation of zero-shot transfer to the Gazebo physics simulator on unseen tasks from three different motions systems, an A^*-based system, an imitation system learned from human demonstrations and spline interpolation of random waypoints. The last column reports the average across all tasks. We evaluate all models on three different robotic platforms, the PR2, HSR, and TIAGo. MPC is not evaluated on the TIAGo as it does not posses velocity controllers.

TABLE VII
SUCCESS RATES IN THE REAL WORLD EXPERIMENTS.

		A^*		Imitation Learning			Spline	Total
	Metric	p&p static	p&p dynamic	cabinet	drawer	door	spline	
PR2	**Success**	**12**	**12**	**28**	**25**	**23**	**24**	**124**
	Base coll.	2	2	0	0	0	0	4
	Arm coll.	0	0	2	0	3	0	5
	Grasp fail	1	1	0	3	4	0	9
	IK fail	0	0	0	2	0	6	8
	Nr. episodes	15	15	30	30	30	30	150
HSR	**Success**	**11**	**12**	**23**	**24**	**24**	**16**	**110**
	Base coll.	0	2	0	0	1	0	3
	Arm coll.	1	0	0	0	1	0	2
	Grasp fail	0	0	6	3	0	0	9
	IK fail	3	1	1	2	4	4	15
	Safety stop	0	0	0	1	0	0	1
	Nr. episodes	15	15	30	30	30	20	140

Notes: The PR2 is evaluated in an environment consisting of two rooms connected by a door. The HSR is directly evaluated in our offices. Each cell denotes the number of episodes. The last column sums over all tasks.

agents' actions by a factor of 0.5 to ensure safe execution of the motions. For the PR2, we also inflate the local occupancy map by $1\,\mathrm{cm}$ for the door opening task and $3\,\mathrm{cm}$ for all the other tasks. Results for all the tasks are reported in Tab. VII. The experiments are shown in Fig. 5 and in the accompanying video.

Pick&Place: We define positions for several tables, marked orange in Fig. 5, as possible pick up and place locations, then draw random pairs of these locations for each episode. We then randomly arrange 2 to 4 obstacles within the map. These obstacles include a large wall segment with a curved star footprint, bins, and a chair for the PR2, and chairs and roll-containers for the HSR. In the second experiment, on top of these static obstacles, we incorporate dynamic obstacles in the form of humans and wheeled objects moved by the authors. For the HSR, we use the *A^*-fwd* planner. We also inflate the local map by $1.5\,\mathrm{cm}$ and include the map as a static layer in the local map, as the robot does not have any sensors in the back. For the dynamic obstacles, we increase the global map inflation from $0.4\,\mathrm{m}$ to $0.5\,\mathrm{m}$ for both robots to obtain smoother trajectories for the end-effector motion generated by the A^*-planner.

Both robots achieve high success rates of 76.6% – 80% for both the static and dynamic obstacles. They demonstrate well-planned movements around obstacles and behavior such as backing out of confined starting poses if necessary. When confronted with dynamic obstacles, they react quickly and are able to evade obstacles that move directly into their base without breaking the end-effector motion. Secondly, the learned behavior is robust to frequently changing end-effector motions as the A^*-planner adapts to dynamic changes in the shortest path.

Spline: For the spline interpolation, we draw waypoints from within a single room of the environment. Both robots are able to follow these motions with success rates of 80%. The main source of failures is too large deviations from the desired end-effector motion when having to follow motions at very large heights (PR2) or unusual orientations (HSR). As an additional difficulty, the HSR's gripper frequently moves so low as to be detected as an obstacle by its base LiDAR. However, the learned policy proved robust to this disturbance.

Cabinet and drawer: For the imitation system motions, we construct tasks with the same objects as in simulation. As the imitation learning system does not incorporate obstacle avoidance into the end-effector motions, we restrict obstacles to objects with a low height, such that the motions can pass over them and choose start poses in the same room as the target object. We rearrange two obstacles every 3 to 5 episodes, deliberately placing at least one of them such that it constrains the opening motion. The PR2 achieves very high success rates of 93.3% and 83.3% on the cabinet and drawer tasks respectively. The HSR succeeds in 76.6% and 80% of all the episodes. Its main failure source is the precision of the grasp, grasping slightly in front of the handle, thereby not opening the object. In one episode, the HSR joints hit a safety limit while being in contact with the drawer which led the controllers to be stopped by the software.

Door: For the PR2, we use the same door and motions as in the simulation. For the HSR, we use a door within our office, with an even smaller frame width of $0.83\,\mathrm{m}$. As the door's opening radius differs, the learned imitation motions do not directly apply to it. Instead, we sample eight waypoints and interpolate them with the spline interpolator to construct an opening motion for the end-effector. As its arm is not strong enough to press down the handle, we do not lock the door's spring for the HSR. The PR2 succeeds in 76.6% of all episodes and performing better than in Gazebo. In general, the real hardware seems more reactive than in the simulator. The HSR achieves success rates of 80%. Failure cases for the PR2 include collisions of the elbow with the door itself, slipping off the handle, and not being able to unlock the door lock. The HSR in a few cases maneuvers to the right of the handle, such that it is not able to pass through the door frame without moving the non-continuous arm roll joints into another configuration (which would require violating the motion constraints). In these situations, it prefers to stay in place until it terminates because the deviations to the desired motion become too large, rather than to produce unsafe behavior such as collisions. The HSR agent proved robust to weird and irregular occupancy maps produced by the glass door.

Across all the tasks, we observe that the learned behaviors directly transfer to the real world with average success rates of 82.6% and 78.6% on the PR2 and HSR respectively. The agents learned to efficiently avoid obstacles and cause very few base collisions. They rather fail the arm motions than drive into an obstacle. The produced motions are robust to noise such as localization error, which could be significant whenever the number of unmapped obstacles are large. Although the overall motions are smooth, at higher speeds the arm motions can sometimes become a bit shaky. This has several reasons: the low-level base and arm controllers are independent and as such cannot take into account each other's errors. Secondly, the hardware introduces several additional noise sources that are not present in the simulation. These include asynchronous control, localization errors, calibration errors, unseen obstacle geometries, and artifacts in the occupancy maps. We found that the acceleration regularization introduced in Sec. III is essential for the smoothness. Finetuning on the real system is a promising avenue to further increase the precision and the feasible velocities on the real systems. The HSR's main failure source are the IK failures. A number of these occurred due to conflicts between the head and arm. Unlike in training in the real world, the head moved to focus the camera on the target object. Thereby leading to different self-collision constraints.

V. Conclusion

We introduced N^{2}M^2, which extends the formulation of kinematic feasibility for mobile manipulation to complex unstructured environments. We generalized its objective function and extended the agent's control to the velocity of the end-effector motions and prevent configuration jumps by learning the torso joints and introducing a regularization to the inverse kinematics. We then introduced a procedurally generated training environment that uses strong randomization and simple elements to produce diverse scenarios. The result is a powerful approach that can successfully act in unseen human-centered real-world environments. In extensive experiments across a variety of robots, physics, and environments, we demonstrated that this approach successfully generalizes in a zero-shot manner to novel tasks, unseen objects and geometries, and dynamic obstacles. By leveraging a hybrid combination of inverse kinematics and reinforcement learning, the agent solves tasks with a vast continuous configuration space in which previous state-of-the-art approaches struggle. Our method outperforms these approaches both in the analytical environment and in the transfer to unseen environments on all robotic platforms. Fig. 6 summarizes the success rates across the different robots and environments.

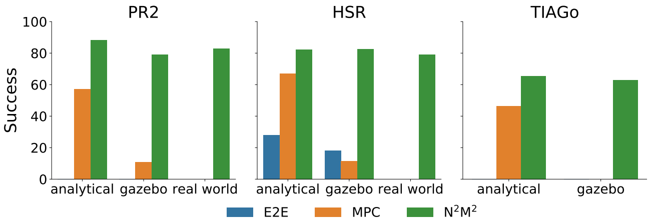


Fig. 6. Average success rates across different robots and environments. Given the low success rates in the analytical environment and the simulated gazebo environment leading to an increased risk of damaging the robot, we did not evaluate the MPC and E2E baselines in the real world.

In this work, we have focused on achieving arbitrary motions and used simple robot-agnostic end-effector motion generators. We purposefully abstracted from optimizing the motions or goals themselves to demonstrate the system's capabilities to achieve challenging motions. In the future, we plan to jointly or iteratively optimize the robot's behaviors and the generated end-effector motions. A particularly exciting direction is to incorporate this work into hierarchical approaches that learn to produce motions or subgoals for the end-effector to achieve high-level goals. Such an approach will benefit from the ability to abstract from complex base behavior to reason in a much simpler space of end-effector motions. This can be done both within a learning-based paradigm or on the motion planning level of task-and-motion planning based pipelines.

Further work includes the incorporation of partially observable and 3D obstacle avoidance for the robot arm. The flexibility of the RL approach means that this can be incorporated based on voxel maps or directly learned from camera inputs. The development of joint low-level controllers and finetuning of the learned policies in the real world are promising avenues for further improvements in precision and velocities. Lastly, we find that the current reinforcement learning methods still face difficulties to explore certain high-dimensional continuous action spaces, as exhibited by the TIAGo robot. Methods to alleviate this problem will be important for robotics. A particularly interesting direction for mobile manipulation is the combination of value learning methods with efficient Monte-Carlo rollouts, combining the best of MPC and learning based approaches.

Acknowledgements

This work was funded by the European Union's Horizon 2020 research and innovation program under grant agreement No 871449-OpenDR. We thank Adrian Röfer and Niklas Wetzel for feedback on the initial draft of the paper.

References

[1] K. Blomqvist, M. Breyer, A. Cramariuc, J. Förster, M. Grinvald, F. Tschopp, J. J. Chung, L. Ott, J. Nieto, and R. Siegwart, "Go fetch: Mobile manipulation in unstructured environments," *arXiv preprint arXiv:2004.00899*, 2020.

[2] J. V. Hurtado, L. Londoño, and A. Valada, "From learning to relearning: A framework for diminishing bias in social robot navigation," *Frontiers in Robotics and AI*, vol. 8, p. 69, 2021.

[3] K. Sirohi, R. Mohan, D. Büscher, W. Burgard, and A. Valada, "Efficientlps: Efficient lidar panoptic segmentation," *IEEE Transactions on Robotics*, 2021.

[4] F. R. Valverde, J. V. Hurtado, and A. Valada, "There is more than meets the eye: Self-supervised multi-object detection and tracking with sound by distilling multimodal knowledge," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 612–11 621.

[5] J. Duan, S. Yu, H. L. Tan, H. Zhu, and C. Tan, "A survey of embodied ai: From simulators to research tasks," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2022.

[6] A. Younes, D. Honerkamp, T. Welschehold, and A. Valada, "Catch me if you hear me: Audio-visual navigation in complex unmapped environments with moving sounds," *arXiv preprint arXiv:2111.14843*, 2021.

[7] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Robot placement based on reachability inversion," in *Int. Conf. on Robotics & Automation*, 2013.

[8] F. Paus, P. Kaiser, N. Vahrenkamp, and T. Asfour, "A combined approach for robot placement and coverage path planning for mobile manipulation," in *Int. Conf. on Intelligent Robots and Systems*, 2017.

[9] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[10] J. Kuffner and S. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Int. Conf. on Robotics & Automation*, vol. 2, 2000, pp. 995–1001.

[11] F. Burget, M. Bennewitz, and W. Burgard, "Bi2rrt*: An efficient sampling-based path planning framework for task-constrained mobile manipulation," in *Int. Conf. on Intelligent Robots and Systems*, 2016.

[12] J. Pankert and M. Hutter, "Perceptive model predictive control for continuous mobile manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6177–6184, 2020.

[13] M. V. Minniti, R. Grandia, K. Fäh, F. Farshidian, and M. Hutter, "Model predictive robot-environment interaction control for mobile manipulation tasks," in *Int. Conf. on Robotics & Automation*, 2021.

[14] F. Xia, C. Li, R. Martín-Martín, O. Litany, A. Toshev, and S. Savarese, "Relmogen: Leveraging motion generation in reinforcement learning for mobile manipulation," in *Int. Conf. on Robotics & Automation*, 2021.

[15] J. Wong, A. Tung, A. Kurenkov, A. Mandlekar, L. Fei-Fei, S. Savarese, and R. Martín-Martín, "Error-aware imitation learning from teleoperation data for mobile manipulation," in *Proc. of the Conference on Robot Learning*, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164, 2022, pp. 1367–1378.

[16] F. Schmalstieg, D. Honerkamp, T. Welschehold, and A. Valada, "Learning long-horizon robot exploration strategies for multi-object search in continuous action spaces," *arXiv preprint arXiv:2205.11384*, 2022.

[17] J. Kindle, F. Furrer, T. Novkovic, J. J. Chung, R. Siegwart, and J. Nieto, "Whole-body control of a mobile manipulator using end-to-end reinforcement learning," *arXiv preprint arXiv:2003.02637*, 2020.

[18] D. Honerkamp, T. Welschehold, and A. Valada, "Learning kinematic feasibility for mobile manipulation through deep reinforcement learning," *IEEE Robotics and Automation Letters*, 2021.

[19] R. Diankov, S. S. Srinivasa, D. Ferguson, and J. Kuffner, "Manipulation planning with caging grasps," in *Int. Conf. on Humanoid Robots*. IEEE, 2008, pp. 285–292.

[20] M. Arduengo, C. Torras, and L. Sentis, "Robust and adaptive door operation with a mobile robot," *Intelligent Service Robotics*, vol. 14, no. 3, pp. 409–425, 2021.

[21] J. Liu, P. Balatti, K. Ellis, D. Hadjivelichkov, D. Stoyanov, A. Ajoudani, and D. Kanoulas, "Garbage collection and sorting with a mobile manipulator using deep learning and whole-body control," in *Int. Conf. on Humanoid Robots*, 2021, pp. 408–414.

[22] S. Jauhri, J. Peters, and G. Chalvatzaki, "Robot learning of mobile manipulation with reachability behavior priors," *arXiv preprint arXiv:2203.04051*, 2022.

[23] F. Burget, A. Hornung, and M. Bennewitz, "Whole-body motion planning for manipulation of articulated objects," in *Int. Conf. on Robotics & Automation*, 2013, pp. 1656–1662.

[24] O. Arslan and P. Tsiotras, "Use of relaxation methods in sampling-based algorithms for optimal motion planning," in *Int. Conf. on Robotics & Automation*. IEEE, 2013, pp. 2421–2428.

[25] M. Otte and E. Frazzoli, "Rrtx: Real-time motion planning/replanning for environments with unpredictable obstacles," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 461–478.

[26] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *Int. Journal of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, 2011.

[27] D. Leidner, A. Dietrich, F. Schmidt, C. Borst, and A. Albu-Schäffer, "Object-centered hybrid reasoning for whole-body mobile manipulation," in *Int. Conf. on Robotics & Automation*, 2014, pp. 1828–1835.

[28] T. Welschehold, C. Dornhege, F. Paus, T. Asfour, and W. Burgard, "Coupling mobile base and end-effector motion in task space," in *Int. Conf. on Intelligent Robots and Systems*, 2018.

[29] J. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, "A unified MPC framework for whole-body dynamic locomotion and manipulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4688–4695, 2021.

[30] M. Mittal, D. Hoeller, F. Farshidian, M. Hutter, and A. Garg, "Articulated object interaction in unknown scenes with whole-body mobile manipulation," *arXiv preprint arXiv:2103.10534*, 2021.

[31] J. Haviland, N. Sunderhauf, and P. Corke, "A holistic approach to reactive mobile manipulation," *IEEE Robotics and Automation Letters*, 2022.

[32] R. Ancona, "Redundancy modelling and resolution for robotic mobile manipulators: a general approach," *Advanced Robotics*, vol. 31, no. 13, pp. 706–715, 2017.

[33] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Perez, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, 2021.

[34] M. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," in *Robotics: Science and Systems*, 2018.

[35] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling, "Ffrob: Leveraging symbolic planning for efficient task and motion planning," *Int. Journal of Robotics Research*, vol. 37, no. 1, pp. 104–136, 2018.

[36] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin, "Combining optimal control and learning for visual navigation in novel environments," in *Proc. of the Conference on Robot Learning*, vol. 100, 2020, pp. 420–429.

[37] T. El-Gaaly, C. Tomaszewski, A. Valada, P. Velagapudi, B. Kannan, and P. Scerri, "Visual obstacle avoidance for autonomous watercraft using smartphones," *Autonomous Robots and Multirobot Systems Workshop (ARMS)*, 2013.

[38] M. Kollmitz, D. Büscher, and W. Burgard, "Predicting obstacle footprints from 2d occupancy maps by learning from physical interactions," in *Int. Conf. on Robotics & Automation*, 2020, pp. 10 256–10 262.

[39] R. Guldenring, M. Görner, N. Hendrich, N. J. Jacobsen, and J. Zhang, "Learning local planners for human-aware navigation in indoor environments," in *Int. Conf. on Intelligent Robots and Systems*, 2020, pp. 6053–6060.

[40] D. Hoeller, L. Wellhausen, F. Farshidian, and M. Hutter, "Learning a state representation and navigation in cluttered and dynamic environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5081–5088, 2021.

[41] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *Int. Conf. on Intelligent Robots and Systems*, 2018, pp. 3052–3059.

[42] U. Patel, N. Kumar, A. J. Sathyamoorthy, and D. Manocha, "Dwa-rl: Dynamically feasible deep reinforcement learning policy for robot navigation in dense mobile crowds," in *Int. Conf. on Robotics & Automation*, 2021.

[43] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *Int. Conf. on Intelligent Robots and Systems*, 2017.

[44] L. Han, F. Gao, B. Zhou, and S. Shen, "Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots," in *Int. Conf. on Intelligent Robots and Systems*. IEEE, 2019, pp. 4423–4430.

[45] K. Ehsani, W. Han, A. Herrasti, E. VanderBilt, L. Weihs, E. Kolve, A. Kembhavi, and R. Mottaghi, "Manipulathor: A framework for visual object manipulation," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2021, pp. 4497–4506.

[46] D. Batra, A. X. Chang, S. Chernova, A. J. Davison, J. Deng, V. Koltun, S. Levine, J. Malik, I. Mordatch, R. Mottaghi *et al.*, "Rearrangement: A challenge for embodied ai," *arXiv preprint arXiv:2011.01975*, 2020.

[47] S. Srivastava, C. Li, M. Lingelbach, R. Martín-Martín, F. Xia, K. E. Vainio, Z. Lian, C. Gokmen, S. Buch, K. Liu *et al.*, "Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments," in *Proc. of the Conference on Robot Learning*, 2022, pp. 477–490.

[48] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. Chaplot, O. Maksymets, A. Gokaslan, V. Vondrus, S. Dharur, F. Meier, W. Galuba, A. Chang, Z. Kira, V. Koltun, J. Malik, M. Savva, and D. Batra, "Habitat 2.0: Training home assistants to rearrange their habitat," *arXiv preprint arXiv:2106.14405*, 2021.

[49] C. Gan, S. Zhou, J. Schwartz, S. Alter, A. Bhandwaldar, D. Gutfreund, D. L. K. Yamins, J. J. DiCarlo, J. H. McDermott, A. Torralba, and J. B. Tenenbaum, "The threedworld transport challenge: A visually guided task-and-motion planning benchmark for physically realistic embodied AI," *arXiv preprint arXiv:2103.14025*, 2021.

[50] L. P. Kaelbling, "Learning to achieve goals," in *Proc. of the 13th Int. Joint Conference on Artificial Intelligence*, 1993, pp. 1094–1098.

[51] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proc. of the 32nd Int. Conf. on Machine Learning*, vol. 37, 2015, pp. 1312–1320.

[52] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[53] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proc. of the National Conference on Artificial Intelligence*, 2017.

[54] L. P. Kaelbling, "Hierarchical learning in stochastic domains: Preliminary results," in *Proc. of the Int. Conf. on Machine Learning*, vol. 951, 1993, pp. 167–173.

[55] C. Li, F. Xia, R. Martin, and S. Savarese, "Hrl4in: Hierarchical reinforcement learning for interactive navigation with mobile manipulators," in *Conference on Robot Learning*, 2019.

[56] A. Röfer, G. Bartels, W. Burgard, A. Valada, and M. Beetz, "Kineverse: A symbolic articulation model framework for model-agnostic mobile manipulation," *IEEE Robotics and Automation Letters*, 2022.

[57] P. Ruppel, N. Hendrich, S. Starke, and J. Zhang, "Cost functions to specify full-body motion and multi-goal manipulation tasks," in *Int. Conf. on Robotics & Automation*. IEEE, 2018, pp. 3152–3159.

[58] F. Pardo, A. Tavakoli, V. Levdik, and P. Kormushev, "Time limits in reinforcement learning," in *Proc. of the Int. Conf. on Machine Learning*, vol. 80, 2018, pp. 4045–4054.

[59] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. of the Int. Conf. on Machine Learning*, vol. 80, 2018, pp. 1861–1870.

[60] P. Klink, C. D'Eramo, J. R. Peters, and J. Pajarinen, "Self-paced deep reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9216–9227, 2020.

[61] M. Dennis, N. Jaques, E. Vinitsky, A. Bayen, S. Russell, A. Critch, and S. Levine, "Emergent complexity and zero-shot transfer via unsupervised environment design," in *Proc. of the Conf. on Neural Information Processing Systems*, 2020.

[62] D. Coleman, I. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: a moveit! case study," *Journal of Software Engineering for Robotics*, vol. 5, no. 1, pp. 3–16, 2014.

[63] T. Welschehold, C. Dornhege, and W. Burgard, "Learning mobile manipulation actions from human demonstrations," in *Int. Conf. on Intelligent Robots and Systems*, 2017.

[64] AWS RoboMaker, "aws-robomaker-bookstore-world," https://github.com/aws-robotics/aws-robomaker-bookstore-world, 2019.

B

Learning Long-Horizon Robot Exploration Strategies for Multi-Object Search in Continuous Action Spaces

Fabian Schmalstieg*, Daniel Honerkamp*, Tim Welschehold, and Abhinav Valada

Abstract Recent advances in vision-based navigation and exploration have shown impressive capabilities in photorealistic indoor environments. However, these methods still struggle with long-horizon tasks and require large amounts of data to generalize to unseen environments. In this work, we present a novel reinforcement learning approach for multi-object search that combines short-term and long-term reasoning in a single model while avoiding the complexities arising from hierarchical structures. In contrast to existing multi-object search methods that act in granular discrete action spaces, our approach achieves exceptional performance in continuous action spaces. We perform extensive experiments and show that it generalizes to unseen apartment environments with limited data. Furthermore, we demonstrate zero-shot transfer of the learned policies to an office environment in real world experiments.

1 Introduction

Exploration and navigation of unmapped 3D environments is an important task for a wide range of applications across both service and industrial robotics. Research in Embodied AI has made substantial progress in integrating high-dimensional observations in a range of navigation and exploration tasks [22, 4, 8]. Recent work has introduced several tasks around multi-object search and exploration [11, 25]. These tasks are particularly challenging in unmapped environments, as they require balancing long-term reasoning about where to go with short-term control and collision avoidance. Without prior knowledge of a floor plan, there is often no obvious optimal policy.

Furthermore, the combination of complex observation space and long horizons remains an open problem with success rates quickly decreasing as the distance to the goal or the number of objects in the task increases.

*These authors contributed equally.

All authors are with the Department of Computer Science, University of Freiburg, Germany,

This work was funded by the European Union's Horizon 2020 research and innovation program under grant agreement No 871449-OpenDR.

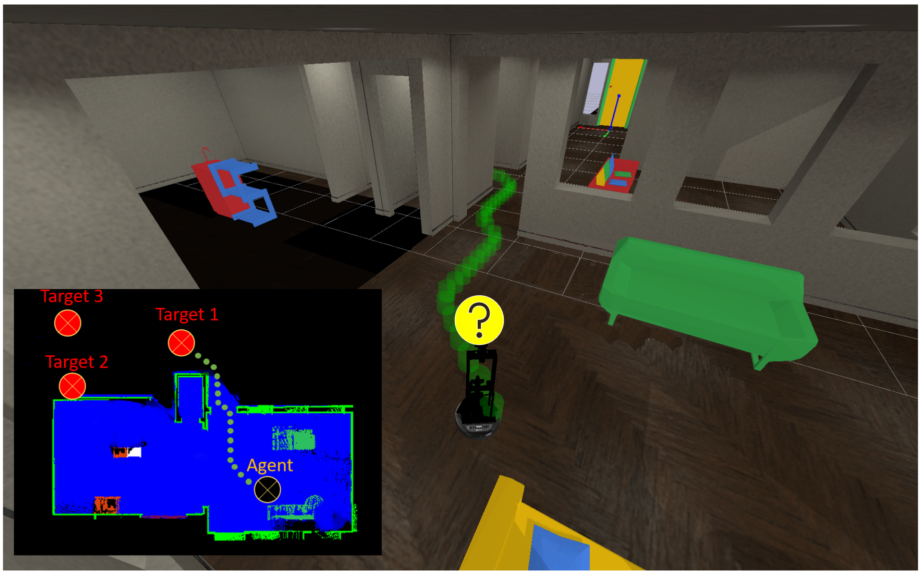


Fig. 1 Starting in an unexplored map and given a set of target objects, the robot faces the complex decision on how to most efficiently find these objects. Our approach continuously builds a semantic map of the environment and learns to combine long-term reasoning with short-term decision making into a single policy by predicting the direction of the path towards the closest target object, shown in green. Note that the agent does not receive the path consisting of the waypoints or the location of the objects.

Previous work has in particular focused on constructing rich map representations and memories for these agents [11, 25, 24] and demonstrated their benefits while acting in granular discrete action spaces. The long-horizon nature of these tasks poses a significant challenge for learning-based methods, as they struggle to learn longer-term reasoning and to explore efficiently over long horizons. This problem is strongly exacerbated in fine-grained continuous action spaces. A common strategy to mitigate this challenge is to instead learn high-level waypoints which are then fed to a low-level controller [5, 7]. While this can simplify the learning problem by acting in a lifted MDP with a much shorter horizon length, it limits the ability of the agent to simultaneously learn to control other aspects such as its camera or arms at a higher control frequency. During inference, high-level actions can be taken at arbitrary frequencies by executing them in a model predictive control style manner but the agent is bound to a low control frequency during training.

In this work, we present a novel approach to reason about long-horizon exploration while learning to directly act in continuous action spaces at high control frequencies and demonstrate its effectiveness in multi-object search tasks.

Our approach learns to predict the direction of the path towards the closest target object. It then learns a policy that observes this prediction, enabling it to express long-term intentions while taking short-term actions based on the full context. As a consequence, the policy can incorporate expected inaccuracies and uncertainties in the predictions and balance strictly following its intentions with short-term exploration and collision avoidance. Figure 1 illustrates the task and our approach.

To jointly learn this behavior with a single model, we introduce a learning curriculum that balances groundtruth and learned intentions. In contrast to many learning-based navigation approaches building on complex features from color or depth images that make generalization to unseen environments hard and data intensive [15], we learn purely from structured, semantic map representations that have more universal features and are therefore more independent of the specific training environment and show only minimal sim-to-real gap. We perform extensive experimental evaluations and demonstrate that our proposed approach achieves strong generalization to unseen environments from a very small number of training scenes and directly transfers to the real world. Lastly, the combination of these simple inputs and expressive navigation intentions ensures good interpretability of the agent's decisions and failures. To the best of our knowledge, this is the first real-world demonstration of learning-based multi-object search tasks.

To summarise, this paper makes the following contributions:

- We present a novel search agent that unifies long-horizon decision making and frequent low-level control into a single time-scale and model.
- We propose the first multi-object search task in continuous action space.
- We demonstrate that our approach is capable to learn from very limited training data and achieves strong generalization performance in unseen apartment environments and zero-shot transfer to the real world.
- We make the code publicly available at `http://multi-object-search.cs.uni-freiburg.de`.

2 Related Work

Embodied AI tasks: Navigation in unmapped 3D environments has attracted a lot of attention in recent work on embodied AI, covering a range of tasks. In PointGoal navigation [22, 4], the agent at each step receives the displacement vector to the goal that it has to reach. Whereas, in AudioGoal navigation [6, 27], the agent at each step receives an audio signal emitted by a target object. Conversely, in ObjectGoal navigation [29, 5, 20, 10], the agent receives an object category that it has to navigate to.

Extending the ObjectGoal task, Beeching *et al.* [2] propose an ordered k-item task in a Viz Doom environment, in which the agent has to find k items in a fixed order. Similarly in the MultiOn task [25] the agent has to find k objects in realistic 3D environments. Fang *et al.* [11] propose an object search task in which the agent has to find k items in arbitrary order. Here the target object locations are defined by the given dataset, i.e. are in a fixed location in each apartment. In contrast, we use a random distribution over the target locations, strongly increasing the diversity. While all of these works focus on discrete actions (move forward, turn left, turn right, stop), we show that our approach can directly learn in the much larger continuous action space and demonstrate the direct transfer to the real world.

Object search and exploration: Approaches for multi-object search and navigation tasks fall into two categories: implicit memory agents and agents that explicitly construct a map of the environment as a memory. Agents without an explicit map include direct visual end-to-end learning from RGB-D images as well as FRMQN [19] and SMT [11] which store an embedding of each observation in memory, then retrieve relevant information with an attention mechanism. On the other side of the spectrum, we have agents that project the RGB-D inputs into a global map, building up an explicit memory. They then commonly extend this representation with semantic object annotations [13, 2]. Wani *et al.* [25] provide a comprehensive comparison of these methods. While they train separate agents for each number of target objects, we train a single agent that generalizes to different numbers of target objects.

SGoLAM [16] combine a mapping and a goal detection module. Then either employ frontier exploration if no goal object is in sight or if a goal is detected, they use a D*-planner to move closer to the goal. They achieve strong results without any learning component. Closely related to object search, previous work has also focused on pure exploration of realistic apartments. This includes frontier based approaches [26] as well as reinforcement learning with the aim to maximise coverage [8].

Learning long horizon goals: Multi-object search and exploration with an embodied agent combine long-horizon thinking with short-horizon control to navigate and avoid collisions. This can pose a challenge for learning-based approaches. This has previously been mitigated by learning higher-level actions at a lower control frequency such as learning to set waypoints or to directly predict task goals [18], which then get passed down to a lower-level planner for navigation [5, 7]. While this shortens the horizon of the MDP the agent is acting in, it makes it difficult to learn to simultaneously control other aspects at a lower time scale, such as controlling a camera joint or a manipulator arm. Our approach directly learns at a high control frequency and as such can directly be extended to integrate such aspects. In our experiments, we furthermore demonstrate that direct prediction of the goal locations does not generalize well for multi-object search. The long horizon problem is further exacerbated in continuous control tasks. While most existing work focuses on granular discrete actions [11, 25, 5, 16], our approach succeeds in a continuous action space.

Mapping: Spatial maps built with Simultaneous Localization and Mapping (SLAM) have been used for tasks such as exploration [28, 3] and FPS games [2]. Both occupancy and semantic maps have commonly been used in embodied AI tasks [25, 16] and several works have presented approaches to build such maps in complex environments [4, 5]. In our approach, we assume access to a method to build such maps.

3 Learning Long-Horizon Exploration

In this section, we first define the multi-object search task and formulate it as a reinforcement-learning problem in Section 3.1. We then introduce our approach for learning a novel predictive task and an effective method to jointly learn low- and high-level reasoning in Section 3.2.

3.1 Problem Statement

In each episode, the agent receives a list of up to k object categories, drawn randomly from a total set of c categories. It then has to search and navigate to these objects which are spawned randomly within an unmapped environment. An object is considered found if the agent has seen it and navigates up to a vicinity of 1.3 m of the target object. In contrast to MultiOn [25], we require a single agent to learn to find variable numbers of target objects and focus on unordered search, meaning the agent can find these objects in any order it likes. On one hand, this provides the agent with more freedom. On the other hand, the optimal shortest-path policy is non-trivial, even in a mapped environment, making this a very challenging task to solve optimally.

This can be formulated as a goal-conditional Partially Observable Markov Decision Process (POMDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T(s'|s,a), P(o|s), R(s,a,g))$, where $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{O}$ are the state, action and observation spaces, $T(s'|s,a)$ and $P(o|s)$ describe the transition and observation probabilities and $R(s,a,g)$ is the reward function. At each step, the agent receives a visual observation o from an RGB-D and semantic camera together with a binary vector g indicating which objects it must find. Its aim is to learn a policy $\pi(a|o,g)$ that maximises the discounted, expected return $\mathbb{E}_\pi[\sum_{t=1}^{T} \gamma^t R(s_t, a_t, g)]$, where γ is the discount factor. The agent acts in the continuous action space of a LoCoBot robot, controlling the linear and angular velocities of its differential drive. During training, these actions are executed at a control frequency of 10 Hz. The agent receives a sparse reward of 10 whenever it finds a target object. It furthermore receives a dense time penalty of -0.0025 per step, a distance reward for getting closer to the next target object, and a collision penalty of -0.1. An episode ends after successfully finding all objects, exceeding 600 collisions or exceeding 3500 steps.

3.2 Technical Approach

We propose a reinforcement learning approach that consists of three components: a mapping module, a predictive module to learn long-horizon intentions, and a reinforcement learning policy. The full approach is depicted in Figure 2.

Mapping: The mapping module uses the 128×128 pixels depth and semantic camera to project the points into a local top-down map. From this, it then updates an internal global map which is further annotated with the agent's trace and encoded into a standard RGB image. In each step, the agent then extracts an egocentric map from it and passes two representations of this map to the encoder: a coarse map of dimension $224 \times 224 \times 3$ at a resolution of 6.6 cm and a fine-grained map of dimension $84 \times 84 \times 3$ at a resolution of 3.3 cm. Meaning they cover 14.8 m × 14.8 m and 2.77 m × 2.77 m, respectively. After the agent has segmented an object correctly, it updates the object's annotation to a fixed color-coding to mark the corresponding object as "found". The coarse map is then encoded into a 256-dimensional feature vector and the fine map into a 128-dimensional feature vector using a convolutional neural network, before being concatenated into the feature embedding f_t. The coarse map is passed through a ResNet-18 [12] pre-trained on

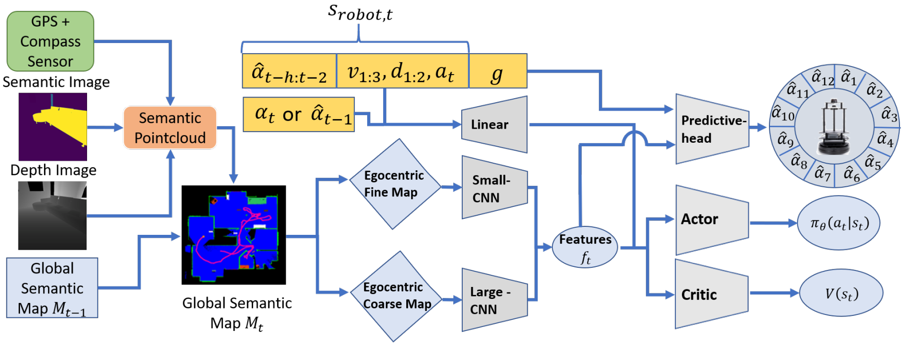

$s_{robot,t}$
GPS + Compass Sensor
Semantic Image
Depth Image
Global Semantic Map M_{t-1}
Semantic Pointcloud
Global Semantic Map M_t
$\hat{\alpha}_{t-h:t-2}$
$v_{1:3}, d_{1:2}, a_t$
g
α_t or $\hat{\alpha}_{t-1}$
Linear
Egocentric Fine Map
Egocentric Coarse Map
Small-CNN
Large - CNN
Features f_t
Predictive-head
Actor
Critic
$\pi_\theta(a_t|s_t)$
$V(s_t)$
$\hat{\alpha}_{12}$ $\hat{\alpha}_1$ $\hat{\alpha}_2$ $\hat{\alpha}_3$ $\hat{\alpha}_4$ $\hat{\alpha}_5$ $\hat{\alpha}_6$ $\hat{\alpha}_7$ $\hat{\alpha}_8$ $\hat{\alpha}_9$ $\hat{\alpha}_{10}$ $\hat{\alpha}_{11}$

Fig. 2 Our proposed model architecture. The mapping module aggregates depth and semantic information into a global map. The predictive module learns long-horizon relationships which are then interpreted by a reinforcement learning policy. During training, the agent receives a state vector with either the groundtruth direction to the closest object α_t or its prediction $\hat{\alpha}_t$. At test time it always receives its prediction. It furthermore receives its previous predictions $\hat{\alpha}_{t-18:t-2}$, the variances of its x- and y-position v_1 and v_2, the circular variance of its predictions v_3, a collision flag d_1, the sum over the last 16 collisions d_2, its previous action a_t, and a binary vector g indicating the objects the agents have to find.

ImageNet [9], while the local map is encoded by a much simpler three-layer CNN with 32, 64, and 64 channels and strides 4, 2, and 1.

Learning Long-horizon Reasoning: While directly learning low-level actions has shown success in granular discrete environments [11, 25], this does not scale to continuous environments as we show in Section 4. We hypothesize that this is due to being unable to explore the vast state-action space efficiently. To learn long-horizon reasoning within a single model, we introduce a predictive task to express the agent's navigation intentions. In particular, the agent learns to predict the direction of the shortest path to the currently closest target object. It does so by estimating the angle to a waypoint generated by an A^* planner in a distance of roughly $0.4\,\mathrm{m} - 0.55\,\mathrm{m}$ from the agent (varying due to the discretized grid of the planner) as illustrated in Figure 1. The plan is generated in a map inflated by 0.2 m to avoid waypoints close to walls or obstacles.

The prediction is parameterized as a neural network $\hat{\alpha} = f_{pred}(f_t, s_{robot,t}, g)$ that takes a shared feature encoding f_t from the map encoding, the robot state $s_{robot,t}$ without the groundtruth vector α_t nor the prediction $\hat{\alpha}_t$ and the goal objects g and predicts a vector of probabilities over the discretized angles to this waypoint. In particular, we discretize the angle into 12 bins and normalize the outputs with a softmax function.

This task is used both in the form of an auxiliary task to shape representation learning, propagating gradients into a shared map encoder, as well as a recursive input to the agent, allowing it to guide itself as the agent observes the previous step's predictions both as an overlay in the ego-centric map and the robot state.

The predictive module minimizes the cross-entropy between the predictions and a one-hot encoding of the groundtruth angle α given by

$$\mathcal{L}_{pred} = \frac{1}{N}\sum_{i=1}^{N} \alpha_i \log \hat{\alpha}_i. \tag{1}$$

The discrete distribution enables the agent to cover multi-modal hypotheses, for example when standing on a floor with several unexplored directions. As a result, the prediction can vary more smoothly over time in contrast to commonly used uni-modal distributions such as a Gaussian, which can fluctuate rapidly if the most likely direction changes to another door.

Policy: The policy receives the concatenated features f_t of the flattened map encodings, the robot state, and a history of its predictions. The policy is then learned with Proximal Policy Optimization (PPO) [23]. The actor and critic are parameterized by a two-layer MLP network with 64 hidden units each and a gaussian policy. The total loss $\mathcal{L}$ given by

$$\mathcal{L} = \mathcal{L}_{ppo} + \lambda\mathcal{L}_{pred}, \tag{2}$$

is jointly minimized with the Adam optimizer [17].

Training: During training we have the choice to provide the policy either with the groundtruth direction or its prediction: Providing it with the groundtruth vector results in strong coupling between the agent's navigation and the maximum value of the vector, which points to the waypoint corresponding to the closest next object. Hence, when deploying the agent with the auxiliary prediction, the agent blindly follows its predictions. Instead, we want to learn a policy that can, on one hand, react to suboptimal intentions and on the other hand can learn more optimal paths than simply finding the closest object. For such behavior, it is desirable to train the policy directly with its predictions.

To avoid instabilities from initially very suboptimal predictions, we introduce a learning curriculum that balances observing the groundtruth and the agent's predictions.

The curriculum starts with a probability of 16% to observe an entire episode with the prediction and otherwise receives the groundtruth. Once the agent has exceeded a success rate of 50%, we linearly increase this probability by 2% every 4 episodes up to a maximum of 72%. This curriculum enables the robot to react to prediction errors and correct its navigation accordingly. To avoid learning a simple mapping from groundtruth to prediction during training, this α is never observed by the predictive module and only passed to the policy.

In specific situations where it is hard to assess a global strategy, the agent sometimes predicts alternating auxiliary angles. We hypothesize that the reason for these highly alternating predictions is that the training episodes run purely with predictions, suffer from suboptimal predictions and navigation and will therefore accumulate more errors contributing to the overall loss. We test this by disabling the gradient flow from the predictive head back into the rest of the network for the episodes with predictions. We observe that this leads to a lower prediction loss. But at the same time the predictions can no longer shape the policy's representations and thus, the agent at times struggles with "imperfect" episodes, when deployed. This is reflected, in a tremendous drop in the success rate when executing prediction episodes. We conclude that the gradient flow from all episodes is a crucial component.

With the focus on structured map inputs and the learning curriculum enables, we are able to train the agent in the comparatively small number of 4,500,000 steps.

4 Experimental Evaluations

To demonstrate the effectiveness of our approach, we perform extensive evaluations in simulation and the real world. With these experiments we aim to answer the following questions:

- Does the learned behavior generalize to unseen environments?
- Does the agent learn to efficiently use the long-term prediction? Does this lead to more efficient exploration than using alternative approaches?
- Does the learned behavior generalize to the real world?

4.1 Experimental Setup

We train a LoCoBot robot in the iGibson environment. The LoCoBot has a differential drive and is equipped with an RGB-D camera with a field of view of 79 degrees and a maximum depth of 5.6 m. The action space consists of the linear and angular velocities for the base. We construct tasks of finding 1-6 target objects, matching the hardest setting in previous work [25]. We use the same eight training scenes as the iGibson challenge* and use the remaining seven unseen apartments for evaluation.

As larger datasets such as Matterport3D or Gibson currently do not support semantic camera observations, we leave evaluations on these to future work. The PPO agent is based on an open-source implementation [21].

Evaluation Metrics: We focus on two metrics: the ability to find all target objects, defined by the success rate, and the optimality of the search paths, measured by the success weighted by Path Length (SPL) [1]. We evaluate each scene for 75 episodes, which results in a total of 600 episodes for the train and 525 for the test set for each approach. For the learning based approaches we evaluate the best performing checkpoint on the training scenes. We report the mean over two training seeds for our approach.

Baselines: To test the effectiveness of learning long-range navigation intentions, we compare our approach against a range of action parametrizations as well as a recent non-learning based state-of-the-art approach.
Map-only represents the standard end-to-end reinforcement learning approach and conceptually matches competitive baselines from previous work. It receives the same map and robot state inputs as our agent and acts directly in the action space of the differential drive, but does not learn to predict long-horizon intentions.

* `http://svl.stanford.edu/igibson/challenge2020.html`

Table 1 Hyperparameters used for training. One sensitive parameter is ppo epoch in combination with the clip parameter. Setting the parameter too high causes some behaviour which is similar to catastrophic forgetting.

Parameter	Value	Parameter	Value
clip param	0.1	γ	0.99
ppo epoch	4	lr	0.0001
num mini batch	64	max grad norm	0.5
value loss coef	0.5	optimizer	Adam
entropy coef	0.005		

Goal-prediction Instead of predicting the direction to the next waypoint towards the target, we also evaluate directly predicting the location of the next target object. The agent learns to predict the angle and distance of the next closest target object relative to its current base frame.

SGoLAM [16] combines non-learning based approaches to achieve very strong performance on the CVPR 2021 MultiOn challenge. It explores the map with frontier exploration until it localizes a target object, then switches to a planner to navigate to the target. We reimplement the author's approach for continuous action spaces, closely following the original implementation where possible. While the original implementation relies on two threshold values, namely ϵ and δ, for goal localization, we directly use the semantic camera which finds objects more reliably. Due to this, our implementation improves the performance of SGoLAM.

4.2 Simulation Experiments

To test the ability to learn long-horizon exploration, we first evaluate the approaches on the seen apartments. Table 2 reports the results for all approaches across different numbers of target objects. While the map-only approach that purely learns raw continuous actions finds around three quarters of the single objects, the success rates quickly deteriorate with more target objects. We hypothesize that this is due to the agent getting lost in the vast continuous action space, being unable to meaningfully explore the space. Directly predicting the goal locations slightly improves the performance, but still fails in the majority of cases with five or more objects. This indicates that the agent is not able to meaningfully predict goal locations, as there are a large number of valid hypotheses while the environment is still largely unexplored.

SGoLAM achieves a better performance, yielding an overall success rate of 65.3%. Note that this approach does not rely on a learning component, therefore it has not encountered any of these apartments before. In contrast to the other learning-based approaches, our agent is able to consistently solve this task even for six target objects. In terms of path optimality measured by the SPL in Table 2, we observe a similar case across the approaches. While outperforming the other approaches, both ours and SGoLAM achieve low absolute values. However, note that a perfect SPL would require to directly follow the shortest path to all objects and as such is not achievable without

Table 2 Evaluation of **seen** environments, reporting the success rate (top) and SPL (bottom).

	Model	1-obj	2-obj	3-obj	4-obj	5-obj	6-obj	Avg 1-6
Success	Map-only	75.0	70.6	54.7	53.3	40.3	37.2	55.1
	Goal prediction	78.1	72.3	58.1	54.4	46.7	43.0	58.7
	SGoLAM	82.0	77.9	62.8	60.5	58.0	51.0	65.3
	Ours	95.4	90.7	89.0	87.6	85.1	83.4	**88.5**
SPL	Map-only	33.5	31.0	26.8	24.9	20.8	21.3	26.3
	Goal prediction	31.4	29.8	25.9	23.1	18.3	22.3	25.1
	SGoLAM	41.6	34.5	32.0	33.7	36.6	38.1	36.0
	Ours	46.4	40.9	42.9	44.2	49.2	53.1	**46.1**

Table 3 Evaluation of **unseen** environments, reporting the success rate (top) and SPL (bottom).

	Model	1-obj	2-obj	3-obj	4-obj	5-obj	6-obj	Avg 1-6
Success	Map-only	74.2	64.5	61.0	60.7	57.9	32.5	58.4
	Goal prediction	74.7	69.8	66.7	61.9	56.1	44.0	62.2
	SGoLAM	89.8	85.3	79.9	75.0	74.3	71.1	79.2
	Ours	93.1	89.4	86.4	82.1	82.5	81.1	**85.7**
SPL	Map-only	30.1	27.0	28.6	29.8	29.4	18.4	27.2
	Goal prediction	29.9	21.9	20.1	21.0	19.7	21.4	22.3
	SGoLAM	47.7	44.0	43.7	46.2	47.5	49.8	**46.4**
	Ours	45.3	40.0	38.5	43.2	46.8	49.6	43.9

knowledge of the object positions. Additionally, one needs to bare in mind that the SPL metric is not guaranteed to find the most efficient path as it computes the path in a greedy manner.

Subsequently, we evaluate all the agents in unseen environments and present the results in Table 3. Interestingly, all learning-based approaches achieve similar performance as on the seen apartments, indicating that the semantic map modality generalizes well to unseen apartments. This is particularly impressive as it only has access to eight training scenes. On the other hand, SGoLAM performs better than in the train split, indicating that the test split might be less challenging. While the gap between the best baseline, SGoLAM, and our approach shrinks, it remains considerable with an average difference in success rates of 6.5%. In terms of SPL it even performs slightly better. This may be due to the path planner, which, once an object is in sight, executes the optimal path to this object. While SGoLAM achieves very strong performance in apartments with a lot of open areas, its performance drops severely in more complex apartments with many corridors, rooms next to each other, and generally long-drawn layouts. In contrast, our approach maintains a more even performance across the different apartment layouts.

Qualitatively, we find that the agent learns efficient navigation behaviors. Figure 3 and the accompanying video show example trajectories in unseen apartments. The agent efficiently looks around rooms and learns maneuvers such as a three-point turn. Where

Table 4 Real world multi-object search experiments on the HSR Robot.

Model	2-obj	3-obj	4-obj	6-obj	Total
Success	**6**	**6**	**5**	**5**	**22**
Collision	4	3	5	4	16
Timeout	0	1	0	1	2
Total Episodes	10	10	10	10	40

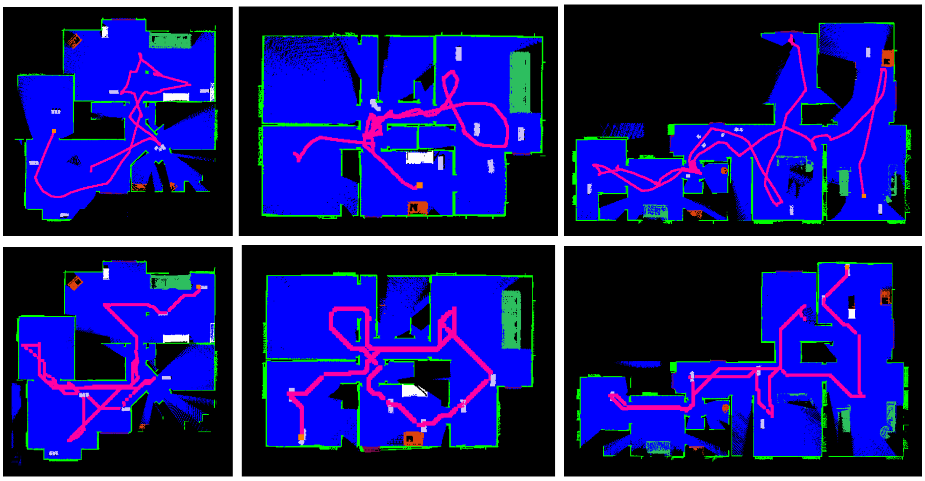


Fig. 3 Example trajectories of our agent (top) and SGoLAM (bottom) in unseen apartments. The maps show the following categories: black: unexplored, blue: free space, green: walls, red: agent trace, grey: (found) target objects, other colors: miscellaneous objects.

confident, it reliably follows its own long-horizon predictions, while deviating if it points into walls, if the predictions are low confidence or if it is possible to explore a lot of additional space with little effort. While SGoLAM randomly picks points on the frontier, often resulting in multiple map crossings and getting lost in very small unexplored spaces, our approach continuously explores and learns to leave out small spots that are unlikely to contain a target object. We further observe an inverse development between the SPL and the success rate with regard to the number of objects for both our approach and SGoLAM. This increase in the SPL most likely stems from a higher number of routes that can be taken which are close to the optimal path. With fewer objects in the scene, large parts of the exploration increase the SPL without finding an object. Nevertheless, this exploration is essential as there is no prior knowledge of the object locations.

4.3 Real World Experiments

To test the transfer to the real world, we transfer the policy onto a Toyota HSR robot. While the HSR has an omnidirectional drive, we restrict its motion to match that of the

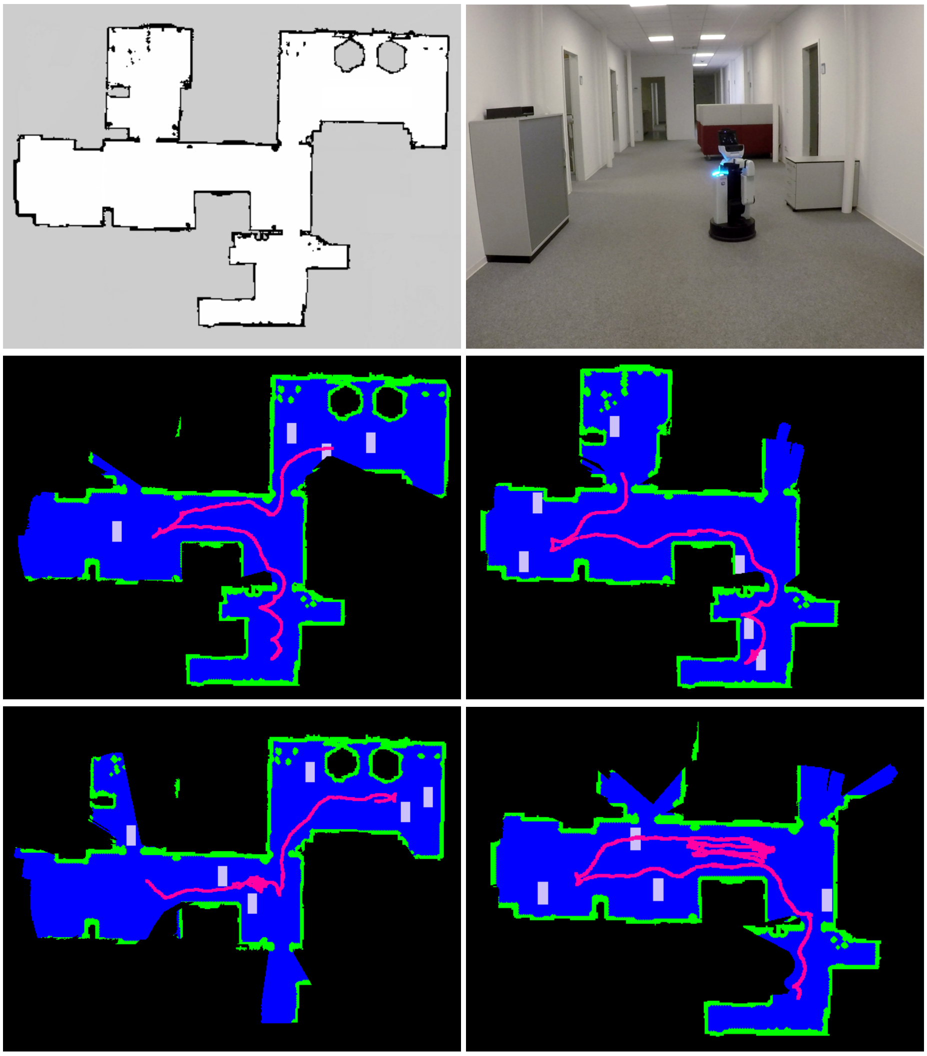


Fig. 4 From top left to bottom right: Floorplan of the real world environment. The HSR robot in the office environment. Example episodes in the real world environment, the (found) target objects are shown in grey. Bottom right: failure case, the agent moves repeatedly back and forth between two rooms until it reaches the timeout.

LoCoBot's differential drive in simulation. We run the experiments in our office building, representing a common office environment. We use rooms covering roughly a size of 320 square meters, excluding small corners that cannot be navigated safely. We assume to have access to an accurate semantic camera. For this, we use the robot's depth camera to construct a map of the explored environment and overlay it with a previously semantically annotated map. In each episode, we randomly spawn 1-6 virtual target objects in this map by adding them to the semantic overlay. The actions are computed on the onboard CPU and executed at roughly 7 Hz. We define a maximum episode length of 6 minutes, roughly matching simulation. The robot is equipped with bumper sensors in its base that stop it upon any collision, in which case we deem the episode unsuccessful. Figure 4 shows the real-world setup.

We make the following adaptations for the real world: To minimize collisions, we inflate the map by 5 cm and scale the actions of the agent by a factor of 0.55. Secondly, we reduce the temperature of the softmax activation of the long-horizon predictions to 0.1. We find that this increases the agent's confidence in its own predictions and leads to more target-driven exploration. We evaluate the agent for a total of 40 episodes, spread across different numbers of target objects. The results from this experiment are presented in Table 4. We observe that the agent successfully transfers to the real world, bridging differences in the robot's motion model, sensors, and environment layouts. Overall it solves 55% of all episodes successfully with almost no decrease as the number of target objects increases. We find two main difficulties in the real world: while generally navigating smoothly, the agent occasionally collides with door frames or small objects such as posts. This may be caused by the mismatch in the robot's motions and controllers as well as due to the training environments consisting of mostly clean edges and little unstructured clutter. Secondly, in a very small number of episodes, the agent gets stuck moving back and forth between close spots, as the long-horizon prediction keeps changing back and forth. These results suggest a potential to further increase the success in the real world by finetuning the learned agent on the real robot, in particular to further reduce the number of collisions. Example trajectories from this experiment are shown in Figure 4 and in the accompanying video.

5 Conclusion

In this paper, we proposed a novel reinforcement learning approach for object search that unifies short- and long-horizon reasoning into a single model. To this end, we introduced a multi-object search task in continuous action space and formulated an explicit prediction task which allows the agent to guide itself over long-horizons. We demonstrated that our approach significantly outperforms other learning-based methods which struggle to perform efficient long-term exploration in this continuous space. By focusing on structured semantic map inputs, our approach learns complex exploration behaviors in comparably few steps and generalizes effectively to unseen apartments. Moreover, we successfully transferred the approach to the real world and find that the agent bridges the sim-real gap and exhibits the potential for further improvement if given the chance to adapt to the motion model of the real robot.

In the future, we aim to further exploit the ability of the approach to learn different actions at a high control frequency. Particular, we will investigate the ability to incorporate control of the head camera which should further improve the agent's success rates. Furthermore, we are interested in the application to mobile manipulation in which the agent has to simultaneously navigate and control its arms [14]. A third promising direction is the ability of learning-based approaches to incorporate data-driven knowledge such as correlations between semantic classes in real-world environments. Training on much larger environments will provide exciting avenues to exploit this.

References

1. Anderson, P., Chang, A., Chaplot, D.S., Dosovitskiy, A., Gupta, S., Koltun, V., Kosecka, J., Malik, J., Mottaghi, R., Savva, M., et al.: On evaluation of embodied navigation agents. arXiv preprint arXiv:1807.06757 (2018)
2. Beeching, E., Debangoye, J., Simonin, O., Wolf, C.: Deep reinforcement learning on a budget: 3d control and reasoning without a supercomputer. In: 25th International Conference on Pattern Recognition (ICPR), pp. 158–165 (2021)
3. Cattaneo, D., Vaghi, M., Valada, A.: Lcdnet: Deep loop closure detection and point cloud registration for lidar slam. IEEE Transactions on Robotics (2022)
4. Chaplot, D.S., Gandhi, D., Gupta, S., Gupta, A., Salakhutdinov, R.: Learning to explore using active neural slam. In: International Conference on Learning Representations (ICLR) (2020)
5. Chaplot, D.S., Gandhi, D.P., Gupta, A., Salakhutdinov, R.R.: Object goal navigation using goal-oriented semantic exploration. Proc. of the Conf. on Neural Information Processing Systems (NeurIPS) **33**, 4247–4258 (2020)
6. Chen, C., Jain, U., Schissler, C., Gari, S.V.A., Al-Halah, Z., Ithapu, V.K., Robinson, P., Grauman, K.: Soundspaces: Audio-visual navigation in 3d environments. In: Proc. of the Europ. Conf. on Computer Vision (ECCV), pp. 17–36 (2020)
7. Chen, C., Majumder, S., Al-Halah, Z., Gao, R., Ramakrishnan, S.K., Grauman, K.: Learning to set waypoints for audio-visual navigation. In: International Conference on Learning Representations (ICLR) (2020)
8. Chen, T., Gupta, S., Gupta, A.: Learning exploration policies for navigation. In: International Conference on Learning Representations (ICLR) (2019)
9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 248–255. Ieee (2009)
10. Druon, R., Yoshiyasu, Y., Kanezaki, A., Watt, A.: Visual object search by learning spatial context. IEEE Robotics and Automation Letters **5**(2), 1279–1286 (2020)
11. Fang, K., Toshev, A., Fei-Fei, L., Savarese, S.: Scene memory transformer for embodied agents in long-horizon tasks. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 538–547 (2019)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016)
13. Henriques, J.F., Vedaldi, A.: Mapnet: An allocentric spatial memory for mapping environments. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 8476–8484 (2018)
14. Honerkamp, D., Welschehold, T., Valada, A.: Learning kinematic feasibility for mobile manipulation through deep reinforcement learning. IEEE Robotics and Automation Letters **6**(4), 6289–6296 (2021)
15. Hurtado, J.V., Londoño, L., Valada, A.: From learning to relearning: A framework for diminishing bias in social robot navigation. Frontiers in Robotics and AI **8**, 69 (2021)
16. Kim, J., Lee, E.S., Lee, M., Zhang, D., Kim, Y.M.: Sgolam: Simultaneous goal localization and mapping for multi-object goal navigation. arXiv preprint arXiv:2110.07171 (2021)

17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
18. Min, S.Y., Chaplot, D.S., Ravikumar, P.K., Bisk, Y., Salakhutdinov, R.: FILM: Following instructions in language with modular methods. In: International Conference on Learning Representations (ICLR) (2022)
19. Oh, J., Chockalingam, V., Lee, H., et al.: Control of memory, active perception, and action in minecraft. In: International Conference on Machine Learning, pp. 2790–2799 (2016)
20. Qiu, Y., Pal, A., Christensen, H.I.: Learning hierarchical relationships for object-goal navigation. In: 2020 Conference on Robot Learning (CoRL) (2020)
21. Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N.: Stable-baselines3: Reliable reinforcement learning implementations. Journal of Machine Learning Research **22**(268), 1–8 (2021)
22. Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., et al.: Habitat: A platform for embodied ai research. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 9339–9347 (2019)
23. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
24. Vödisch, N., Cattaneo, D., Burgard, W., Valada, A.: Continual slam: Beyond lifelong simultaneous localization and mapping through continual learning. arXiv preprint arXiv:2203.01578 (2022)
25. Wani, S., Patel, S., Jain, U., Chang, A., Savva, M.: Multion: Benchmarking semantic map memory using multi-object navigation. Proc. of the Conf. on Neural Information Processing Systems (NeurIPS) **33**, 9700–9712 (2020)
26. Yamauchi, B.: A frontier-based approach for autonomous exploration. In: Proc. of the IEEE Int. Symp. on Comput. Intell. in Rob. and Aut. (CIRA), pp. 146–151 (1997)
27. Younes, A., Honerkamp, D., Welschehold, T., Valada, A.: Catch me if you hear me: Audio-visual navigation in complex unmapped environments with moving sounds. arXiv preprint arXiv:2111.14843 (2021)
28. Zhang, J., Tai, L., Boedecker, J., Burgard, W., Liu, M.: Neural slam: Learning to explore with external memory. arXiv preprint arXiv:1706.09520 (2017)
29. Zhu, Y., Mottaghi, R., Kolve, E., Lim, J.J., Gupta, A., Fei-Fei, L., Farhadi, A.: Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: Int. Conf. on Robotics & Automation, pp. 3357–3364 (2017)

C

Active Particle Filter Networks: Efficient Active Localization in Continuous Action Spaces and Large Maps

Daniel Honerkamp, Suresh Guttikonda, and Abhinav Valada

***Abstract*— Accurate localization is a critical requirement for most robotic tasks. The main body of existing work is focused on passive localization in which the motions of the robot are assumed given, abstracting from their influence on sampling informative observations. While recent work has shown the benefits of learning motions to disambiguate the robot's poses, these methods are restricted to granular discrete actions and directly depend on the size of the global map. We propose Active Particle Filter Networks (APFN), an approach that only relies on local information for both the likelihood evaluation as well as the decision making. To do so, we couple differentiable particle filters with a reinforcement learning agent that attends to the most relevant parts of the map. The resulting approach inherits the computational benefits of particle filters and can directly act in continuous action spaces while remaining fully differentiable and thereby end-to-end optimizable as well as agnostic to the input modality. We demonstrate the benefits of our approach with extensive experiments in photorealistic indoor environments built from real-world 3D scanned apartments. Videos and code are available at `http://apfn.cs.uni-freiburg.de`.**

I. Introduction

The ability of a robot to accurately localize itself is a core requirement across almost all robotic tasks from navigation [1], [2] to mobile manipulation [3], [4], [5]. Accordingly, a broad body of research has been devoted to this topic. The by far most common approach is to first define an initial guess of the robot's pose, then manually move the robot until the localization algorithm has roughly converged and continue to constantly localize the robot while it executes its tasks. This is known as passive, local localization.

Most localization algorithms rely on a form of feature matching between the current observations and a given (2D) map of the environment. As such their performance strongly depends on the current observations, which in turn are decided by the robot's motions which decide what parts of the map will be observed. But the ability to sample informative observations has remained largely unexplored. In this work, we investigate the benefits of *active localization*, in which the robot can actively seek observations that are most informative of its current pose in the environment. Furthermore, the agent can counteract the strengths and weaknesses of particular localization modules by actively avoiding ambiguous situations and failure modes of the localization module.

Previous work has extended Adaptive Markov Localization to active control by greedily maximizing information theoretic quantities [6], [7], but for the most part, remained restricted to analytical observation models and structured observations. More recently, learning-based methods have shown the benefits of active decision making for localization [8], [9], though have remained constrained to simple environments [8] or discrete actions and small maps [9], having to process the global map at every possible orientation of the agent at each step.

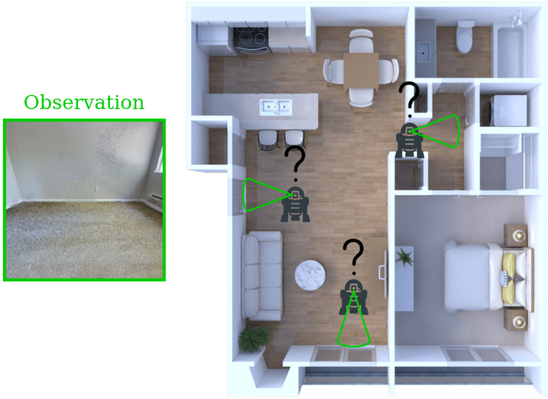


Fig. 1: To localize itself, a robot has to match its observations with a given map of the environment. To distinguish ambiguous poses requires to find observations that maximally disambiguate among the true pose given the robot's belief over its current pose. To do so, certain trajectories through the apartment reveal clearly more information than others. We propose Active Particle Filter Networks which combines learned particle filters with active decision making to sample the most informative observations.

We present an approach that couples probabilistic and learning-based methods through learned particle filters [10] and deep reinforcement learning (RL) to generalize to continuous action spaces and arbitrary sensor modalities independent of map size. Particle filters [11] enable efficient representation of multi-modal beliefs over large maps. These mechanisms can be made fully differentiable [10], [12], enabling us to learn the components of a particle filter end-to-end, thereby extending it to abstract observations such as pixels or depth maps. Importantly, these networks only need to process local information for each particle. We then train a reinforcement learning agent that selects actions to minimize the overall localization error, following the same principle of processing only local information over the most likely hypotheses through a hard attention mechanism. In contrast to previous work, this enables us to process hypotheses over continuous poses [8], [9] while at the same time breaking the dependency on processing the full map with a neural network.

Department of Computer Science, University of Freiburg, Germany. This work was funded by the European Union's Horizon 2020 research and innovation program under grant agreement No 871449-OpenDR.

We evaluate our approach in extensive photorealistic scenes of real-world 3D scanned apartments from the gibson dataset [13] in the iGibson simulator [14] and find substantial improvements in localization error over the baselines, demonstrating the benefits of the learned policy.

To summarize, this work makes the following main contributions:

- We leverage the combination of probabilistic principles with learned methods to achieve a very flexible and fully differentiable approach for active localization which does not depend on a specific sensor modality.
- We break the dependency of learning-based approaches on action granularity and map size, enabling the approach to work in continuous action spaces and arbitrary map sizes.
- We demonstrate the benefits of this approach in large photo-realistic indoor environments built from real-world 3D scanned apartments.
- We make the code publicly available at `http://apfn.cs.uni-freiburg.de`.

II. Related Work

Localization is a well studied field with a long-standing history. In the following, we discuss both passive and active localization methods which have been tackled using classical and learning-based techniques.

Passive Localization: A large number of established localization approaches rely on Bayesian filtering-based techniques. These include methods based on Kalman filters [15] which are restricted to modeling unimodal (Gaussian) beliefs, Multi-Hypothesis Kalman filters that use mixtures of Gaussians [16] and non-parametric particle filters which can model arbitrary distributions. Particle filters are widely used in methods such as Monte Carlo Localization and Adaptive Monte Carlo Localization (AMCL) [11]. Though these methods usually rely on structured observations and analytic observation models and therefore are most commonly used with LiDAR observations. While there are approaches that incorporate depth or camera images [17], [18], constructing observation models for them is extremely challenging. Recently, fully differentiable versions of particle filters have been introduced [10], [12]. These fully differentiable versions enable the use of arbitrary modalities through end-to-end optimization. LASER extends MCL with learned circular features and rendering in latent space [19]. Learning-based methods have also been proposed to extract explicit features such as room layout edges [20] or to estimate odometry directly from visual inputs [21], [22].

Active Localization: Active localization has received comparably little attention in the past. Active versions of both Markov Localization [6], [7] and Kalman filters [23] have been proposed. These methods inherit the need for structured observations or expert-specified observation models and as such cannot easily incorporate contextual clues or high-dimensional observations. Their objectives are to maximize information theoretic quantities such as the reduction in entropy of the belief. Chaplot *et al.* [8] introduce a learnable Bayesian filtering approach in combination with reinforcement learning. While they are able to learn good active policies, the model relies on access to observations from across the environment to compute features ahead of time and at each step has to process the full map for every possible discrete orientation. As a consequence, the approach does not easily generalize to different map sizes at test time and does not scale well to large maps or continuous actions. Gottipati *et al.* [9] introduce a hierarchical likelihood model in which the full map only has to be processed at a coarse resolution and only likely areas are processed at higher resolutions. Nonetheless, the dependency on the map size remains and only discrete actions can be evaluated. For both approaches, the dimensionality of the reinforcement learning agent's inputs scales linearly with the discretization of the rotation actions. In contrast, our approach never has to process the full map with a neural network and can directly evaluate continuous poses and actions.

Active SLAM: In simultaneous localization and mapping both information gain objectives [24], [25] and reinforcement learning based approaches [26], [27] have been used to control a robot while performing both the mapping and localization. In contrast to active localization, these approaches do not have access to the prior knowledge of the map to steer the robot towards informative states [28].

III. Active Particle Filter Networks

We propose Active Particle Filter Networks (APFN) consisting of two modules: a learned particle filter network maintaining a distribution over the current belief of the robot's pose and a reinforcement learning agent taking decisions based on the current observations and belief. An overview of our approach is depicted in Figure 2.

In the following, we will define the active localization task that we aim to solve and then introduce our approach.

A. Problem Statement

We assume a mobile robot that receives exteroceptive sensor readings $sens_t$ and proprioceptive odometry measurements m_t, placed randomly in an environment. Given a map M of the environment, we seek the sequence of actions $a_{1:T}$ that minimizes the pose error of the robot over a fixed time horizon T. We may be given an initial guess of the initial pose of the robot (local localization) or have to start from a uniform belief over the full map (global localization).

B. Localization Module

The robot starts with an initial belief b_0, either uniformly distributed over the map or based on an initial guess. Given the current observation $o_t = [sens_t, m_t]$, we then use a differentiable particle filter network (PF-net) [10] to update the current belief over the robot's pose. PF-net uses neural networks to present the observation and transition model of a

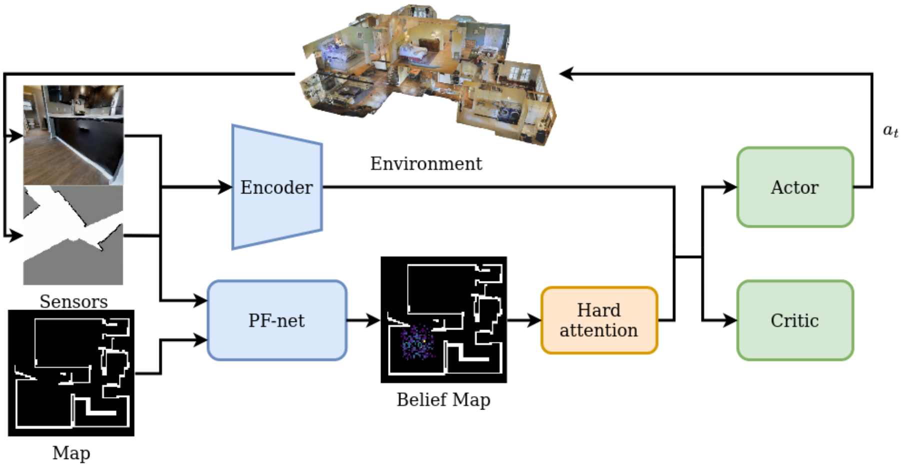


Fig. 2: Overview of the proposed Active Particle Filter Networks. Given the robot's observations, the PF-net updates the belief over the current pose of the robot, modeled as a particle distribution. This distribution is then projected into a belief map over the environment. The RL agent attends to the local regions across the most likely hypotheses as well as the raw robot observations and produces actions a_t to move the robot's base, which then result in the next sensory observations for the PF-net.

particle filter. By using a soft-resampling, where new particle weights $w_t^{\prime k}$ of K particles are sampled from a distribution

$$q(k) = \alpha w_t^k + \frac{(1-\alpha)}{K} \tag{1}$$

the gradients are non-zero for values of $\alpha \neq 1$, enabling us to optimize through the whole network. The observation model calculates the likelihood f_t^k of a particle based on an encoding of the current sensor readings and particle-centric local map which is extracted from the global map through a differentiable spatial transformer module [29]. As a result, the likelihood of each particle can be evaluated based on local information without the need to process the full global map.

This provides a number of advantages for active localization: (i) the network is fully differentiable and thereby can be jointly optimized with deep reinforcement learning algorithms, (ii) it is flexible to arbitrary robot sensors, making it applicable to a wide range of robotic platforms and (iii) it can handle continuous actions and arbitrary map sizes.

The model is trained end-to-end to minimize the mean squared pose error

$$\mathcal{L}_{pfnet} = \sum_t (\hat{x}_t - x_t^*)^2 + (\hat{y}_t - y_t^*)^2 + \beta(\hat{\phi}_t - \phi_t^*)^2, \tag{2}$$

where $\hat{x}, \hat{y}, \hat{\phi}$ and x^*, y^*, ϕ^* are estimated and ground-truth pose of the robot and β is weighting term. We follow the architecture of the original work [10] which uses convolutional encoders for both the raw observations and the local maps, then process the concatenated features with a stack of locally fully-connected and fully-connected layers.

C. Active Localization

We aim to learn a policy to move the agent such that, given the current belief about the robot's pose and the localization module, it can best disambiguate the true pose. The agent is operating in a Partially Observable Markov Decision Process (POMDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}(s'|s,a), P(o|s), r(s,a))$ where $\mathcal{S}, \mathcal{O}$ and $\mathcal{A}$ are the state, observation and action spaces, $\mathcal{T}$ and P describe the transition and observation probabilities, and r and γ are the reward and discount factor. The agent's objective is to learn a policy $\pi(a|\cdot)$ that maximises the expected return $\mathbb{E}_\pi[\sum_{t=1}^T \gamma^t r(s_t, a_t)]$.

Belief Representation: As the ground truth robot pose is not directly observable, the agent has to act based on its current belief over the state. The PF-net provides us with a multi-modal belief b_t over the global map, represented by the particle state. We transform this into a spatial, permutation invariant representation by projecting the particles into a belief map of dimension $H \times W \times 4$ where H and W are the height and width of the global map and the first channel is the occupancy map, the second channel is the aggregated weights for all particles in a given cell and the third and fourth channel are the weighted sine and cosine of all particles in a given cell. The sine and cosine are used to circumvent the non-linearity in the angles.

Agent: We propose a reinforcement learning agent that observes both its current belief together with the low-level robot observations and learns a policy $\pi(a_t|b_t, o_t; l)$ where l is the localization module. This allows it to improve the localization in two ways: (i) actively sample the most informative sensor readings and (ii) take into account the localization module's strengths and weaknesses, e.g. avoid observations where the localization module does not perform well.

While the belief stretches the full map, we find that within very few steps the particles concentrate on a small number of most likely regions. As such we apply the principle of local information to break the dependency on the full map. To do so, we extract local maps around the modes of the particle distribution from the belief map. The agent then

Parameter	PF-net	RL
Train steps	400,000	1,000,000
batch size	8	256
lr	2.5e−3	3e−4
resample	false	true
α	–	0.5
β	0.36	0.36
T	25	50
particles	30	500
initial distribution	tracking	semi-global
initial std translation	0.3	0.3
initial std angular	$\pi/6$	$\pi/6$
transition noise translation	0.0	0.01
transition noise angular	0.0	$\pi/36$
control frequency	1.7Hz	1.7Hz
τ	–	0.005
γ	–	0.99
replay buffer size	–	50,000
entropy coefficient	–	learned
$\lambda_{collision}$	–	0.1

TABLE I: Training hyperparameters for the PF-net (left) and the reinforcement learning agent (right).

observes a stack of k local belief maps, each centered and oriented according to a mode of the distribution. This is akin to a hard attention mechanism, which can be made fully differentiable if desired [30]. In practice, we find that just using the mean position and orientation of the particles works well, but extending this to cover the top k modes is straightforward. In contrast to previous work, this allows us to process and generalize to arbitrary map sizes and arbitrary continuous poses and actions.

The agent is trained to directly minimize the prediction error of the localization network. At each step, it receives a reward

$$r = -\mathcal{L}_{pfnet} - \lambda_{collision} * \mathbb{1}_{collision}, \tag{3}$$

where $\mathcal{L}_{pfnet}$ is the prediction loss of the PF-net, $\mathbb{1}_{collision}$ is a binary collision indicator and $\lambda_{collision}$ is a weighting constant. The agent has a fixed number of environment steps to localize itself, after which the episode terminates.

Training: While the approach is fully differentiable and can be optimized end-to-end, we find it beneficial to pretrain the localization network for better stability. Though joint finetuning may be able to further improve results. For pretraining we use a goal-reaching agent (see Section IV-A for details) to collect a dataset of 4,000 episodes of length 25 and then perform supervised training following Karkus *et al.* [10], using a tracking task with only 30 particles. We train the RL agent with soft-actor critic (SAC) [31], which has been shown to produce strong policies in continuous control and robotics tasks. Hyperparameters for all components are reported in Table I.

IV. Experimental Results

To evaluate the effectiveness of our approach, we perform extensive experiments in photorealistic indoor environment. With these experiments we aim to answer the following questions:

- Does the PF-net localization module scale to complex, photorealistic indoor environments and generalize to unseen apartments in these settings?
- Does our proposed approach learn to localize itself in both seen and unseen apartments and across different tasks from local to global localization?
- Is the learned policy able to find trajectories that achieve better localization than alternative control policies?

A. Experiment Setup

To evaluate our approach, we train a LoCoBot robot in the photorealistic iGibson simulator [14]. The LoCoBot robot has a differential drive and is equipped with an RGB-D camera with a field-of-view of 90° and a maximum depth of $10\,\mathrm{m}$ as well as a LiDAR with a range of 240°. The action space consists of the linear and angular velocities for the base. We use a subset of 45 apartment scenes from the gibson dataset [13], split into 38 training and 7 unseen test apartments. The test apartments are completely unseen by both the PF-net and the RL agent.

Baselines: To evaluate the policy of the reinforcement learning agent, we compare our approach against the following baselines:

- *Avoid:* A simple heuristic policy that drives forward until its depth camera recognizes a close object. We divide the depth image into four horizontal quarters and, depending on in which quarter of the depth image the close object is, drive backwards or turn away from the obstacle.
- *Goalnav:* A policy that navigates towards a random target in the environment. It uses a path-planner based on access to the ground truth traversability map and robot pose to reach this goal.
- *Turn:* An agent that always turns in place at maximum angular velocity.

Tasks: We focus on three localization tasks, ranging from local to global localization. These are

- *Tracking:* the initial particles are sampled from a multivariate Gaussian distribution with a standard deviation of $0.3\,\mathrm{m}$ and 30° and centered at a random pose sampled with the same standard deviations around the ground truth robot pose. The PF-net uses 300 particles.
- *Semi-global localization:* We uniformly sample 500 particles from a box of $3.3{\times}3.3\,\mathrm{m}$ around the initial guess.
- *Global localization:* We sample 3,000 particles uniformly across the traversable area of the whole map.

Metrics: We report the root mean squared positional error in centimeters and root mean squared angular error in radians, referred to as *position* and *orient* in the tables. All metrics are averaged over 50 episodes.

Task	seen						unseen					
	Tracking		Semi-Global		Global		Tracking		Semi-Global		Global	
Modality	position	orient	position	orient	position	orient	position	orient	position	orient	position	orient
LiDAR	20.8	0.13	27.2	0.21	111.4	0.33	18.9	0.12	23.7	0.16	141.2	0.38
RGB-D	24.8	0.15	30.2	0.20	126.6	0.30	24.5	0.16	29.2	0.18	144.1	0.34

TABLE II: Passive localization results on the iGibson dataset for different localization tasks. We report the average root mean squared positional error in centimeter (*position*) and the root mean squared orientation error of the robot's yaw in radians (*orient*). Evaluated with the pretraining settings for $T = 25$ timesteps.

Task	seen						unseen					
	Tracking		Semi-Global		Global		Tracking		Semi-Global		Global	
Agent	position	orient	position	orient	position	orient	position	orient	position	orient	position	orient
Goalnav	16.8	0.12	18.2	0.12	99.3	0.24	14.9	0.11	21.4	0.15	113.3	0.21
Avoid	15.8	0.13	22.4	0.15	152.0	0.32	15.8	0.12	33.5	0.19	162.9	0.29
Turn	**11.8**	0.80	14.6	0.09	103.1	0.30	13.9	0.10	19.8	0.12	115.9	0.31
APFN (ours)	13.4	**0.10**	**11.7**	**0.08**	**74.8**	**0.16**	**11.1**	**0.08**	**16.3**	**0.11**	**63.3**	**0.17**

TABLE III: Active localization results in the iGibson simulator in seen and unseen apartments. The localization module is based on LiDAR occupancy maps. We report the average root mean squared positional error in centimeter (*position*) and the root mean squared orientation error of the robot's yaw in radians (*orient*).

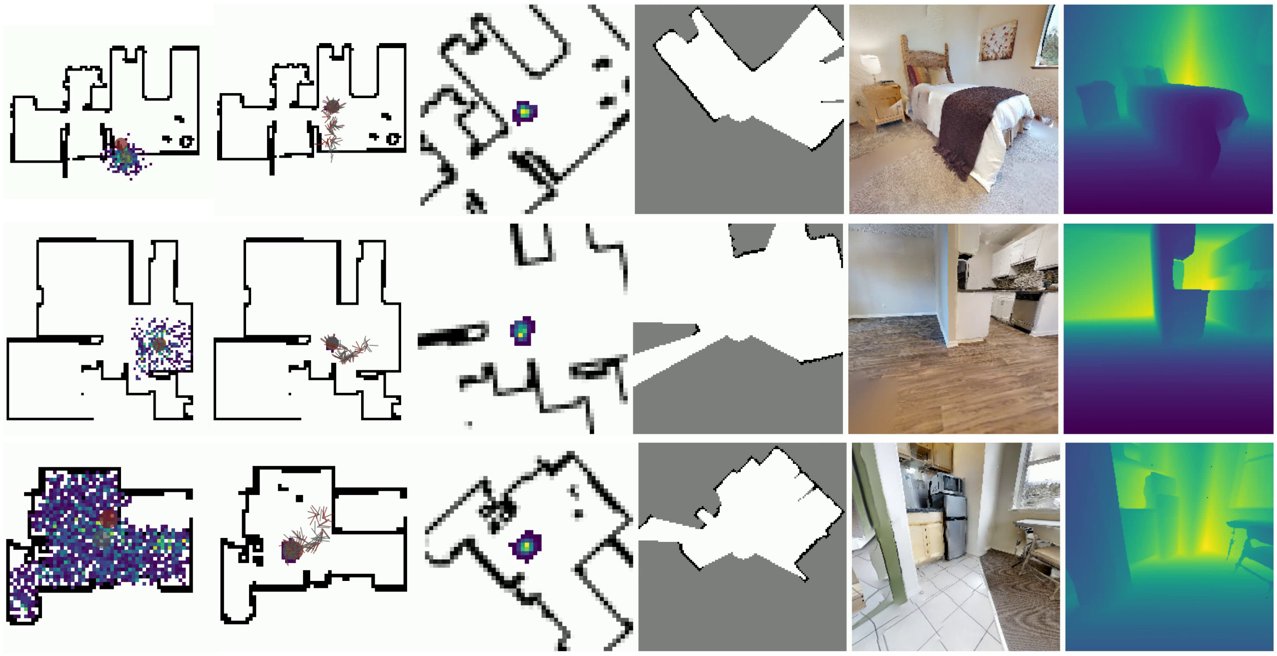


Fig. 3: Examples for the tracking (top), semi-global (mid) and global (bottom) localization tasks. Left: the initial particle distribution, second from left: the global map and trajectory of the agent. Green arrows denote the estimated poses and red the ground truth poses at each step. Circles denote the final estimated and ground-truth poses. Third from left to right: the local belief map observed by the RL agent, the current observations: occupancy grid, RGB and depth.

B. Passive Localization

The original PF-net model has focused on evaluation in the simpler, static House3D dataset [32]. We implement a version of this model based on the author's code for the iGibson simulator and evaluate it on scenes from the photorealistic gibson dataset, which are based on real-world 3D scans of apartments. We report the results for passive localization for different modalities based on the goalnav agent that collected the training data in Table II. LiDAR scans are converted to occupancy maps in which 0 is free space, 1 unexplored, and 2 occupied.

We find that the PF-net performs well in these more complex scenes, achieving a positional error of around 20-25 cm for tracking, which, for both modalities, is actually lower than the $40 - 49\,\mathrm{cm}$ error reported on the House3D dataset [10]. Moreover, the network generalizes well to unseen apartments, showing no significant generalization gap. Even though the field-of-view of the RGB-D camera is much smaller than what the LiDAR can sense, both modalities achieve relatively similar performance, highlighting that the network is able to extract rich information in the complex pixel observations.

C. Active Localization

For active localization, we focus on the best performing LiDAR modality. The RL agent observes the robot state consisting of current forward and angular velocities, a collision flag, and the remaining steps in the episode together with the occupancy map from the LiDAR and the local belief map. To ensure it can learn full obstacle avoidance, it also receives the RGB-D observations. The policy consists of a shared feature encoder, made up of three convolutional networks, one for RGB-D and one for the occupancy map. Each network consists of layers with (channels, kernel size, stride) of [(32, (3, 3), 2), (64, (3, 3), 2), (64, (3, 3), 1), (64, (2, 2), 1)]. These features are then concatenated with the robot state and passed to the actor and critic, consisting of a two-layer MLP with 512 neurons. All intermediate layers are followed by ReLU activations. All pixel-based observations are of size 56×56. While we train the PF-net on ground truth odometry data, during the policy training we add zero-centered Gaussian noise with standard deviation of $1\,\mathrm{cm}$ and $5°$ to the transitions. We train the policy with 500 particles and at test time evaluate with varying numbers of particles as defined for the different tasks.

Table III reports the results for the active localization tasks for both seen and unseen apartments. First of all, we find large differences in localization performance across the different motion models. This highlights the strong dependence on the robot's movements and confirms the importance of active decision making for globalization. We find that the reinforcement learning agent consistently achieves the best localization across all tasks. The only exception is the positional error in the tracking task, in which the turn policy achieves a very low positional error, but suffers from a large angular error. Note that in this task the initial particle distribution is already fairly accurate, as such it may actually be beneficial to remain in place. Moreover, the agent successfully generalizes to unseen apartments. Note that these apartments have not been seen by both the RL agent and the localization module. To succeed in these apartments, the agent has to learn general movement patterns and the ability to seek out informative regions. Lastly, we find that differences in localization performance are particularly large in the global localization task with our approach reducing the positional error by over 60% in comparison to other motions. This is expected as in global localization we have the least amount of prior information about the robot's pose.

Qualitatively we find that the reinforcement agent performs targeted movements through the room with frequent rotations which reveals a large area of the apartments and is aligned with the strong performance of the turn baseline. Examples of the agent's trajectories and inputs are shown in Figure 3 and in the accompanying video.

V. Conclusion

In this work, we introduce Active Particle Filter Networks which combines probabilistic filtering methods with learned decision making to accurately localize a robot in realistic indoor environments. In contrast to previous methods, our approach scales to continuous action spaces and arbitrary map sizes by selectively attending to only local information. In extensive experiments, we evaluate this ability in photorealistic indoor environments and find that it is able to accurately localize itself in both seen and completely unseen apartments. The learned policy considerably outperforms the baselines, demonstrating strong improvements in localization performance by sampling informative observations.

In future work, we aim to extend the approach to simultaneously control sensors such as actuated cameras, which promises to benefit even more from active perception. Another promising avenue is the extension of learning-based localization and attention mechanisms to dynamic environments and noisy, partial or incorrect maps in which it becomes important to selectively filter out uncertain or incorrect observations. Lastly, the trade-off between active localization and other task objectives is an exciting direction for future work.

References

[1] J. V. Hurtado, L. Londoño, and A. Valada, "From learning to relearning: A framework for diminishing bias in social robot navigation," *Frontiers in Robotics and AI*, vol. 8, p. 650325, 2021.

[2] A. Younes, D. Honerkamp, T. Welschehold, and A. Valada, "Catch me if you hear me: Audio-visual navigation in complex unmapped environments with moving sounds," *arXiv preprint arXiv:2111.14843*, 2021.

[3] G. N. DeSouza and A. C. Kak, "Vision for mobile robot navigation: A survey," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 237–267, 2002.

[4] D. Honerkamp, T. Welschehold, and A. Valada, "$\mathrm{N}^2\mathrm{m}^2$: Learning navigation for arbitrary mobile manipulation motions in unseen and dynamic environments," *arXiv preprint arXiv:2206.08737*, 2022.

[5] ——, "Learning kinematic feasibility for mobile manipulation through deep reinforcement learning," *IEEE Robotics and Automation Letters*, 2021.

[6] D. Fox, W. Burgard, and S. Thrun, "Active markov localization for mobile robots," *Robotics and Autonomous Systems*, vol. 25, no. 3-4, pp. 195–207, 1998.

[7] ——, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, vol. 11, pp. 391–427, 1999.

[8] D. S. Chaplot, E. Parisotto, and R. Salakhutdinov, "Active neural localization," *arXiv preprint arXiv:1801.08214*, 2018.

[9] S. K. Gottipati, K. Seo, D. Bhatt, V. Mai, K. Murthy, and L. Paull, "Deep active localization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4394–4401, 2019.

[10] P. Karkus, D. Hsu, and W. S. Lee, "Particle filter networks with application to visual localization," in *Proc. of the Conference on Robot Learning*, 2018, pp. 169–178.

[11] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001.

[12] R. Jonschkowski, D. Rastogi, and O. Brock, "Differentiable particle filters: End-to-end learning with algorithmic priors," in *Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

[13] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 9068–9079.

[14] C. Li, F. Xia, R. Martín-Martín, M. Lingelbach, S. Srivastava, B. Shen, K. E. Vainio, C. Gokmen, G. Dharan, T. Jain, A. Kurenkov, C. K. Liu, H. Gweon, J. Wu, L. Fei-Fei, and S. Savarese, "igibson 2.0: Object-centric simulation for robot learning of everyday household tasks," in *2021 Conference on Robot Learning (CoRL)*, ser. Proceedings of Machine Learning Research, vol. 164, 2021, pp. 455–465.

[15] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous robot vehicles*, 1990, pp. 167–193.

[16] I. J. Cox and J. J. Leonard, "Modeling a dynamic environment using a bayesian multiple hypothesis approach," *Artificial Intelligence*, vol. 66, no. 2, pp. 311–344, 1994.

[17] F. Dellaert, W. Burgard, D. Fox, and S. Thrun, "Using the condensation algorithm for robust, vision-based mobile robot localization," in *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*, vol. 2, 1999, pp. 588–594.

[18] B. Coltin and M. Veloso, "Multi-observation sensor resetting localization with ambiguous landmarks," *Autonomous Robots*, vol. 35, no. 2, pp. 221–237, 2013.

[19] Z. Min, N. Khosravan, Z. Bessinger, M. Narayana, S. B. Kang, E. Dunn, and I. Boyadzhiev, "Laser: Latent space rendering for 2d visual localization," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2022, pp. 11 122–11 131.

[20] F. Boniardi, A. Valada, R. Mohan, T. Caselitz, and W. Burgard, "Robot localization in floor plans using a room layout edge extraction network," in *Int. Conf. on Intelligent Robots and Systems*, 2019, pp. 5291–5297.

[21] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, "Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem," in *Proc. of the National Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[22] N. Vödisch, D. Cattaneo, W. Burgard, and A. Valada, "Continual slam: Beyond lifelong simultaneous localization and mapping through continual learning," *arXiv preprint arXiv:2203.01578*, 2022.

[23] P. Jensfelt and S. Kristensen, "Active global localization for a mobile robot using multiple hypothesis tracking," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 5, pp. 748–760, 2001.

[24] H. J. S. Feder, J. J. Leonard, and C. M. Smith, "Adaptive mobile robot navigation and mapping," *The International Journal of Robotics Research*, vol. 18, no. 7, pp. 650–668, 1999.

[25] C. Leung, S. Huang, and G. Dissanayake, "Active slam using model predictive control and attractor based exploration," in *Int. Conf. on Intelligent Robots and Systems*, 2006, pp. 5026–5031.

[26] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural slam," in *International Conference on Learning Representations (ICLR)*, 2020.

[27] J. A. Placed and J. A. Castellanos, "A deep reinforcement learning approach for active slam," *Applied Sciences*, vol. 10, no. 23, p. 8386, 2020.

[28] J. A. Placed, J. Strader, H. Carrillo, N. Atanasov, V. Indelman, L. Carlone, and J. A. Castellanos, "A survey on active simultaneous localization and mapping: State of the art and new frontiers," *arXiv preprint arXiv:2207.00254*, 2022.

[29] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," *Advances in neural information processing systems*, vol. 28, 2015.

[30] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Int. Conf. on Machine Learning*, 2015, pp. 2048–2057.

[31] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Int. Conf. on Machine Learning*, vol. 80, 2018, pp. 1861–1870.

[32] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian, "Building generalizable agents with a realistic and rich 3d environment," *arXiv preprint arXiv:1801.02209*, 2018.

D

robotics

MDPI

Article

Sim-to-Real Deep Reinforcement Learning for Safe End-to-End Planning of Aerial Robots

Halil Ibrahim Ugurlu , Xuan Huy Pham and Erdal Kayacan *

Artificial Intelligence in Robotics Laboratory (AiR Lab), The Department of Electrical and Computer Engineering, Aarhus University, 8000 Aarhus C, Denmark
* Correspondence: erdal@ece.au.dk

Abstract: In this study, a novel end-to-end path planning algorithm based on deep reinforcement learning is proposed for aerial robots deployed in dense environments. The learning agent finds an obstacle-free way around the provided rough, global path by only depending on the observations from a forward-facing depth camera. A novel deep reinforcement learning framework is proposed to train the end-to-end policy with the capability of safely avoiding obstacles. The Webots open-source robot simulator is utilized for training the policy, introducing highly randomized environmental configurations for better generalization. The training is performed without dynamics calculations through randomized position updates to minimize the amount of data processed. The trained policy is first comprehensively evaluated in simulations involving physical dynamics and software-in-the-loop flight control. The proposed method is proven to have a 38% and 50% higher success rate compared to both deep reinforcement learning-based and artificial potential field-based baselines, respectively. The generalization capability of the method is verified in simulation-to-real transfer without further training. Real-time experiments are conducted with several trials in two different scenarios, showing a 50% higher success rate of the proposed method compared to the deep reinforcement learning-based baseline.

Keywords: deep reinforcement learning; obstacle avoidance; quadrotors; sim-to-real transfer

check for updates

Citation: Ugurlu, H.I.; Pham, X.H.; Kayacan, E. Sim-to-Real Deep Reinforcement Learning for Safe End-to-End Planning of Aerial Robots. *Robotics* **2022**, *11*, 109. https://doi.org/10.3390/robotics11050109

Academic Editors: Xiaowei Huang, Wenjie Ruan and Xingyu Zhao

Received: 30 August 2022
Accepted: 10 October 2022
Published: 13 October 2022

1. Introduction

Autonomous aerial robots are increasingly deployed in applications that require safe path planning in dense environments, such as a greenhouse covered with dense plants, search and rescue operation in an unstructured collapsed building or navigation in a forest. Traditionally, autonomous navigation is solved under separate problems such as state estimation, perception, planning, and control [1]. This approach may lead to higher latency combining individual blocks and system integration issues. On the other hand, recent developments in machine learning, particularly in reinforcement learning (RL) and deep reinforcement learning (DRL), enable an agent to learn various navigation tasks end-to-end with only a single neural network policy that generates required robot actions directly from sensory input. These methods are promising for solving navigation problems faster computationally since they do not deal with the integration of subsystems that are tuned for their particular goals.

This study attempts to address the end-to-end planning problem of a quadrotor UAV in dense indoor environments. The quadrotor deployed with a depth camera is required to find its way around the global trajectory. We propose a DRL-based safe navigation methodology for quadrotor flight. The learned DRL policy, utilizing the depth images and the knowledge of a global trajectory, generates safe waypoints for the quadrotor. We develop a Webots-based simulation environment where the DRL agent is trained with obstacle tracks where the obstacle locations, shapes, and sparsity are randomized for every episode of policy training for better generalization. Furthermore, we introduce safety boundaries to be considered during training in addition to collision checks. The safety boundaries enable the agent to prevent risky situations that make the method more robust to uncertainties.

Contributions

The contributions of this paper are fourfold:

- A novel DRL simulation framework is proposed for training an end-to-end planner for a quadrotor flight, including a faster training strategy using non-dynamic state updates and highly randomized simulation environments.
- The impact of continuous/discrete actions and proposed safety boundaries in RL training are investigated.
- We open-source the Webots-based DRL framework, including all training and evaluation scripts (the code, trained models, and simulation environment will be available at https://github.com/open-airlab/gym-depth-planning, accessed on 1 July 2022).
- The method is evaluated with extensive experiments in Webots-based simulation environments and multiple real-world scenarios, transferring the network from simulation to real without further training.

The remainder of this paper is organized as follows. Section 2 reviews the related literature. Section 3 explains the end-to-end planning methodology for a quadrotor UAV with the formalization of the RL problem. Section 4 provides the experimental setup and the comprehensive tests of the proposed method in the simulation environment. The section also provides the results of the real-time tests. Finally, Section 5 concludes this work with future research directions.

2. Related Work

As a machine learning paradigm, RL aims to solve sequential decision-making problems through the interaction of a learning agent with its environment [2]. With the success of the deep learning models in machine learning, it is also applied to RL, which brings about the DRL field with success in several benchmark problems such as video games [3] or continuous control tasks [4]. Several methods are proposed to optimize deep neural networks to learn the value function [3], policy function [5], or both [4,6] in the RL domain, such as the proximal policy optimization (PPO) [7] algorithm, a state-of-the-art method utilized in this work. RL and its successor DRL have gained attention in robotics applications as it is encouraging a complete framework for intelligent robots to learn by interacting with their environment.

Since deep learning-based methods require plenty of data, they have emphasized using simulation data as an alternative to expensive real-world data. The usefulness of simulations becomes more crucial for DRL considering potential hardware failures during exploration in the real-world [8]. However, there is a gap between simulation and real-world data, as sensor signal qualities may not be preserved due to the lack of realistic noise. Earlier works have shown that certain data modalities provide a better abstraction for sim-to-real transfer, such as using depth images [9] or applying morphological filters [10]. Another gap between simulation and reality comes from the limitations in modeling real-world dynamics, which are generally countered by domain randomization, e.g., randomizing physical parameters [11] or randomizing observations gathered by visual sensors [12].

Deep neural network-based methods are utilized in the control and navigation of several robotics applications, including real-world demonstrations. Those applications can be classified into two categories considering the input to the neural network: the state information, such as positions and velocities, or raw sensory data, such as color or depth images. Using state information directly, neural network policies have similar functionality with a controller block in quadrotor UAVs, such as in attitude control [13] or position control [11,14] level. Furthermore, various output configurations from motion primitives [15] to lowest-level motor voltage commands [16] for the learned policies are also investigated. Compared to conventional control theoretic approaches, those methods are lacking in providing mathematical guarantees such as stability analysis [17]. However, it is an active research area where the most recent works promisingly show that DRL-based cascaded control outperforms classical proportional-integral-derivative (PID) controllers [18] and demonstrates challenging control tasks such as high-speed flight control [19].

Prior to deep learning-based methods, the planning methods for robotics have been extensively studied. In particular, graph-based (e.g., A* [20] and D* [21]), potential field-based [22], and sampling-based [23] methods can be counted as subfields of conventional planning algorithms, which require a graph or map representation of the configuration space. Conventional planning algorithms are also an active research area for the application of quadrotor flight [24], as well as other fields, such as collision avoidance of near-Earth space systems [25]. Unlike conventional planning algorithms, DRL enables the learning of so-called neural network end-to-end planners or visuomotor controllers that can generate actions directly from sensory input without any map. Although several applications for ground robots utilize lidar sensors for obstacle avoidance tasks [26,27], visual sensors are more commonly used in aerial applications such as color or depth images. End-to-end navigation is broadly investigated for quadrotor UAVs in several domains such as corridor following [28], drone racing [1,29], aerial cinematography [30], autonomous landing [31,32] or obstacle avoidance [33,34], which is the application in this paper. A recent study demonstrates the capabilities of DRL in a safety critic mission, leveraging the depth and semantic images for an emergency landing [32]. Similarly, a high-speed quadrotor flight with obstacle avoidance has been shown with an imitation learning-based neural network policy recently [35]. In the context of the present study, safe navigation is considered rather than agility. Furthermore, instead of imitation learning, DRL is studied. More similar to the present study, Camci et al. [36] utilize a quadrotor with a depth camera for obstacle avoidance but with discrete actions. Dooraki et al. [37] also propose a similar application with continuous actions in the position domain. The present research differs by proposing safety boundaries and enabling heading angle steps together with position steps.

3. End-to-End Motion Planning of UAV

3.1. Reinforcement Learning Formalization of the Environment

An RL problem is generally formalized as a Markov decision process (MDP) with state, action, and reward components with discrete timesteps, t. The common variables are shown in Appendix A. For the problem of end-to-end local planning, the state is defined as multi-modal data containing the depth image and the vector representing the moving target point,

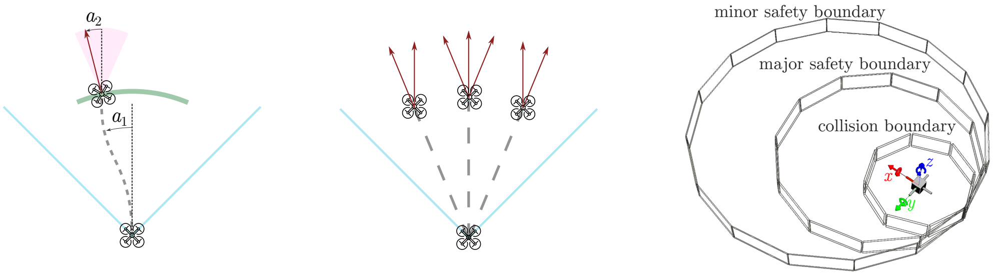
$$s_t = (I_{depth,t}, \mathbf{p}_t), \tag{1}$$

where $I_{depth,t}$ is 64 × 64 matrix representing depth image and $\mathbf{p}_t = [x_t, y_t]^T$ is 2 × 1 vector representing the position of the target point with respect to the body frame. As shown in Figure 1c, x-axis and y-axis represent the forward and the left direction, respectively.

The MDP environment is constructed for both continuous and discrete action spaces for comparison purposes. For the formation of continuous action space, a vector of length two is selected,

$$\mathbf{a}_t \in \{[a_1, a_2]^T | -\pi/8 \leq a_1, a_2 \leq \pi/8, a_1, a_2 \in \mathbb{R}\} \tag{2}$$

where a_1 defines the direction of 1 m position step and a_2 defines the rotation in yaw angle with respect to the current body frame. The distribution of actions is illustrated in Figure 1a. The boundaries of the action space are selected to match the information from the single-depth camera by assuring that all the actions are taken into a known area. On the other hand, yaw angle change enables a sharper turn around an obstacle, as well as a change in point of view if required.



(**a**) Continuous actions. (**b**) Discrete actions. (**c**) Safety boundaries.

Figure 1. The actions and the safety boundaries of the end-to-end planning agent are illustrated. (**a**) Continuous actions from the top view are defined by two angles: a_1 represents the direction of the position step (dashed lines), and a_2 represents the heading angle (red arrow). FOV of the depth camera is presented as blue lines. (**b**) Seven possible discrete actions: position and yaw angle steps. (**c**) Illustration of quadrotor body reference frame where the x-axis is the forward-looking direction and circular safety and collision boundaries considered in the simulation environment. The diameter of collision, major and minor safety boundaries are 1, 2, and 3 m, respectively.

The discrete action space, which is a subset of the aforementioned continuous action set, is comprised of seven actions, defined as a combination of a 1 m position step in three possible directions and a turn in yaw angle with respect to the drone's reference frame, as shown in Table 1. The possible actions are also illustrated in Figure 1b. Both the direction of position step and heading angle are a combination of the spatial limits of a continuous action set and forward direction while moving and opposite turning sides are neglected.

The discrete action set is mainly constructed for comparison with previous work [36], with some modifications. First, the position step direction, a_1, is kept small in order to fit with the field-of-view (FOV) of the depth camera so that the UAV does not hit an unseen object. Second, yaw angle change is enabled to match the capabilities of continuous actions. Finally, since we restrict the problem definition for constant altitude flight, we disable the actions that change altitude. We believe these updates facilitate a fair comparison between continuous and discrete action selections in such a problem domain.

Table 1. Discrete actions: each action is applied as a position step and a turn in heading angle with respect to the drone's reference frame.

Choice	Corresponding Continuous Action $[a_1, a_2]$
Action 1	$[\pi/8, \pi/8]$
Action 2	$[\pi/8, 0]$
Action 3	$[0, \pi/8]$
Action 4	$[0, 0]$
Action 5	$[0, -\pi/8]$
Action 6	$[-\pi/8, 0]$
Action 7	$[-\pi/8, -\pi/8]$

An episode begins when the UAV is at the beginning of a track defining a global trajectory of length L and obstacles placed. At each timestep, an action is applied to the UAV, then the depth image and next target point are obtained as the new state. Figure 2 illustrates the selection of the target point projected on the global path and 5 m ahead of the drone for consecutive timesteps. The episode is terminated under three conditions: crashing into an obstacle, deviating from the global trajectory, and finalizing the route.

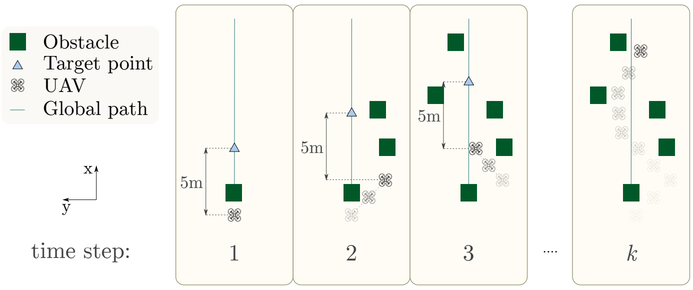


Figure 2. Moving target generation for the end-to-end agent. The target point is located 5 m ahead of the current location of the UAV projected onto the global path. The obstacles, generated target points, UAV, and global path are represented as shown in the legend. The UAV takes a position step at each timestep and is informed by the vector showing the generated target point. The overall trajectory is demonstrated at the *k*'th timestep.

A circular boundary is defined around the quadrotor with a diameter of 1 m to encounter collisions. Whenever a part of this boundary is violated by an obstacle, the collision is counted, and the episode is terminated. In addition to the collision boundary, two safety boundaries, major and minor, are defined with diameters 2 m and 3 m, centered at $[0.5, 0, 0]$ and $[1, 0, 0]$ on the quadrotor body frame. These safety boundaries are located toward the frontal area of the drone to detect risky objects in the short-term action path, as shown in Figure 1c. Unlike the collision, the violation of safety boundaries is not resulting in the termination of the episode, but it adds a negative reward to avoid being close to obstacles. Since the quadrotor motion is considered in the forward direction, these safety boundaries are chosen to be tangent with the collision boundary from the reverse direction. The diameters of major and minor safety boundaries are chosen to occupy the regions in two and three consecutive action steps, respectively.

The reward signal is based on the UAV's relative motion and the occupation of safety boundaries at every timestep if the episode is not terminated. For termination of an episode, both the collision and excessive deviation are punished with constant values. On the other hand, finishing a route without a crash is rewarded. The reward signal is defined as,

$$r_t = \begin{cases} 2\Delta x - d_y - 0.3 d_\theta - 10\mathbb{1}_{major-safety} - 2\mathbb{1}_{minor-safety}, & \text{for non-terminal steps,} \\ R_{dp}, & \text{for } d_y > 5 \text{ m,} \\ R_{cp}, & \text{for collision,} \\ R_{fr}, & \text{for finishing normally,} \end{cases} \tag{3}$$

where Δx, d_y and d_θ are the distance traveled forward, the distance to the global trajectory and the yaw angle difference from forward-looking; $R_{dp} = -10$, $R_{cp} = -20$ and $R_{fr} = 20$ are punishment for excessive deviation, punishment for collision, and reward for finishing an episode without any crash; $\mathbb{1}_{major-safety}$ and $\mathbb{1}_{minor-safety}$ are indicator functions returning one or zero when corresponding safety boundary is occupied or not. This reward logic enables the agent to learn to avoid obstacles while quickly navigating toward the goal, as well as keeping a distance from obstacles thanks to safety boundaries.

3.2. Randomization of the Environment

For every episode of training, the obstacles in the environment are randomized. The randomization strategy is summarized in Algorithm 1. The algorithm randomly creates a corridor and places obstacles with varying shapes, sizes, orientations, and locations.

Algorithm 1 Randomized obstacle environment.

```
$with\_wall \sim \mathcal{U}(\{True, False\})$
if $with\_wall$ then
    $wall\_width \sim \mathcal{U}(4, 10)$
end if
$\#obstacles \sim \mathcal{U}(2, 7)$
for each obstacle do
    $obstacle\_shape \sim \mathcal{U}(\{Box, Sphere, Cylinder\})$
    $obstacle\_position.x \sim \mathcal{U}(3, L)$
    $obstacle\_position.y \sim \mathcal{N}(0, 2.5)$
    $obstacle\_position.z \sim \mathcal{U}(2, 3)$
    $randomize\_obstacle\_orientation()$
    if $obstacle\_shape$ is cylinder then
        $radius \sim \mathcal{U}(0.5, 1.5)$
        $height \sim \mathcal{U}(1, 3)$
    end if
    if $obstacle\_shape$ is box then
        $length \sim \mathcal{U}(0.5, 2.5)$
    end if
    if $obstacle\_shape$ is sphere then
        $radius \sim \mathcal{U}(0.5, 1.5)$
    end if
end for
```

3.3. Deep Reinforcement Learning: Actor and Critic Network Architecture

The actor and critic networks trained by PPO [7] rely on the same feature extractor. PPO is a policy gradient algorithm that optimizes the parameterized policy (actor) function, $\pi_\theta(a_t|s_t)$, with parameters, θ, using the clipped objective [7],

$$J^{CLIP} = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)], \quad (4)$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ measures how different the new and old policy parameters. $\hat{A}_t$ is the advantage estimate in the given timestep, which measures how much a certain action acquires an extra long-term reward return using the parameterized value (critic) function. The optimization runs after every rollout of n_{steps} number of timesteps. Using this objective function, PPO increases the probability of good action decisions while suppressing bad decisions, similar to a previous DRL method trust region policy optimization (TRPO) [5], which only uses the expected value of $r_t(\theta)\hat{A}_t$. PPO introduces the clipped objective, which prevents large policy updates with only first-order optimization.

The actor–critic neural network feeds the depth image to three convolutional layers with the number of filters, kernel size, stride, and activation functions, as given in Figure 3. The convolutional layer is flattened and then reduced to a tensor of 256 neurons by a fully connected layer. This tensor is concatenated with the moving target input to create the feature vector that is shared by both actor and critic networks. The critic network utilizes two fully connected layers with 64 neurons each and a tangent hyperbolic (tanh) activation to regress the value function. The actor (policy) network has similar hidden layers to the critic network, but the output layer consists of n_a neurons where n_a is equal to two for continuous actions and seven for discrete actions.

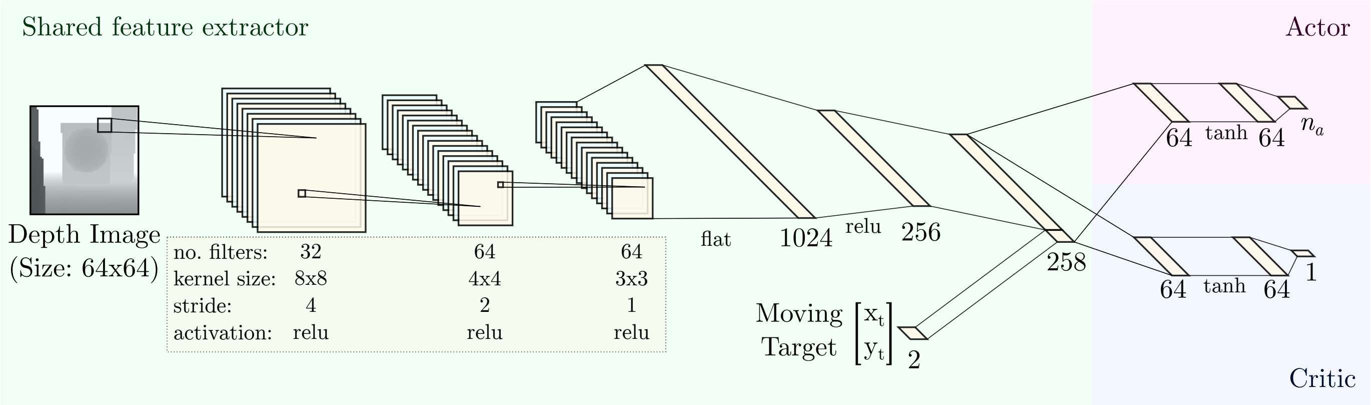


Figure 3. Actor–critic network structure of the end-to-end local planner. The actor network takes inputs as depth images through convolutional layers and the moving target through fully connected layers and outputs the next actions as position and yaw angle. The critic network shares the same feature extractor and outputs the value estimation.

4. Experiments and Results

4.1. Simulation Setup

A Webots-based simulation environment has been developed to train and test the proposed algorithms. Webots [38] is an open-source robot simulator that allows different programming interfaces, such as python, or robot operating system (ROS), for several kinds of robots. The 2021a release of Webots has been utilized to develop the cluttered environment and deploy the UAV with the required sensory equipment. A third-party software package, ArduPilot, is selected to implement a quadrotor UAV in Webots to benefit from its MAVLink extendable communication-featured stable and reliable UAV with its Webots SITL extension. The UAV robot is then equipped with a depth camera to provide the required information to carry out end-to-end planning operations. The environment needs to be reset every time a collision occurs in training, which is a benefit we can have in simulation throughout the trial-and-error process.

The simulation environment is wrapped as an OpenAI gym environment [39] to allow the required communication between the DRL algorithm and the environment. ROS [40] handles this communication between the gym wrapper and the simulation. Specifically, the MAVROS package is used to acquire the state estimation of the quadrotor UAV and send position commands. The remaining information, such as depth images and collision, is communicated directly by individual Webots ROS topics. The gym environment interfaces with the simulation environment as an MDP for the DRL algorithm, as explained in Section 3.1.

4.2. Training in Simulation

The agent is trained in Webots with randomized obstacle environments to present a variety of data for the deep network. The agent is subject to different obstacle shapes, sizes, locations, and densities for every episode of training. The randomization enables the agent to generalize the experience during RL training. Each episode begins on a randomly created route and terminates either at the end of the route or in a collision. Sample environment configurations used for evaluation purposes are shown in Figure 4.

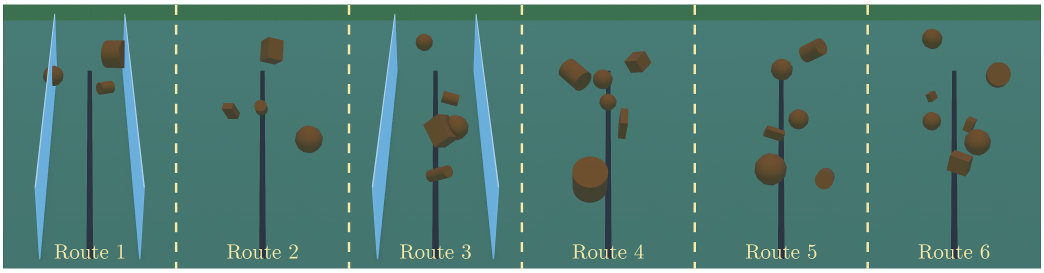


Figure 4. Six evaluation routes generated by the proposed environment randomizer. The black lines represent the projection of a 30 m global trajectory on the ground.

Since the policy network generates waypoints to travel, the robot is transported in the training simulations to acquire a new observation. This method reduces the computational burden for physical dynamics in each update step, thus fastening the overall training time. The transportation is also randomized in position and orientation in order to improve the variety of the training data as well as to address the possible poor performance of the controller in following the waypoints.

The policy network is trained with the PPO algorithm implementation in stable-baselines3 [41]. The 'number of steps to run per update' hyperparameter, n_{steps}, is set to 1024, while other hyperparameters are kept as default. The algorithm is trained through 100,000 timesteps, and the best network is stored during training based on the reward performance in the recent 20 episodes.

4.3. Simulation Results

The same randomization method is used for creating the evaluation routes. A set of six unique routes has been determined to test and compare the methods fairly, as shown in Figure 4. The routes are numbered with increasing order of the number of obstacles contained, which roughly makes the route more challenging. Furthermore, each route starts with a random offset of ±0.5 m in the horizontal positioning of the drone for evaluation purposes. Each method is evaluated ten times in each lane. The success and the distance traveled without collision are recorded for every trial. In Table 2, the success rate and average traveled distance are listed for ten trials in each route. In addition, a safety cost is measured based on the inverse distance of the objects closer than 3 m, and the average of this safety cost over all runs is reported.

The proposed method, the safe continuous depth planner (SCDP), is compared with two DRL-based versions and a potential field-based planner. The continuous depth planner (CDP) considers the same method without introducing safety boundaries. The discrete depth planner (DDP) is considered as a baseline, which is a modified version of previous research [36] using a discrete action domain, as explained in Section 3.1. An artificial potential field-based planner (APF) is also implemented as a conventional baseline method [22]. The chosen baselines represent two important classes of motion planning algorithms for quadrotors: learning-based and model-based methods.

In the implementation of APF, each pixel in the middle row of the depth image creates a repulsive force, and the moving target creates an attractive force. As such, APF uses the same observation for the end-to-end planner. The action is also selected from the continuous action set according to the direction of the common artificial force in the reference frame of the UAV. The angle of the artificial force is mapped to the action set. When the angle is above $\pi/8$ in magnitude, it also activates yaw angle turning actions. The parameters of attractive and repulsive forces are tuned on the training routes and then tested to compare with the proposed method fairly.

As can be seen from Table 2, the proposed safety boundaries demonstrate better performance than the plain case in terms of the success rate. It is also observed that the final policy avoids becoming closer to the obstacles, considering the reported safety cost, because the rewards encountered with safety boundaries help the agent avoid dangerous situations.

The same observations are valid when the continuous action set is compared against the discrete set. The continuous depth planner is significantly more capable of handling dense obstacle scenarios, such as routes #5 and #6, since it can generate finer trajectories. Lastly, the learning-based end-to-end planners perform better than the baseline artificial potential field method because of their learning capabilities to handle uncertainties.

Table 2. Average travel distance (in meters) and success rate (in percentage) of methods—safe continuous depth planner (SCDP), continuous depth planner (CDP), discrete depth planner (DDP), and artificial potential field (APF)—over 10 runs at 6 test routes.

		Route 1	**Route 2**	**Route 3**	**Route 4**	**Route 5**	**Route 6**	**Overall**	**Safety Cost**
SCDP	**distance**	30	30	30	25.8	30	30	**29.7**	0.51
	success rate	100	100	100	70	100	90	**93**	
CDP	**distance**	30	30	30	19.4	30	27.4	28.0	0.52
	success rate	100	100	100	0	100	80	80	
DDP	**distance**	30	28.8	8.93	16.6	28.0	25.8	23.1	0.57
	success rate	100	90	0	0	80	60	55	
APF	**distance**	27.5	24.7	29.0	10.7	9.7	20.7	20.4	0.87
	success rate	90	10	90	10	0	60	43	

In order to provide a qualitative comparison, sample trajectories obtained from route #6 are presented in Figure 5. In parallel with the observations in the comparison table, SCDP tries to avoid risky situations. Additionally, the generated path by SCDP is smoother, implying consistency in the sequential actions. Intuitively having a smoother trajectory reduces the controller effort to follow provided waypoints, which is another advantage of SCDP against other methods.

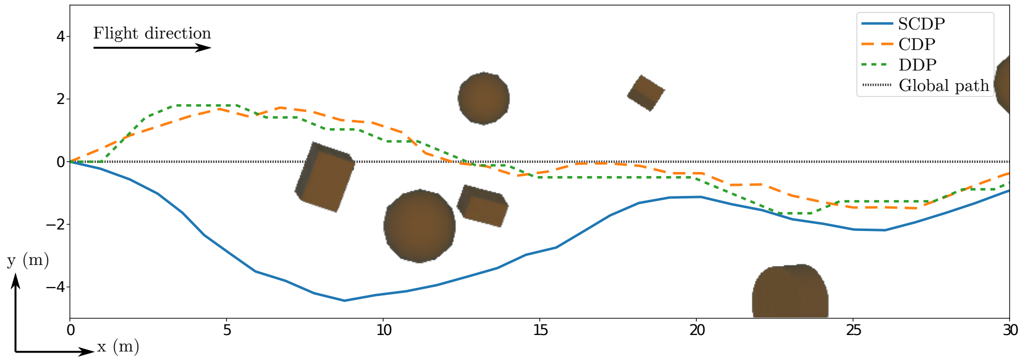


Figure 5. Comparison of the sample trajectories collected in route #6.

4.4. Real-Time Experiments

The trained model is deployed for real-time experiments in a custom quadrotor carrying an Intel Realsense D435i depth camera, as shown in Figure 6. The drone is controlled by a Pixhawk autopilot [42]. The overall framework runs entirely onboard on an NVIDIA Jetson TX2 computer, except that the robot's localization is provided by a motion capture system. The overall pipeline can run up to 8 Hz. A geometric controller [43] is used to track the poses generated by the policy accurately, with a linear speed of around 1 m/s.

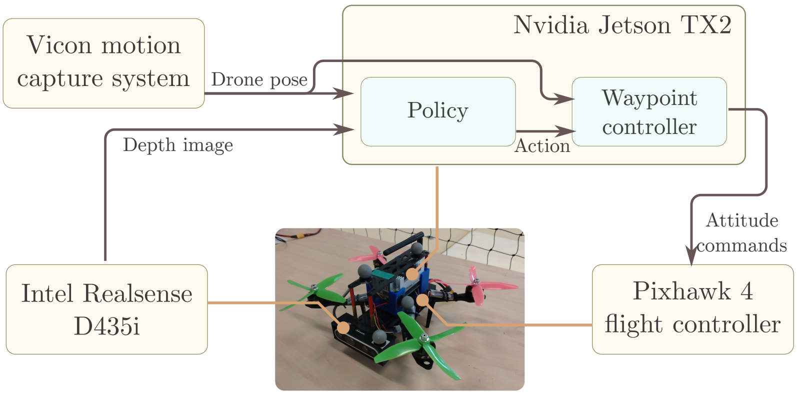


Figure 6. Custom drone used in real-time experiments. The depth images are acquired from an Intel Realsense D435i depth camera. The end-to-end planner is running onboard by an NVIDIA Jetson TX2. Pixhawk 4 flight controller is utilized for following the waypoints.

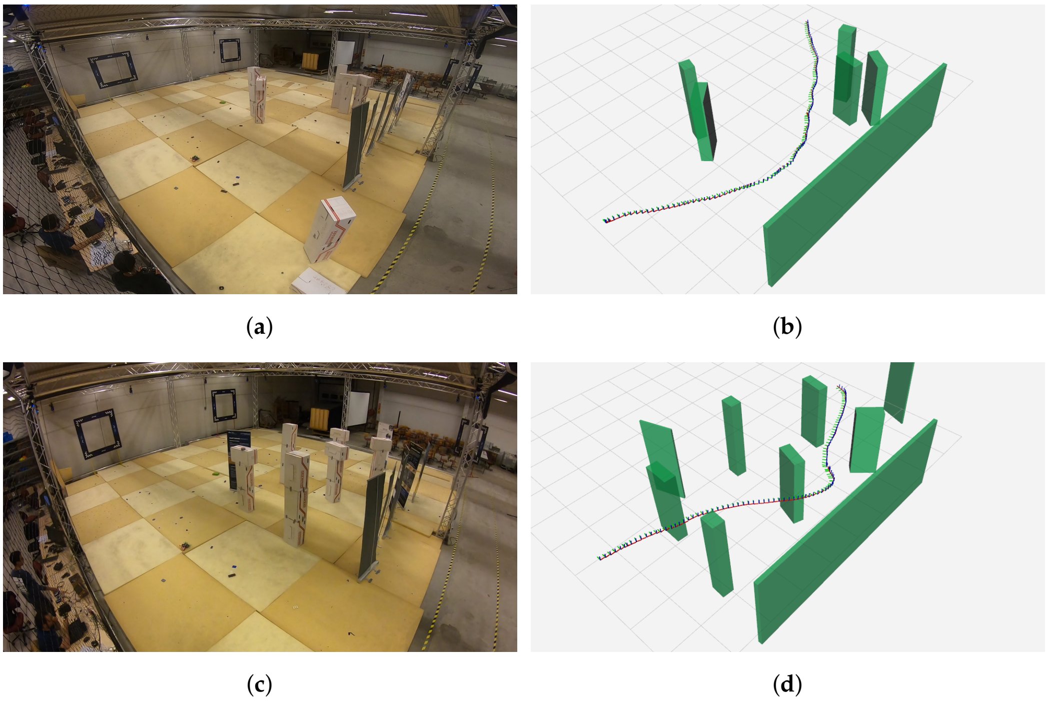
To cope with noisy real depth data, input images are enhanced by using a fast depth dilation algorithm [44] and then resized by cropping the top part of the image to 64 × 64 to feed the policy network. We find that processed depth images help to bridge the gap between simulation and the real world. Unlike the simulations, the generated actions are applied at the same frequency before reaching the waypoint to prevent the quadrotor from stopping after each action, which causes a lot of noise due to pitch movements. Additionally, the quadrotor can track the trajectory faster and smoother. Furthermore, the applied action is calculated as the mean of the recent two actions generated, which prevents the robot from applying oscillating actions, which might cause a failure due to noise. The consecutive oscillating actions are especially expected in narrow passages, where the drone successively observes the obstacles on the right and left and thus decides to switch directions.

The real-time experiments are conducted in two different scenarios, as shown in Figure 7. For the first scenario, a moderate-level experimental setup is designed by grouping obstacles into two groups with wider free space and making the obstacles larger and, hence, easier to observe. The second scenario is denser and more complex, using eight obstacles distributed around the global trajectory. The obstacles are created with cardboard boxes grouped in various configurations. Additionally, a wall-like structure is created using banners on the right side of the flight route (the video can be found: https://youtu.be/HPXXc_R3re8, accessed on 12 July 2022).

SCDP and DDP methods are executed five times each in both moderate and difficult-level scenarios. Table 3 presents the comparison of the two methods in both scenarios. Similar to the simulations, the drone successfully navigates through obstacles, with the narrowest passage being approximately three times the drone's size. The SCDP method succeeds in all trials in the moderate scenario, while one collision is observed with DDP. Similarly, SCDP outperforms in the difficult scenario yet encounters one collision. Although the DDP method also successfully avoids obstacles in most cases, the track cannot be finalized successfully; instead, the drone exits the global trajectory, which shows our framework handles the noisy and complicated inputs better by learning confident actions.

Table 3. Comparison of the SCDP and DDP in real-time experiments. The moderate and difficult-level scenarios are evaluated five times for both methods.

		Moderate Scenario	**Difficult Scenario**
SCDP	**success rate**	100%	80%
	collision rate	0%	20%
	distance (meters)	8	7.4
DDP	**success rate**	80%	0%
	collision rate	20%	20%
	distance (meters)	7.7	7.1

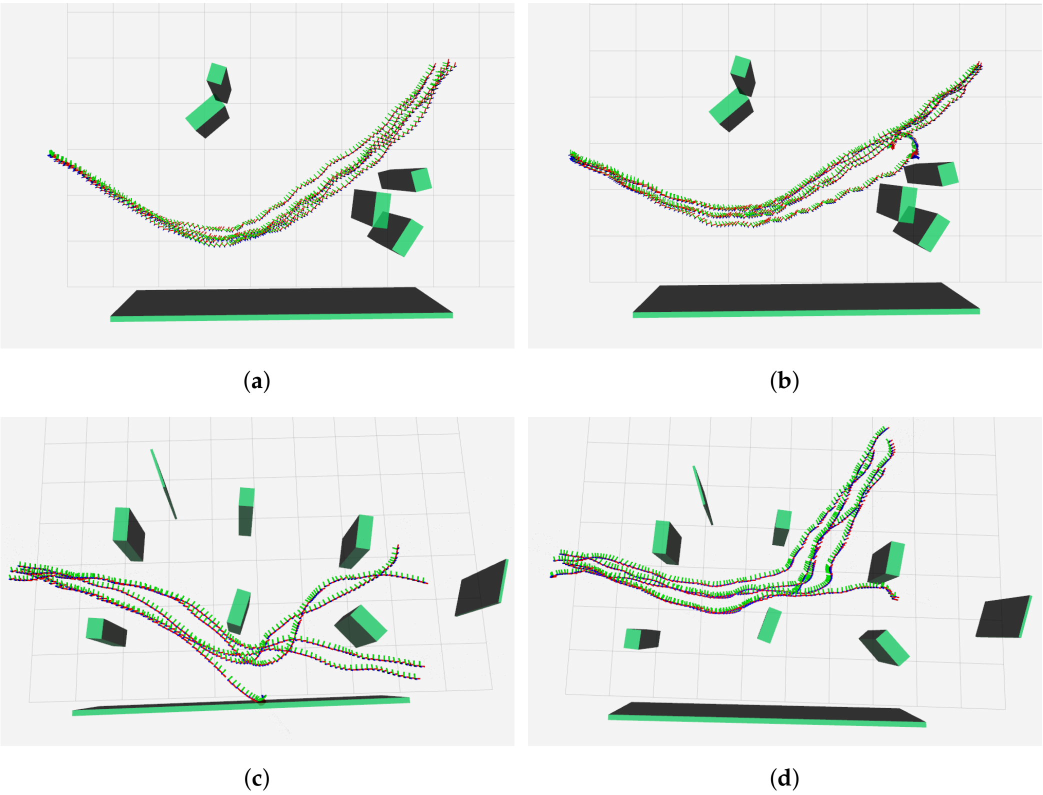


(**a**) (**b**)

(**c**) (**d**)

Figure 7. Moderate and difficult-level real-time experimental setups. (**a**) Real-time evaluation track with moderate-level obstacle configuration. (**b**) Visualization of the moderate-level obstacle configuration and a sample trajectory in RVIZ. (**c**) Real-time evaluation track with difficult-level obstacle configuration. (**d**) Visualization of the difficult-level obstacle configuration and a sample trajectory in RVIZ.

The trajectories obtained with each method and each scenario are visualized in Figure 8. Although it is practically more challenging to obtain the variety of obstacle configurations in real experiments than in simulation, the difficult scenario is observed to contain significant challenges to benchmark algorithms considering the variation of the resulting five trajectories. In contrast, in the moderate scenario, all trajectories follow a similar pattern. Together with the challenge of higher maneuverability, the difficult scenario also introduces more diverse, in-depth observations. Similar to the simulation results, the trajectories obtained by SCDP are smoother than the baseline method.



(**a**) (**b**)

(**c**) (**d**)

Figure 8. Trajectories acquired by five runs in the moderate and difficult scenario by SCDP and DDP methods. (**a**) SCDP in the moderate scenario. (**b**) DDP in the moderate scenario. (**c**) SCDP in the difficult scenario. (**d**) DDP in the difficult scenario.

5. Conclusions

In this work, an end-to-end planner is trained with DRL for safe navigation in cluttered obstacle environments. The end-to-end planning algorithm is trained and tested in comprehensive simulations developed in Webots. While the training of the policy network is handled without dynamics and control to save time, it is successfully sim-to-real transferred for physical evaluations. Moreover, safety boundaries for training are introduced, which successfully prevents the quadrotor from being in hazardous situations. The method is also deployed in real-world indoor environments successfully. The end-to-end planner outperforms a baseline implementation based on the artificial potential field method, which has a lower success rate, especially in cluttered obstacle settings. This shows that SCDP has learned to make better long-term decisions. The real-world experiments demonstrate that the proposed UAV planner trained solely with simulation can work directly in a real environment.

There are also certain limitations of the proposed method to be addressed in future work. First, although the proposed planning method does not require the computation of a map, the neural network-based method still requires significant computational resources in training and also in deployment. Currently, the inference time of the used network is not suitable for real-time robot control. If the algorithm can run continuously in real-time, there is a possibility to provide lower-level control commands, instead of waypoints, to the UAV, which can improve the tracking performance of the robot. Second, due to the black box characteristics of neural networks, the planner cannot be theoretically analyzed similarly to conventional planning methods, such as its completeness.

Author Contributions: Conceptualization, H.I.U. and E.K.; methodology, H.I.U.; software, H.I.U.; validation, H.I.U.; formal analysis, H.I.U.; investigation, H.I.U. and X.H.P.; resources, E.K.; data curation, H.I.U.; writing—original draft preparation, H.I.U.; writing—review and editing, H.I.U.; visualization, H.I.U.; supervision, E.K.; project administration, E.K.; funding acquisition, E.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the European Union's Horizon 2020 Research and Innovation Program (OpenDR) grant number 871449. This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All of the codes and data will be available at https://github.com/open-airlab/safe-continuous-depth-planning.git, accessed on 2 July 2022.

Acknowledgments: The authors would like to thank Abdelhakim Amer for his technical support in conducting real-time experiments.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

DRL	deep reinforcement learning
UAV	unmanned aerial vehicle
PPO	proximal policy optimization
FOV	field of view
PID	proportional-integral-derivative
MDP	Markov decision process
SITL	software-in-the-loop
ROS	robot operating system
SCDP	safe continuous depth planner
CDP	continuous depth planner
DDP	discrete depth planner
APF	artificial potential field

Appendix A

The following variables are used in this manuscript:

Table A1. Common variables in the manuscript.

t	discrete timestep
s_t	state at timestep t
a_t	action at timestep t
r_t	reward at timestep t
I_{depth}	matrix representing the depth image
L	length of the global trajectory
$\mathcal{U}$	uniform distribution
$\mathcal{N}$	normal distribution

References

1. Pham, H.X.; Ugurlu, H.I.; Le Fevre, J.; Bardakci, D.; Kayacan, E. Deep learning for vision-based navigation in autonomous drone racing. In *Deep Learning for Robot Perception and Cognition*; Elsevier: Amsterdam, The Netherlands, 2022; pp. 371–406.
2. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
3. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]

4. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
5. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust region policy optimization. In Proceedings of the International Conference on Machine Learning, PMLR, Lille, France, 6 July–11 July 2015; pp. 1889–1897.
6. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.
7. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
8. Muratore, F.; Ramos, F.; Turk, G.; Yu, W.; Gienger, M.; Peters, J. Robot learning from randomized simulations: A review. *Front. Robot. AI* **2022**, *9*, 799893. [CrossRef] [PubMed]
9. Hoeller, D.; Wellhausen, L.; Farshidian, F.; Hutter, M. Learning a state representation and navigation in cluttered and dynamic environments. *IEEE Robot. Autom. Lett.* **2021**, *6*, 5081–5088. [CrossRef]
10. Pham, H.X.; Sarabakha, A.; Odnoshyvkin, M.; Kayacan, E. PencilNet: Zero-Shot Sim-to-Real Transfer Learning for Robust Gate Perception in Autonomous Drone Racing. *arXiv* **2022**, arXiv:2207.14131.
11. Molchanov, A.; Chen, T.; Hönig, W.; Preiss, J.A.; Ayanian, N.; Sukhatme, G.S. Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; pp. 59–66.
12. Morales, T.; Sarabakha, A.; Kayacan, E. Image generation for efficient neural network training in autonomous drone racing. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8.
13. Koch, W.; Mancuso, R.; West, R.; Bestavros, A. Reinforcement learning for UAV attitude control. *ACM Trans. Cyber Phys. Syst.* **2019**, *3*, 22. [CrossRef]
14. Ugurlu, H.I.; Kalkan, S.; Saranli, A. Reinforcement Learning versus Conventional Control for Controlling a Planar Bi-rotor Platform with Tail Appendage. *J. Intell. Robot. Syst.* **2021**, *102*, 77. [CrossRef]
15. Camci, E.; Kayacan, E. Learning motion primitives for planning swift maneuvers of quadrotor. *Auton. Robot.* **2019**, *43*, 1733–1745. [CrossRef]
16. Dooraki, A.R.; Lee, D.J. An innovative bio-inspired flight controller for quad-rotor drones: Quad-rotor drone learning to fly using reinforcement learning. *Robot. Auton. Syst.* **2021**, *135*, 103671. [CrossRef]
17. Brunke, L.; Greeff, M.; Hall, A.W.; Yuan, Z.; Zhou, S.; Panerati, J.; Schoellig, A.P. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annu. Rev. Control Robot. Auton. Syst.* **2022**, *5*, 411–444. [CrossRef]
18. Han, H.; Cheng, J.; Xi, Z.; Yao, B. Cascade Flight Control of Quadrotors Based on Deep Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2022**, *7*, 11134–11141. [CrossRef]
19. Kaufmann, E.; Bauersfeld, L.; Scaramuzza, D. A Benchmark Comparison of Learned Control Policies for Agile Quadrotor Flight. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 10504–10510.
20. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
21. Stentz, A. Optimal and efficient path planning for partially known environments. In *Intelligent Unmanned Ground Vehicles*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 203–220.
22. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. In Proceedings of the 1985 IEEE International Conference on Robotics and Automation, St. Louis, MO, USA, 25–28 March 1985; Volume 2, pp. 500–505.
23. LaValle, S.M. Rapidly-Exploring Random Trees: A New Tool for Path Planning. 1998. Available online: https://citeseerx.ist.psu.edu/viewdoc/citations;jsessionid=FDD7D4058FECC1206F4FA333A3286F56?doi=10.1.1.35.1853 (accessed on 14 July 2022)
24. Zhou, D.; Wang, Z.; Schwager, M. Agile coordination and assistive collision avoidance for quadrotor swarms using virtual structures. *IEEE Trans. Robot.* **2018**, *34*, 916–923. [CrossRef]
25. Raigoza, K.; Sands, T. Autonomous Trajectory Generation Comparison for De-Orbiting with Multiple Collision Avoidance. *Sensors* **2022**, *22*, 7066. [CrossRef] [PubMed]
26. Feng, S.; Sebastian, B.; Ben-Tzvi, P. A Collision Avoidance Method Based on Deep Reinforcement Learning. *Robotics* **2021**, *10*, 73. [CrossRef]
27. Dooraki, A.R.; Lee, D.J. An end-to-end deep reinforcement learning-based intelligent agent capable of autonomous exploration in unknown environments. *Sensors* **2018**, *18*, 3575. [CrossRef] [PubMed]
28. Kang, K.; Belkhale, S.; Kahn, G.; Abbeel, P.; Levine, S. Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 6008–6014.
29. Bonatti, R.; Madaan, R.; Vineet, V.; Scherer, S.; Kapoor, A. Learning visuomotor policies for aerial navigation using cross-modal representations. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020; pp. 1637–1644.
30. Bonatti, R.; Wang, W.; Ho, C.; Ahuja, A.; Gschwindt, M.; Camci, E.; Kayacan, E.; Choudhury, S.; Scherer, S. Autonomous aerial cinematography in unstructured environments with learned artistic decision-making. *J. Field Robot.* **2020**, *37*, 606–641. [CrossRef]

31. Polvara, R.; Patacchiola, M.; Hanheide, M.; Neumann, G. Sim-to-Real Quadrotor Landing via Sequential Deep Q-Networks and Domain Randomization. *Robotics* **2020**, *9*, 8. [CrossRef]
32. Bartolomei, L.; Kompis, Y.; Pinto Teixeira, L.; Chli, M. Autonomous Emergency Landing for Multicopters Using Deep Reinforcement Learning. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022), Kyoto, Japan, 23–27 October 2022.
33. Muñoz, G.; Barrado, C.; Çetin, E.; Salami, E. Deep reinforcement learning for drone delivery. *Drones* **2019**, *3*, 72. [CrossRef]
34. Doukhi, O.; Lee, D.J. Deep reinforcement learning for end-to-end local motion planning of autonomous aerial robots in unknown outdoor environments: Real-time flight experiments. *Sensors* **2021**, *21*, 2534. [CrossRef] [PubMed]
35. Loquercio, A.; Kaufmann, E.; Ranftl, R.; Müller, M.; Koltun, V.; Scaramuzza, D. Learning High-Speed Flight in the Wild. In Proceedings of the Science Robotics, New York, NY, USA, 27 June–1 July 2021.
36. Camci, E.; Campolo, D.; Kayacan, E. Deep reinforcement learning for motion planning of quadrotors using raw depth images. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–7.
37. Dooraki, A.R.; Lee, D.J. A Multi-Objective Reinforcement Learning Based Controller for Autonomous Navigation in Challenging Environments. *Machines* **2022**, *10*, 500. [CrossRef]
38. Michel, O. Cyberbotics Ltd. Webots™: Professional mobile robot simulation. *Int. J. Adv. Robot. Syst.* **2004**, *1*, 5. [CrossRef]
39. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. Openai gym. *arXiv* **2016**, arXiv:1606.01540.
40. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An open-source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009; Volume 3, p. 5.
41. Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; Dormann, N. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *J. Mach. Learn. Res.* **2021**, *22*, 1–8.
42. Meier, L.; Tanskanen, P.; Fraundorfer, F.; Pollefeys, M. Pixhawk: A system for autonomous flight using onboard computer vision. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 2992–2997.
43. Faessler, M.; Franchi, A.; Scaramuzza, D. Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robot. Autom. Lett.* **2017**, *3*, 620–626. [CrossRef]
44. Ku, J.; Harakeh, A.; Waslander, S.L. In Defense of Classical Image Processing: Fast Depth Completion on the CPU. In Proceedings of the 2018 15th Conference on Computer and Robot Vision (CRV), Toronto, ON, Canada, 9–11 May 2018; pp. 16–22.

E

IMPROVING INERTIAL-BASED UAV LOCALIZATION USING DATA-EFFICIENT DEEP REINFORCEMENT LEARNING

Dimitrios Tsiakmakis, Nikolaos Passalis, and Anastasios Tefas

Computational Intelligence and Deep Learning Group
Artificial Intelligence and Information Analysis Lab., Department of Informatics,
Aristotle University of Thessaloniki, Thessaloniki, Greece
{*dtsiakma, passalis, tefas*}*@csd.auth.gr*

ABSTRACT

Precise localization is a critical task for many Unmanned Aerial Vehicle (UAV)-based applications. Inertial-based navigation, which relies on Inertial Measurement Units (IMUs), is extensively used to this end, due to its low-cost and small footprint. However, IMU-based localization leads to accumulating significant localization errors. To overcome this limitation, in this paper we propose a data-efficient Deep Reinforcement Learning (DRL) method that enables learning how to correct localization errors from IMUs leading to more precise localization. In contrast with supervised approaches, the proposed method employs a novel data augmentation and regularization approach, which requires collecting a minimal number of real examples, while it is also platform-agnostic and can account for manufacturing impressions. The effectiveness of the proposed method is demonstrated both in a simulation environment, as well as using a real UAV.

Index Terms— Deep Reinforcement Learning, Inertial-based Localization, Data Augmentation

1. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are increasingly used in various applications, ranging from precision agriculture [1] and search and rescue missions [2] to indoor surveillance [3]. A common point between these applications, along with virtually every UAV-based application, is the need for precise UAV localization. UAV localization is critical both for mission control purposes, i.e., some tasks are related to the location of a UAV, as well as for safety purposes, i.e., avoid flights over restricted areas. Several different approaches have been developed for UAV localization, with each one relying on different sensors and providing a different level of accuracy.

Perhaps among the most well known localization approaches is using satellite-based radio-navigation systems, such as the Global Positioning System (GPS) [4, 5]. Despite its low cost the accuracy of GPS and related systems is usually low. Indeed, according to the official GPS documentation, GPS-enabled devices are normally accurate to within a 4.9 meters (16 feet), which is unacceptable for many applications. At the same time, there are several locations where there is no GPS coverage [6], while such approaches cannot be used indoors. The use of real-time kinematic positioning can further reduce the errors introduced in satellite-based radio navigation [7], yet it typically requires the use of extra base stations, which increases the cost and reduces the flexibility of UAVs. Light detection and ranging approaches [8, 9], also known as LIDAR, can be also used to provide accurate localization, especially when coupled with simultaneous localization and mapping (SLAM) approaches [10]. However, such approaches involve the use of very expensive sensors and they have greater computational and energy demands.

On the other hand, the use of Inertial Measurement Units (IMUs) [11], which is a combination of accelerometers, gyroscopes, and magnetometers can provide very low-cost solutions that also do not rely on any kind of external hardware or communication (e.g., satellites, base stations, etc.). The localization is accomplished by utilizing IMU data for dead reckoning, called Inertial Navigation System (INS) [12]. The recent demand for smaller sensors that can be integrated into cutting-edge technologies, has prompted engineers to build a Micro Electro-Mechanical System (MEMS) which can provide low-cost and low-footprint sensors that can be very easily integrated with virtually any UAV and provide real-time measurements. Despite the cost and flexibility benefits of such systems, they also come with accuracy limitations. IMU sensors monitor the linear acceleration and rotational velocity of the body with just a very small degree of inaccuracy every time. However, over long periods these errors can accumulate leading to significant position drifts that can comprise their application, especially when used as a sole localization sensor in mission critical applications.

These limitations have fueled research on methods for improving inertial-based navigation for UAVs [13, 14, 15].

This work was supported by the European Union's Horizon 2020 Research and Innovation Program (OpenDR) under Grant 871449. This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

Many recent approaches built upon Deep Learning (DL)-based models that allow for significantly improving the localization process. However, despite these improvements, these approaches suffer from a significant drawback. They mostly rely on supervised learning (either regression-based or classification-based), which in turn requires a large number of samples to be collected and annotated to train the corresponding methods. At the same time, such approaches are typically linked to the hardware used for data collection and their performance deteriorates when deployed on different hardware, requiring collecting data again and re-training the models. Furthermore, even when using the same hardware, manufacturing tolerances might lead to sensors that have different noise characteristics, which make the application of supervised learning approaches challenging.

Deep Reinforcement Learning (DRL) can overcome these limitations [16], since it enables autonomous agents to learn just by interacting with the environment. Indeed, DRL methods have shown to achieve remarkable results in a variety of tasks in recent years, often outperforming humans [17, 18]. However, directly applying DRL for improving inertial-based navigation for UAVs is not directly feasible since: a) a feedback signal is still required in order to measure the quality of the learned policy and b) a large number of episodes are typically required for learning. Even though the first limitation can be easily addressed, e.g., by using visual cues to provide a feedback signal, the low-data efficiency of DRL approaches still pose a significant limitation that prohibits such approaches from being deployed in practice.

Based on the aforementioned observations, in this work we propose a pipeline that can allow for easing these limitations, enabling data-efficient DRL on UAVs for inertial-based navigation. The proposed method employs a two-stage pipeline. In the first stage, a backbone is trained using supervised learning in a simulator. Acquiring ground truth annotations in a simulator is easy and cheap, so this approach can enabled us to train a backbone that can capture the dynamics of the behavior of IMUs without targeting a specific sensor. Then, the employed DL model is fine-tuned using DRL on a real UAV. Since this can be an especially data-intensive process, we further propose: a) a data augmentation method that can generate multiple simulated episode trajectories just from one real episode and b) a regularizer than can provide additional feedback when fine-tuning the learned policy based on the sign of the measured reward signal. For acquiring a reward signal, we propose a simple, yet efficient visual landmark-based approach that can be used even with low-resolution cameras. As we demonstrated through extensive experiments on regressing the 2D position of a UAV, the proposed method can indeed lead to significant performance improvements over the employed baseline approaches.

The rest of the paper is structured as follows. First, Section 2 introduces the proposed methodology, while the experimental evaluation of the proposed method is provided in Section 3. Finally, conclusions are drawn in Section 4.

2. PROPOSED METHOD

2.1. Background

The simplest method to localize a UAV using a inertial-based approach is to employ a first-order numerical approach to solve ordinary differential equations (ODEs), which is sometimes referred to as Euler's method. Specifically, Euler's method employs the basic formula:

$$y(t+h) = y(t) + h * f(t,x), \tag{1}$$

where the $f(t,x)$ is simply the dx/dt amount. In our case, time-step is represented by h, time by t, position by x, and velocity by $f(t,x)$. Thus, we estimate the next instant position, taking into account an initial position at every constant time-step. Note, we assume that velocity between two measurements remains constant throughout the flight. This simple approach enables UAV localization through IMU sensors that can be provide acceleration/speed estimates. However, the noise that it is introduced by IMUs can lead to a significant drift in the estimation of UAV position using this approach.

Neural Networks (NNs) can be employed in a supervised learning setting in order to learn how these errors should be corrected, allowing for improving the localization accuracy. Let $G_{\mathbf{W}} : \mathbb{R}^m \to \mathbb{R}^n$ denote a regression model, parameterized by weights $\mathbf{W}$, with m inputs and n outputs. Also, let $\mathbf{v} \in \mathbb{R}^m$ denote a vector that contains the most recent velocity measurements, including the current one, provided by the IMU. Then, the model $\mathbf{y} = G_{\mathbf{W}}(\cdot)$ can be trained to provide corrected estimates for the current velocity, denoted by $\mathbf{y} \in \mathbb{R}^n$. Note that typically $n = 2$, since we are interested in estimating the speed in the 2d plane, ignoring the speed in the vertical to this plane axis (height), since altimeter sensors can provide reliable estimates for the vertical speed. Similarly, m is typically set to $m = 2 \times T$, where T denotes the history (number of time steps) to include in the input that will be fed to the neural network that will provide the corrected speed estimates. Training $G_{\mathbf{W}}(\cdot)$ is straightforward, since we just need to collect enough training samples of IMU velocity estimates and the corresponding ground truth velocities. Then, the mean square error can be used for training the neural network estimator using gradient descent. Furthermore, note that typically the estimator $G_{\mathbf{W}}(\cdot)$ is fitted to regress the velocity errors instead of the actual velocities, since this accelerates the learning process. After estimating the velocity error, then the corrected velocity can be used in (1) to acquire a more reliable estimation of the UAV's position.

2.2. Data-efficient DRL-based training

Even through the aforementioned process can be easily performed inside a simulation environment, it is very expensive to perform using real UAVs, since extra equipment is required for measuring the accurate position of a UAV and a large number of samples need to be collected. Therefore, in this paper we propose a two step pipeline that consists of the following steps: a) train a generic DL-based backbone model in a simulator to correct generic IMU errors and b) fine-tune this model on a real UAV using DRL. This process can overcome the need to collect a large number of annotated training samples using a real UAV. However, as mentioned in Introduction, DRL methods are also data intensive. To overcome this limitation, we proposed to use a data augmentation method coupled with a regularizer that can increase data efficiency.

In this work, we propose to employ a DRL agent in order to provide *continuous corrections* to UAVs estimates. More specifically, we introduce a *virtual agent* that controls the estimation of the UAV's position. Hence, there are two positions: the actual UAV and a sphere indicating its estimated position. The DRL agent controls the latter by providing continuous corrections in the two axes of the 2d plane. This setup also enables an easy way to acquire the feedback signal for training the agent both in simulation and in real word. More specifically, in simulation, for each episode the UAV runs a predetermined course, e.g,. 2 meters to the North and 1 meter to the East. Then, when the episode is finished, we project the virtual UAV's position as a black mark onto the floor, and then, the UAV uses its camera to snap an image and provide the reward signal. To present this concept with an example to be more intuitively, if the position of the UAV is accurate, the black mark will be centered in the captured image. In contrast, the black mark would be in a different location if the positions of the actual and virtual locations are different. Then, the reward for each axis k can be calculated as:

$$R_k = \frac{1}{1+|p_k|}, \tag{2}$$

where p_k is the distance in pixels between the black mark and the center of the captured image (which represents the position of the UAV). In real deployment, the black mark will represent the desired UAV position based on the provided control command. Then, the reward can be calculated in a similar fashion and provide the same behavior (maximize as the agent better corrects the displacement estimations). This process enables training the DRL agent without having access to ground truth data regarding the actual speed and/or displacement on each step.

In this work, we employ Proximal Policy Optimization (PPO) [19] for training the agent. This is without loss of generality, since any DRL method that can support continuous action spaces can be used. Furthermore, since the aim is to accelerate the learning process as much as possible, we employed the supervised learning model that was pre-trained on the simulator to initialize the weights of the actor model. Therefore, the DRL method is employed to fine-tune the DL model to the actual hardware used in the UAV. To further increase the efficiency of the learning process we designed and used a data augmentation method to create additional episodes during the training. The main concept is that the reward of an episode remains unchanged if the angle of velocity vectors and the actions are rotated simultaneously. To this end, the proposed method selects the episode with the highest reward from the buffered episodes and then several synthetic episodes are created by rotating the velocities and actions by a random angle $\phi \in [0, 360)$.

Finally, to further increase the learning speed and minimize the number of training episodes required to fine-tune the agent to the actual IMU used, we propose employing a hint regularizer that provides additional supervision based on whether the agent is currently overshooting or undershooting the desired position (as indicated by the sign of the distance in (2)). Therefore, the regularizer for each axis is defined as:

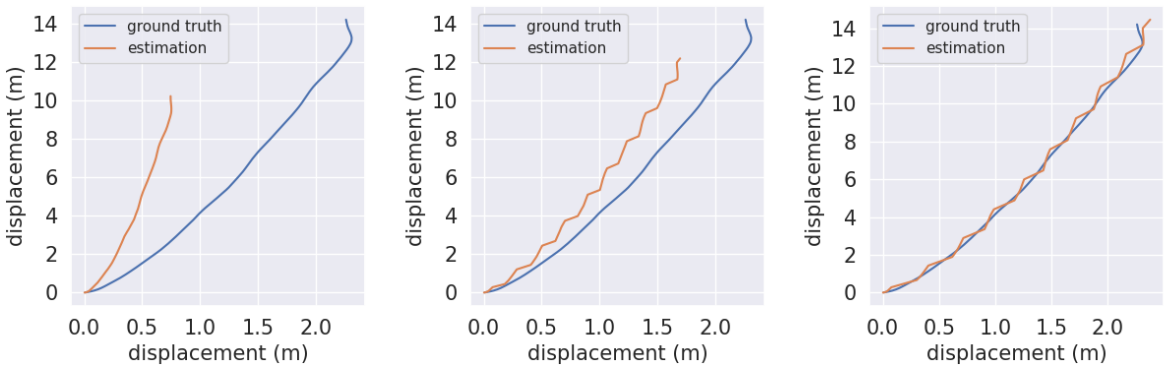
$$\mathcal{L}_{reg,k} = -\alpha_{reg} \cdot \delta_k \cdot g_{RL}(x) \tag{3}$$

where α_{reg} is the weight of the regularizer, δ_k it is a binary variable $\{-1, 1\}$ indicating whether we are currently overshooting or undershooting the target position and $g_{RL}(x)$ is the agent's output. Then, the overall loss is calculated by simply adding the regularizer for both axis to the PPO loss.

3. EXPERIMENTAL EVALUATION

We conducted experiments using both a simulated environment, i.e., for supervised learning and validation of the proposed DRL approach, as well as a real UAV. For the simulated experiments, we employed Webots [20]. For supervised learning, we collected 500 episodes with velocities and ground truth positions. We also experimentally found that the IMU measurement is biased depending on the vehicle's velocity and it is always underestimated. Therefore, we estimate the velocity bias in the simulation environment as $v/(1 + 1/(1 + c * |v|))$, where c is an IMU-depended factor and v is the ground truth velocity. For the supervised learning model, we used $c = 5$, while for the evaluation we used in all cases $c = 2$ to simulate the drift that can occur due to hardware changes.

The IMU was pooled with a frequency of 25Hz, while each episode has a total length of 10 seconds. We also used an MLP with two hidden layers as a backbone, with 12 neurons each with the `tanh` activation. Then, the network culminates in two branches that output corrections for each dimension. In every branch, there are two extra trainable parameters, which are used for shifting and scaling the output of the network. We found that when we re-train the network with new data from alternative sensors, the convergence succeeds more quickly due to these variables, which allows for promptly shifting and scaling the output without refitting all the weights of the backbone. The network receives a one-second time frame of velocities, i.e., 25 measurements along each of the two axes, and returns two corrections, one of each



Baseline **Supervised method** **RL-based fine-tuning**

Fig. 1. Comparison between baseline (Euler's method) (left), supervised training (middle) and RL-based fine-tuning (right). Each axis corresponds to the displacement of a UAV in the 2d space when flying on a pre-determined course.

axis. The mean squared error was used for training the supervised model using the Adam optimizer and a learning rate of 3×10^{-2}. The optimization ran for 10 epochs with a batch size of 512, while the learning rate was reduced when the learning process approached a plateau (the reduction factor was set to 0.1 and patience to 10). For the DRL setup, we used the same configuration as the previous model for the actor, while an MLP with 2 hidden layers with 12 neurons each was used as a critic. The models were trained for 30 epochs with a learning rate of 9×10^{-5} and 5×10^{-3} in actor and critic accordingly. PPO algorithm was used for fitting the DRL agent, while the clipping factor was set to 0.15.

First, we evaluated the proposed method in simulation using Webots. The experimental comparison between Euler's method with no corrections, the supervised method trained on a model with $c = 5$, and the proposed RL-based fine-tuning of the supervised model are shown in Fig. 1. As demonstrated, using DRL to fine-tune the model trained in simulation to adjust the actual characteristics of a specific UAV leads to significant improvements. Based on these observations we evaluated the ability of the proposed data-efficient RL approach compared to the baseline Euler method. The results are reported in Table 1 where we report the mean squared error (MSE), mean distance (MD), mean positional error (MPE), and absolute trajectory error (ATE) between the ground truth displacement and the one estimated by the DRL models. These results indicate that the proposed method can improve DRL agents' performance when training under a constrained number of episodes (i.e., 30 episodes). Note that as the duration of an episode increases, the error still accumulates. Nonetheless, the proposed method manages to significantly reduce all the error metrics compared to the baseline. Finally, we also validated the proposed method using data collected from a DJI Mavic mini 2 UAV using different velocities. The results reported in Table 2 again confirm that for a wide range of different speeds the proposed method still leads to better performance.

Table 1. DRL fine-tuning evaluation on Webots using a distribution shift scenario (c changes from 5 to 2).

	10 secs		
metrics	baseline	supervised	**proposed**
MSE	12.297	2.975	**0.101**
MD	3.438	1.692	**0.286**
MPE	1.714	0.931	**0.212**
ATE	2.008	1.073	**0.251**
	100 secs		
MSE	1225.922	312.100	**18.832**
MD	34.437	17.444	**3.816**
MPE	18.376	141.05	**2.053**
ATE	21.014	10.696	**7.613**

Table 2. DRL fine-tuning evaluation using a DJI Mavic 2 UAV. The percentage of estimated distance covered to the true distance (MPE, %) is reported for different flying speeds.

Vel. (m/s)	0.1	0.3	1.1	1.4	2.2	2.8
baseline (%)	46.58	86.20	87.51	94.46	96.06	95.70
proposed (%)	**80.59**	**98.14**	**106.89**	**99.31**	**102.98**	**98.14**

4. CONCLUSIONS

In this paper, we proposed a data-efficient DRL approach for improving inertial-based navigation for a UAV. The proposed method employed a two-stage pipeline: in the first stage, a backbone is trained using supervised learning, while in the second stage a data-efficient DRL-based approach for fine-tuning is used. We demonstrated that the proposed method can indeed allow for improving inertial-based navigation, focusing on cases where the IMUs used in UAVs can have different characteristics requiring UAV-specific fine-tuning using a very small number of real episodes.

5. REFERENCES

[1] Panagiotis Radoglou-Grammatikis, Panagiotis Sarigiannidis, Thomas Lagkas, and Ioannis Moscholios, "A compilation of uav applications for precision agriculture," *Computer Networks*, vol. 172, pp. 107148, 2020.

[2] Ebtehal Turki Alotaibi, Shahad Saleh Alqefari, and Anis Koubaa, "Lsar: Multi-uav collaboration for search and rescue missions," *IEEE Access*, vol. 7, pp. 55817–55832, 2019.

[3] Natthawat Boonyathanmig, Sarun Gongmanee, Prachaya Kayunyeam, Piyavat Wutticho, and Sethakarn Prongnuch, "Design and implementation of mini-uav for indoor surveillance," in *Proceedings of the 9th International Electrical Engineering Congress (iEECON)*, 2021, pp. 305–308.

[4] Salah Sukkarieh, Eduardo Mario Nebot, and Hugh F Durrant-Whyte, "A high integrity imu/gps navigation loop for autonomous land vehicle applications," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 3, pp. 572–578, 1999.

[5] Songlai Han and Jinling Wang, "Integrated gps/ins navigation system with dual-rate kalman filter," *GPS Solutions*, vol. 16, no. 3, pp. 389–404, 2012.

[6] "GPS Accuracy," `https://www.gps.gov/systems/gps/performance/accuracy/`, Accessed: 2022-07-21.

[7] Patrick Henkel, Ulrich Mittmann, and Michele Iafrancesco, "Real-time kinematic positioning with gps and glonass," in *Proceedings of the 24th European Signal Processing Conference (EUSIPCO)*, 2016, pp. 1063–1067.

[8] Michel Jaboyedoff, Thierry Oppikofer, Antonio Abellán, Marc-Henri Derron, Alex Loye, Richard Metzger, and Andrea Pedrazzini, "Use of lidar in landslide investigations: a review," *Natural Hazards*, vol. 61, no. 1, pp. 5–28, 2012.

[9] Frederick G Fernald, "Analysis of atmospheric lidar observations: some comments," *Applied Optics*, vol. 23, no. 5, pp. 652–653, 1984.

[10] Dinh Van Nam and Kim Gon-Woo, "Solid-state lidar based-slam: A concise review and application," in *Proceedings of th IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2021, pp. 302–305.

[11] Norhafizan Ahmad, Raja Ariffin Raja Ghazilla, Nazirah M Khairi, and Vijayabaskar Kasi, "Reviews on various inertial measurement unit (imu) sensor applications," *International Journal of Signal Processing Systems*, vol. 1, no. 2, pp. 256–262, 2013.

[12] Billur Barshan and Hugh F Durrant-Whyte, "Inertial navigation systems for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 3, pp. 328–342, 1995.

[13] Martin Brossard, Axel Barrau, and Silvere Bonnabel, "Rins-w: Robust inertial navigation system on wheels," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 2068–2075.

[14] Santiago Cortés, Arno Solin, and Juho Kannala, "Deep learning based speed estimation for constraining strapdown inertial navigation on smartphones," in *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2018, pp. 1–6.

[15] Sachini Herath, Hang Yan, and Yasutaka Furukawa, "Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, & new methods," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3146–3152.

[16] Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee, "Sample-efficient reinforcement learning with stochastic ensemble value expansion," *Proceedings of the Advances in Neural Information Processing Systems*, vol. 31, 2018.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[18] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al., "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[19] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[20] Olivier Michel, "Cyberbotics ltd. webots™: professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp. 5, 2004.

F

SingleDemoGrasp: Learning to Grasp From a Single Image Demonstration

Amir Mehman Sefat[1], Alexandre Angleraud[1], Esa Rahtu[2] and Roel Pieters[1]

***Abstract*— Learning-based grasping models typically require a large amount of training data and training time to generate an effective grasping model. Alternatively, small non-generic grasp models have been proposed that are tailored to specific objects by, for example, directly predicting the object's location in 2/3D space, and determining suitable grasp poses by post processing. In both cases, data generation is a bottleneck, as this needs to be separately collected and annotated for each individual object and image. In this work, we tackle these issues and propose a grasping model that is developed in four main steps: 1. Visual object grasp demonstration, 2. Data augmentation, 3. Grasp detection model training and 4. Robot grasping action. Four different vision-based grasp models are evaluated with industrial and 3D printed objects, robot and standard gripper, in both simulation and real environments. The grasping model is implemented in the OpenDR toolkit at: `https://github.com/opendr-eu/opendr/tree/master/projects/control/single_demo_grasp`.**

***Index Terms*— Grasping, Deep Learning in Grasping and Manipulation, Perception for Grasping and Manipulation**

I. INTRODUCTION

Collaborative robots have gained popularity in industry as they are designed to be safe, particularly where human and robot share the workspace. Accompanied by intuitive programming interfaces, robot tasks can be programmed efficiently [1]. Despite the benefits, the application of cobots in industrial settings are mainly limited to offline tasks where the actions and targets are defined to the system beforehand [2]. For example, in the majority of pick and place tasks, object poses are fixed, and the robotic arm should reach a predefined grasp pose. Although there is great interest in the generation of object grasp models from visual data, [3], limitations still exist, for example, in terms of object type coverage, grasp success, training complexity, model inference time, etc. In particular, while grasp models have reported high success rate (e.g., Dex-Net 4.0 [4] achieves above 95% accuracy), this typically only holds for the task at hand, i.e., bin picking with generic household items. Evaluating such grasping model on objects that exhibit different properties (e.g., industrial parts) might result in unsuccessful grasp attempts and an overall lower accuracy. In addition, grasp modelling requires vast amounts of training data and considerable training time on high-performance computing clusters. Consequently, state of the art grasping models can be large in size and slow to execute [5].

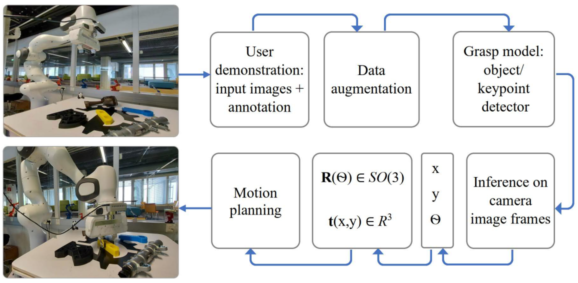

User demonstration: input images + annotation
Data augmentation
Grasp model: object/ keypoint detector
Inference on camera image frames
x
y
Θ
R(Θ) ∈ *SO*(3)
t(x,y) ∈ R^3
Motion planning

Fig. 1: Overview of the proposed grasping model.

Extending an existing dataset and retraining a grasping model is in most cases not an option, due to unavailable data or limitations in resources and computation power. These problems exist in particular for small and medium sized enterprises (SME), which typically don't have the knowledge and resources available for data collection, model training and fine-tuning.

Our main observation to motivate this work is that collecting or generating training data for a grasp detection model is a tedious, time-consuming and costly task, which is often out of reach in industrial environments. Even though plenty datasets can be found [6], [7], each are limited (to some extend) to the objects they contain. Industrial SMEs require the handling of objects that, in most cases, do not resemble objects in these datasets, or the objects themselves can change depending on a customer's requirement. Moreover, as the handling of such objects requires a human preselected grasp pose, a single generic model for all objects is unfeasible.

In this work, we aimed to tackle this issue by investigating visual learning-based approaches for object grasp detection, with human annotation of a desired object grasp pose. For this, different variants of the R-CNN architecture from Detectron2 [8] are evaluated for the fast generation of a grasping model. Single or multiple image demonstrations with human annotations of an object grasp are collected and utilized to generate an augmented object training dataset, from which a detection model is trained. Object grasp detection results (object grasp position and orientation on a plane) are transformed to a 3D grasp pose and given as input for robot motion planning (see Fig. 1). Four different networks are developed and evaluated in simulation (Webots) with eight different objects. The grasp detection model with best performance was then implemented and evaluated in real robot experiments (Franka robot with standard gripper).

[1]Unit of Automation Technology and Mechanical Engineering, [2]Unit of Computing Sciences, Tampere University, 33720, Tampere, Finland; `firstname.surname@tuni.fi`

The main contributions of our work are as follows:

- **Four different planar grasp models** based on pre-trained object detection models.
- **Data generation by image augmentation** of image demonstrations with human annotation.
- **Training of grasp detection networks** in short time.
- **Evaluation of the grasping approach** in simulation and with real experiments.

The paper is organized as follows: Section II reviews related works and state of the art in computer vision and grasping methods. Section III and Section IV define the considered problem statement and research methodology, respectively. Section V describes the implementation details of the proposed grasping model, and Section VI reports the results and provides an analysis. Finally, Section VII concludes the work.

II. Related Work

In the context of robotics, object detection, pose estimation and grasp detection are closely related, as grasp poses or grasp actions can be directly generated from an object pose. This section presents a brief overview of related approaches.

A. Object Detection and Pose Estimation

Traditionally, object detection and pose estimation algorithms have utilized classical 2D features that exploit local salient details, such as corners, edges and ridges. Well-known detectors like SIFT [9] can extract robust keypoints from a scene by relying on texture on objects or of the scene itself. Texture-less keypoint detection, on the other hand, utilizes geometrical primitives as features in methods such as BIND [10]. In addition, alternatives to traditional keypoints are template matching, where a image patch provides the template to localize within an image, or deep features that extract keypoints based on high-level cues captured by convolutional neural networks. The latter is a recent development that has gained popularity due to their data-driven property and promising performance [11], as compared to hand-crafted features. Analogous to 2D keypoints for RGB images, 3D keypoints can be extracted from 3D data representations, such as pointclouds or volumetric images [12]. Following the detection of keypoints from a raw image, follow-up steps include the description of the keypoint and the matching of them over two or multiple images. In a similar manner, Convolutional Neural Network (CNN) based detectors, such as Faster R-CNN [13], could be utilized to detect objects, after which a grasp pose needs to be be extracted.

Object pose estimation on the other hand directly estimates the 6D pose of an object. Similar to object detection, different approaches exist, such as correspondence-based methods 3DMatch [14], template-based methods such as PoseCNN [15] and voting based methods such as DenseFusion [16]. Again, once an object pose is extracted, this needs to be converted to a grasp pose suitable for a robot to hold an object.

B. Grasp Detection

Object grasp detection aims to derive a grasp pose directly from sensor measurements and can be divided in several categories to differentiate between approaches and their assumptions. For example, the representation of a grasp is an important consideration and determines the complexity of the problem and its application. When considering only a planar grasp pose representation, grasp detection is simplified to finding the object and its orientation on a planar surface, typically represented as an (oriented) bounding box, where the center of the box is the grasp position [17], [18]. On the other hand, in case a complete 3D pose is required for grasping, detection should return the full 3D position and 3D orientation [19]. In context of learning-based grasp detection, typical data-driven approaches differentiate between the utilization of RGB [20], depth (in form of pointclouds [21], [22]) or a combination of both (RGB-D, [23], [24]). In addition, objects to be grasped can be known, similar (i.e., different instance of a known category) or novel, which should be considered when deciding (or developing) on the data representation, collection and training approach [25].

The methods explained generate a grasp pose and require motion planning to execute a grasping action. Such motion planning approaches can be generally listed as motion primitive-based methods, imitation learning and reinforcement learning methods [5], [25].

C. Datasets

Existing datasets for 2D object detection, such as Pascal VOC [26], COCO [27] and, more recently, Objectron [28] for 3D objects, are widely available, including common objects that are present in everyday scenes. There are also datasets designed specifically for grasping such as EGAD! that contains 3D meshes with diverse properties [6] to cover variations in object properties, and datasets that utilize simulation for the grasp data collection, e.g., Jacquard [29] and ACRONYM [7]. These publicly available datasets include different categories of objects enabling a reasonable comparison and performance evaluation of grasping models. However, they are not suitable for applications where the target objects are not included in the dataset, simply because no success rate can be guaranteed.

III. Problem Statement

The robot object grasping scenario considers a robot manipulator with standard gripper and objects that are located on a planar table in front of it (see Fig. 2). Objects of interest are unknown beforehand (e.g., industrial objects) and can have both simple or complex geometry. All objects should allow for a stable grasp, without alteration to the gripper or object pose and be light enough to be lifted (< 1 kg). As general rule, we denote that each object can be represented by a 2D planar position and 1D orientation $\{x, y, \theta\}$, from which a grasp pose is extracted, with rotation and translation defined as $\mathbf{R}(\theta) \in SO(3)$ and $\mathbf{t}(x, y) \in \mathbb{R}^3$. Our observation is that one common grasp model for a selection of objects is difficult to generate. Instead, our approach aims to generate

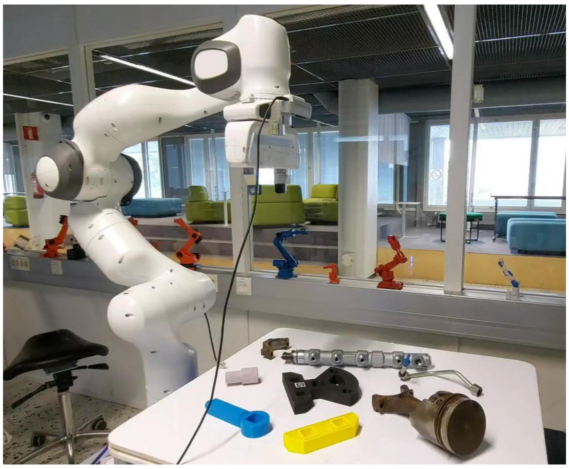


Fig. 2: Object grasping scenario.

a grasp model for individual objects, thereby avoiding modelling conflicts with objects that have different properties. In addition, grasp models need to be generated and deployed fast, without large computational resources, thus restricting the data generation and model training process. This implies that for perception only RGB images are used, with a camera located on the end-effector of the robot.

In summary, the grasping problem can then be stated as follows: from a single image demonstrations of an object annotated with its grasp, generate suitable training data and train a grasp detection model that can run in real-time to successfully grasp the object.

IV. Methodology

Four different grasp detection modules are developed and implemented to find the most robust approach for extracting planar grasp poses. In all cases, the grasping approach consists of the following four distinct steps (Fig. 1):

1) **Human input** - captures and annotates the object in the field of view of the camera.
2) **Training data** - is generated automatically by applying data augmentation techniques.
3) **Object grasp pose** - is estimated based on different state of the art neural networks.
4) **Grasping action** - is done after converting planar grasp to 3D Cartesian pose.

Following, we describe the four different grasp detection modules and their required image annotations. An overview of the models is depicted in Fig. 3 and described in Table I.

A. *Faster R-CNN-based Grasp Detection*

Model A separates the object grasp location and orientation estimation into two different detection models, i.e., Faster R-CNN and CNN, respectively. The Faster R-CNN network takes images and their corresponding annotation as input for training, and generates a bounding box around the objects if they are present in the image scene. The center location of the bounding box is then used as grasp position. In order to predict the grasp orientation, a CNN network is implemented where the final layer consists of 360 output nodes to represent the object's orientation. The first layer of this CNN network accepts image arrays with a size of ($224 \times 224 \times 3$) to extract features and classifies the object based on the highest score to predict the corresponding orientation. Input annotation on the image is done by defining a bounding box around the object. One additional step is required for the orientation, by using the bounding box to crop and resize the region of interest that are labeled with the corresponding orientation.

B. *Keypoint R-CNN-based Grasp Detection by Bounding box*

Model B utilizes the Keypoint R-CNN network to detect both object, represented as a bounding box, and keypoints of an object. For human annotation, a bounding box and two keypoints on the object need to be defined. The keypoints represent the reference orientation for the grasp, from which the augmentation will add ten more keypoints. The estimated bounding box center can then be used as the robot's intermediate hover position, before a relative orientation is extracted from the keypoints to form the grasp pose.

C. *Keypoint R-CNN-based Grasp Detection*

Model C is an improvement of Model B by retrieving the grasp position and orientation directly from the keypoints. Therefore, the same network as in model B is used, and the bounding box information is not utilized. Annotation follows the same approach as Model B, with 12 keypoints used in total for grasp detection.

D. *Mask R-CNN-based grasp detection*

Model D utilizes Mask R-CNN to predict an object mask and returns a planar object position and orientation. This is possible, as the grasping approach only requires 2D object information, based on the object mask that separates the object from the background. One additional step is necessary to determine the grasp orientation, which is done by converting the mask over the object to a binary image followed by local feature extraction methods (SIFT) to estimate the relative orientation. Annotation of the input image requires a bounding box to be drawn around the object and also a set of keypoints to construct a mask/polygon around the object.

E. *Augmented Dataset Generation*

Dataset generation utilizes a single or multiple input images (RGB, $480 \times 640 \times 3$, see Fig. 4a) taken above the workspace in which the target object is visible. These images are then annotated by a person as explained in previous sections and as illustrated in Fig. 4b, depending on their corresponding model input format (see Table I). Then, a sequence of image augmentation techniques are performed to the input annotation, consisting of cropping, zooming, rotation, translation, etc. (see Fig. 4c). For each model, 1500 training samples and 200 validation samples are generated, for position and orientation data, respectively. For the CNN network of model A, the number of generated samples for training an orientation predictor is 5000. More details about each model can be found in Table I.

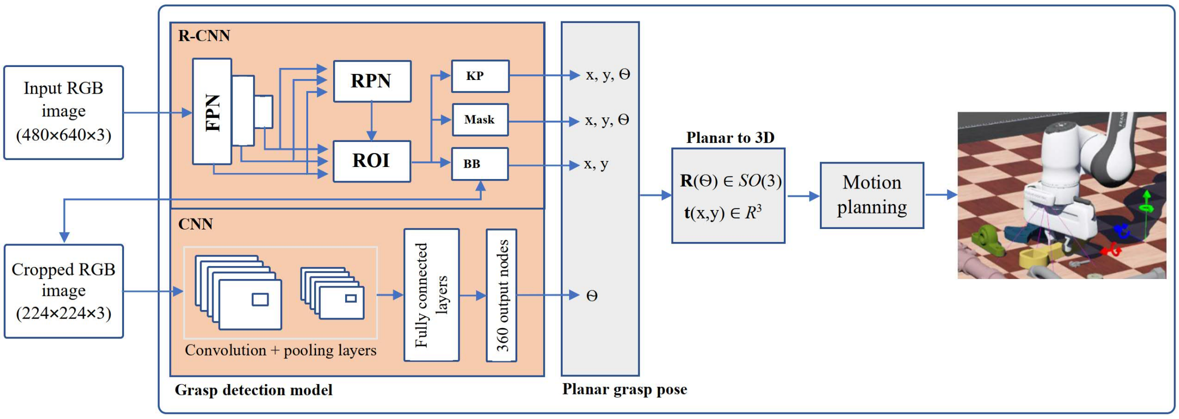


Fig. 3: Overview of the grasp detection approach from a single RGB image demonstration. The grasp detection models utilize the R-CNN architecture family from Detectron2 to generate keypoints, mask and bounding box and extract an object grasp pose. Separately, one CNN network is developed to obtain the object grasp orientation for model A. Abbreviations denote; FPN: Feature Pyramid Network, RPN: Region Proposal Network, ROI: Region Of Interest, KP: Keypoints, BB: bounding box.

TABLE I: Object grasp detection models. Abbreviations: BB - bounding box, KP - keypoints, CL - class label.

	Object position estimation		**Object orientation estimation**			
Grasping model	**Pre-trained model**	**Output format**	**Pre-trained model**	**Method**	**Human input annotation**	**Training data**
A	Faster R-CNN	BB	CNN	Classification	BB	BB, cropped box
B	Keypoint R-CNN	BB and KP	Keypoint R-CNN	KP	BB, 2 KP	BB, KP, CL
C	Keypoint R-CNN	KP	Keypoint R-CNN	KP	2 KP	KP, CL
D	Mask R-CNN	BB and object mask	Mask R-CNN	mask + SIFT	BB, mask	BB, mask, CL

V. Implementation

Implementation of the proposed grasping models include the overall architecture, the grasp detection networks, as well as simulation environment.

A. Architecture

The grasping approach is divided into two major computation nodes, which are explained as follows.

Perception - feeds the neural network models with the input images, runs inference and generates an output. The raw output of the models as explained in Section IV are then used to calculate the planar grasp pose of the object with respect to the image plane. This 2D information is then transformed into 3D coordinates in the world frame and sent to the motion controller node. The transformation from 2D to 3D is done by utilizing a pin-hole camera model, as all the intrinsic and extrinsic parameters of the camera are available. The structure of the perception node is illustrated in Fig. 3.

Motion control - generates the actions and motions of the robot manipulator and gripper in order to execute a grasp. It receives input from the perception node, and directly commands a grasping action. Motion generation is done using ROS MoveIt[1], with a Cartesian position controller running on the robot at 1000 Hz.

B. Grasp Detection Networks

All the models utilize Detectron2 [8] as their perception module. The generated training data and their corresponding labels are fed to the learner, according to the corresponding model input format (see Table I), resulting in one unique grasp detection model for each object. The object grasping approach is implemented in PyTorch, with ROS for communication, and integrated in the OpenDR toolkit [30]. As for the training hyperparameters, object detection for model A utilizes a learning rate of 0.005 and 8 images per batch to train for 500 iterations. For orientation prediction in model A training utilizes a batch size of 32, for 15 epochs with a categorical cross-entropy loss function. Models B, C and D utilize the same hyperparameters, with a learning rate of 0.0008, 2 images per batch for 1000 iterations.

C. Simulation Environment

For fast evaluation of the developed grasp detection models, the entire grasp detection and execution framework has been implemented in robotics simulation. This enables grasping models to be assessed without costly robotic hardware, speeding up developments considerably. For this purpose, 3D models of all objects are included and relevant object and physics parameters can be changed to understand the capabilities and limitations of the grasping models. Webots (see Fig. 5a) is utilized to demonstrate the functionalities, and is freely available to the research community.

[1] `https://moveit.ros.org`

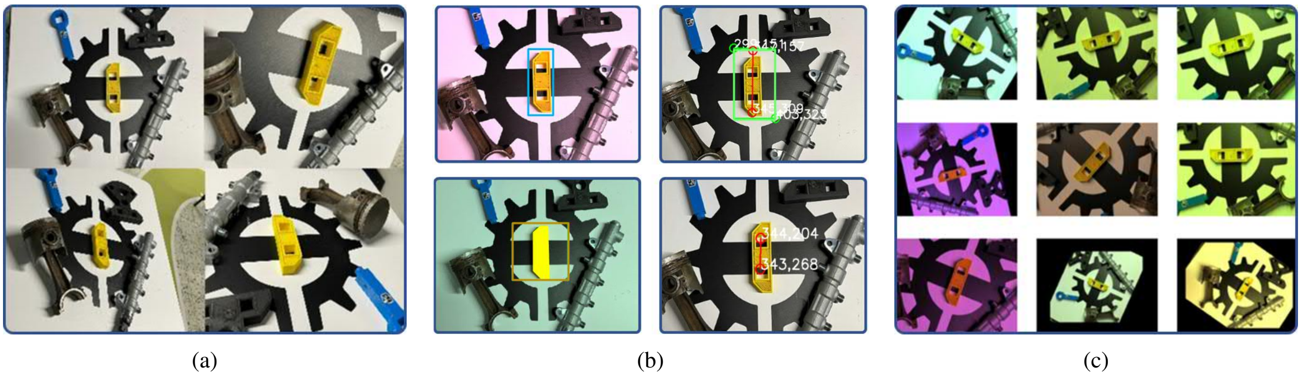


Fig. 4: Training data generation for the planar grasp models. (a) depicts examples of input images for human annotation. (b) depicts the annotations for the four different grasp models (bounding box, keypoints and/or mask, see Table I). (c) depicts augmented images, with variation in brightness, translation, rotation and scale, generated from input and annotated images.

VI. Results and Comparison

Experimental results, their analysis and a comparison to other related work follows in this section.

A. Object Grasping Scenario

To evaluate the performance of the developed models, the grasping scenario is first performed in simulation, after which the best performing model is evaluated by experiments on a real robot. In both cases the scenario includes a collaborative robot (Franka Emika), RGB camera (Intel Realsense D-435) and standard gripper (see Fig. 2). The objects selected for evaluation are depicted in Fig. 5b and include parts from a real Diesel engine and several 3D-printed objects. Variations in object properties are thereby included in terms of mass distribution, texture, symmetry and scale, which is useful as human annotation input determines the object grasp pose. For example, the Diesel engine fuel line (curved pipe) has a small width and a non symmetric shape with even mass distribution, while the Diesel engine piston has a symmetric shape, low aspect ratio and an uneven mass distribution. All objects are placed at random configuration on the table in front of the robot and 10 robot grasp attempts are executed for each object from different robot starting configurations.

B. Grasp Detection Results

Table II lists the performance of each developed model for all objects, expressed as the percentage of successful grasps. For each object 10 grasp detections and grasp attempts are made, therefore, for each model 80 grasp attempts are made in total. While the success rate of all models are within a similar range (i.e., between 78%-93%), some crucial differences can be identified as follows.

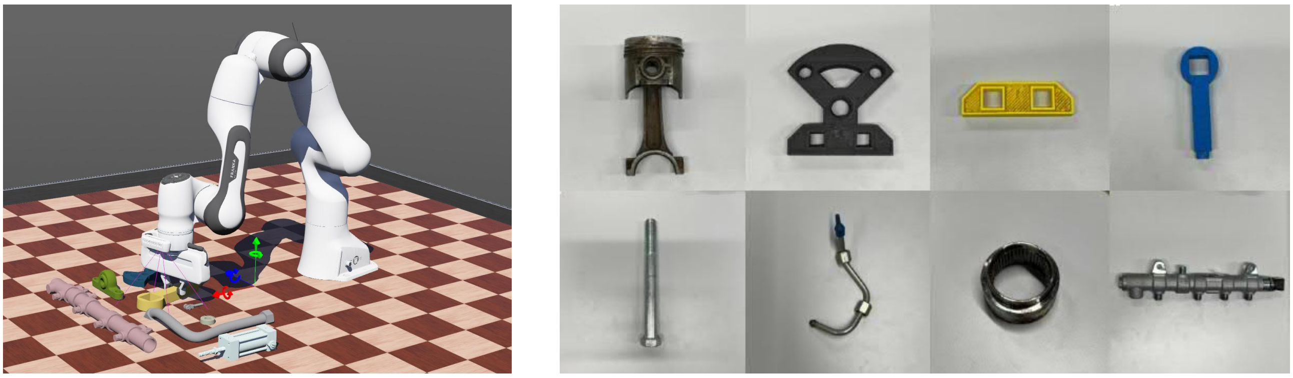
Model A is essentially a two-stage detector, with two separate training datasets and training steps, and, during inference, two consecutive predictions, to estimate the position and orientation of an object grasp. This, unfortunately, makes the model computationally and practically less efficient, compared to the other models. The high success rate is found to be due to a more robust bounding box detection by Faster R-CNN, compared to the keypoint detectors, in terms of the Minimum Area Rectangle (MAR).

Model B and C utilize keypoints for object grasp detection, with the difference that model B utilizes an estimated bounding box for the grasp orientation, while model C extracts this information from the keypoints themselves. While model C demonstrated the highest success rate among all models, keypoints have the limitation that a direct relation between detected keypoints and the actual grasp pose is difficult to realize. We discuss this further in Section VI-D. The high grasp success rate is in this case also achieved by increasing the number of input images and their annotations to seven.

Finally, model D achieved the lowest grasp success rate, partly due to the complexity of detecting the mask of an object. This requires a large number of features on the object, complicating the problem when objects share similarities with each other. In addition, the annotation of an input image is slightly more difficult as the user has to draw a mask/polygon over the object. Fig. 6 depicts several successful object grasping results with model C.

C. Computational Performance

All developed models could be trained to relatively high success rate (i.e., $\approx 80\%$) with a manageable dataset size. This implies around 1500 image samples, leading to a training time below ten minutes. The only exception is model A, where a slightly larger set of images was required for the object grasp orientation estimation and a longer overall training time. All trained models are light-weight (below 0.5GB), meaning they allow for training and real-time execution on a standard GPU (see Table II). In all, with the required data augmentation, dataset size and grasp model training time, it is possible to generate an object grasp model from single or multiple image demonstrations in under 15 minutes. This is very short, compared to other state of the art, e.g., 24hrs in case of [4]. However, it has to be noted that such comparison should take into account crucial differences between each method, such as grasp representation (planar vs. 6DOF) and image format (RGB vs. depth).



(a)

(b)

Fig. 5: Evaluation of the grasp detection models is done in Webots simulation environment (a) and by experiments with real objects (b). Objects used for evaluation include Diesel engine parts and 3D printed objects.

TABLE II: Object grasping results are averaged for eight objects, with 10 attempts per object (80 attempts in total).

	Training			**Inference**		
Model	**Dataset size**	**Training time** hr:min:sec	**Model size**	GTX 1080 Ti (FPS)	GeForce 940mx (FPS)	**Success rate** (%)
A	Faster R-CNN: 1500 CNN: 5000	00:14:00 00:02:00	Faster R-CNN: 300 MB CNN: 8 MB	20	2.5	91
B	1500	00:07:30	450 MB			83
C (simulation)	1500	00:07:00	450 MB			**94**
C (real experiment)	2000	00:08:00	450 MB			**89**
D	1500	00:06:30	330 MB			78

D. Limitations

During the experiments, random objects were placed in the view of the camera to observe the effect of unseen and similar objects to the grasp detection models. Even though this did not pose any major issues (i.e., no false positives), typical challenges in visual detection, such as illumination effects and object overlap, remain. While such effects can be taken into account in the training dataset, this would increase the size of the grasp model. Similarly, since only a single view of an object is used for demonstration, situations might occur where a current viewpoint of the camera does not capture the object well. One solution to this is to include multiple views of an object, each with their individual annotations, which increases the robustness of grasp detection. In our experiments, the results for model C where obtained with seven different views and annotations of the object.

The proposed approach only utilizes RGB images, meaning depth information is not taken into account, as compared to other work [21], [22]. Therefore, the grasping height must be known or estimated prior to a grasp action. This can be solved either by calibrating the camera with respect to the robot and its work area and assuming a fixed grasp height above the table, or by hand-guiding the robot to a desired grasp height. In this work, the former approach was taken.

A further limitation of our approach is the choice of object grasp annotations. For all models, an object grasp is only defined by a bounding box around the object (similar to [18]) and/or several keypoints. In some cases, this does not represent well the grasp pose of an object, for example when a grasp position is not in the center of the bounding box or keypoint set. In such case, the grasp position should be offset by the required distance from the center.

Finally, the planar grasp representation limits the approach to only top grasps with an end-effector pose perpendicular to the table (see Fig. 2). Other end-effector and/or grasp poses, would need to be modelled and integrated separately. One possible solution is to include depth sensing to extract the distance between object and gripper for 6D grasps.

VII. CONCLUSIONS

This work proposed a fast modelling approach for vision-based object grasp detection. Based on a single human object grasp annotation, an augmented dataset of RGB training images is generated, to be utilized for training a grasp detection model. Four different planar grasp detection models, each with different human annotations and grasp detection approach, are evaluated and implemented in simulation. All models are light-weight (below 0.5GB), enabling real-time inference. Best results were obtained with a keypoint-based model, which was further demonstrated with real robot grasping experiments. In all, from a human object grasp annotation, the augmented dataset and grasp model training, the approach enables the generation of a planar object grasp model in under 15 minutes.

ACKNOWLEDGEMENTS

Project funding was received from European Union's Horizon 2020 research and innovation programme, grant no. 871449 (OpenDR) and no. 825196 (TRINITY).

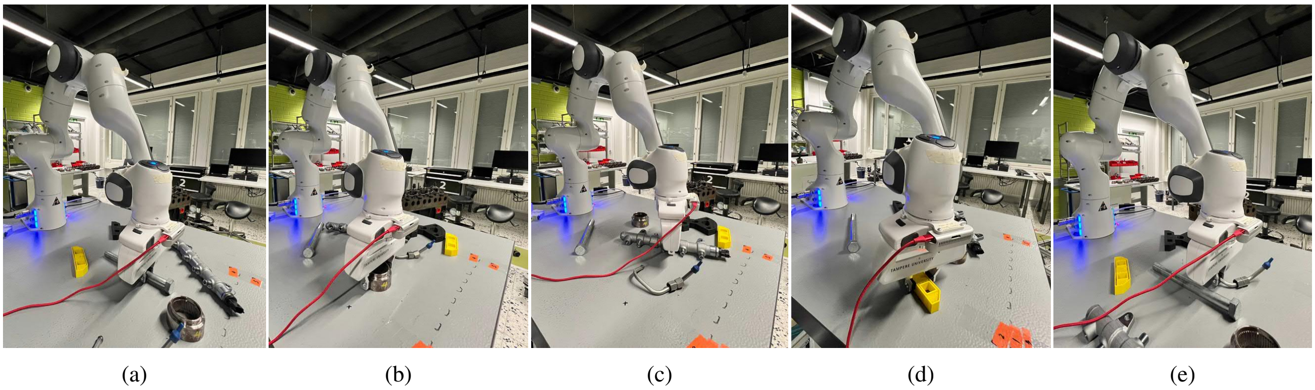

(a) (b) (c) (d) (e)

Fig. 6: Successful object grasping results with model C for different parts: (a) bolt, (b), gear casing, (c) Diesel engine common rail, (d) 3D printed part and (e) Diesel engine fuel line.

References

[1] V. Villani, F. Pini, F. Leali, and C. Secchi, “Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications,” *Mechatronics*, pp. 248–266, 2018.

[2] S. El Zaatari, M. Marei, W. Li, and Z. Usman, “Cobot programming for collaborative industrial tasks: An overview,” *Robotics and Autonomous Systems*, vol. 116, pp. 162–180, 2019.

[3] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, “A survey on learning-based robotic grasping,” *Current Robotics Reports*, pp. 1–11, 2020.

[4] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, “Learning ambidextrous robot grasping policies,” *Science Robotics*, vol. 4, no. 26, 2019.

[5] S. Caldera, A. Rassau, and D. Chai, “Review of deep learning methods in robotic grasp detection,” *Multimodal Technologies and Interaction*, vol. 2, no. 3, p. 57, 2018.

[6] D. Morrison, P. Corke, and J. Leitner, “EGAD! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation,” *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4368–4375, 2020.

[7] C. Eppner, A. Mousavian, and D. Fox, “ACRONYM: A large-scale grasp dataset based on simulation,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6222–6227.

[8] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, “Detectron2,” https://github.com/facebookresearch/detectron2, 2019.

[9] D. G. Lowe, “Object recognition from local scale-invariant features,” in *IEEE international conference on computer vision (ICCV)*, vol. 2, 1999, pp. 1150–1157.

[10] J. Chan, J. Addison Lee, and Q. Kemao, “BIND: Binary integrated net descriptors for texture-less object recognition,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2068–2076.

[11] J. Ma, X. Jiang, A. Fan, J. Jiang, and J. Yan, “Image matching from handcrafted to deep features: A survey,” *International Journal of Computer Vision*, vol. 129, no. 1, pp. 23–79, 2021.

[12] F. Tombari, S. Salti, and L. Di Stefano, “Performance evaluation of 3D keypoint detectors,” *International Journal of Computer Vision*, vol. 102, no. 1-3, pp. 198–220, 2013.

[13] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *International Conference on Neural Information Processing Systems*, 2015, p. 91–99.

[14] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, “3DMatch: Learning local geometric descriptors from RGB-D reconstructions,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1802–1811.

[15] C. Capellen, M. Schwarz, and S. Behnke, “ConvPoseCNN: Dense convolutional 6D object pose estimation,” *arXiv preprint arXiv:1912.07333*, 2019.

[16] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, “Densefusion: 6D object pose estimation by iterative dense fusion,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3343–3352.

[17] Y. Jiang, S. Moseson, and A. Saxena, “Efficient grasping from RGBD images: Learning using a new rectangle representation,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3304–3311.

[18] E. De Coninck, T. Verbelen, P. Van Molle, P. Simoens, and B. Dhoedt, “Learning robots to grasp by demonstration,” *Robotics and Autonomous Systems*, vol. 127, p. 103474, 2020.

[19] A. Mousavian, C. Eppner, and D. Fox, “6-DOF graspnet: Variational grasp generation for object manipulation,” in *IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 2901–2910.

[20] H. Zhang, X. Lan, X. Zhou, Z. Tian, Y. Zhang, and N. Zheng, “Visual manipulation relationship network for autonomous robotics,” in *IEEE International Conference on Humanoid Robots (Humanoids)*, 2018, pp. 118–125.

[21] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, “Grasp pose detection in point clouds,” *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473, 2017.

[22] B. Zhao, H. Zhang, X. Lan, H. Wang, Z. Tian, and N. Zheng, “REGNet: Region-based grasp network for single-shot grasp detection in point clouds,” *arXiv preprint arXiv:2002.12647*, 2020.

[23] H.-S. Fang, C. Wang, M. Gou, and C. Lu, “Graspnet-1billion: A large-scale benchmark for general object grasping,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 444–11 453.

[24] P. Wang, F. Manhardt, L. Minciullo, L. Garattoni, S. Meier, N. Navab, and B. Busam, “DemoGrasp: Few-shot learning for robotic grasping with human demonstration,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 5733–5740.

[25] G. Du, K. Wang, S. Lian, and K. Zhao, “Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review,” *Artificial Intelligence Review*, pp. 1–58, 2020.

[26] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (VOC) challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[27] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common objects in context,” in *IEEE European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 740–755.

[28] A. Ahmadyan, L. Zhang, A. Ablavatski, J. Wei, and M. Grundmann, “Objectron: A large scale dataset of object-centric videos in the wild with pose annotations,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 7822–7831.

[29] A. Depierre, E. Dellandréa, and L. Chen, “Jacquard: A large scale dataset for robotic grasp detection,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3511–3516.

[30] N. Passalis *et al.*, “OpenDR: An open toolkit for enabling high performance, low footprint deep learning for robotics,” *arXiv preprint arXiv:2203.00403*, 2022.

G

Adaptively Calibrated Critic Estimates for Deep Reinforcement Learning

Nicolai Dorka[1] and Tim Welschehold[1] and Joschka Bödecker[1] and Wolfram Burgard[2]

***Abstract*— Accurate value estimates are important for off-policy reinforcement learning. Algorithms based on temporal difference learning typically are prone to an over- or underestimation bias building up over time. In this paper, we propose a general method called Adaptively Calibrated Critics (ACC) that uses the most recent high variance but unbiased on-policy rollouts to alleviate the bias of the low variance temporal difference targets. We apply ACC to Truncated Quantile Critics [1], which is an algorithm for continuous control that allows regulation of the bias with a hyperparameter tuned per environment. The resulting algorithm adaptively adjusts the parameter during training rendering hyperparameter search unnecessary and sets a new state of the art on the OpenAI gym continuous control benchmark among all algorithms that do not tune hyperparameters for each environment. ACC further achieves improved results on different tasks from the Meta-World robot benchmark. Additionally, we demonstrate the generality of ACC by applying it to TD3 [2] and showing an improved performance also in this setting.**

I. Introduction

Off-policy reinforcement learning is an important research direction as the reuse of old experience promises to make these methods more sample efficient than their on-policy counterparts. This is an important property for many applications such as robotics where interactions with the environment are very time- and cost-intensive. Many successful off-policy methods make use of a learned Q-value function [2], [3], [4], [5]. If the action space is discrete the Q-function can be directly used to generate actions while for continuous action spaces it is usually used in an actor-critic setting where the policy is trained to choose actions that maximize the Q-function. In both cases accurate estimates of the Q-values are of crucial importance.

Unfortunately, learning the Q-function off-policy can lead to an overestimation bias [6]. Especially when a nonlinear function approximator is used to model the Q-function, there are many potential sources of bias. Different heuristics were proposed for their mitigation, such as the double estimator in the case of discrete action spaces [7] or taking the minimum of two estimates in the case of continuous actions [2]. While these methods successfully prevent extreme overestimation, due to their coarse nature, they can still induce under- or overestimation bias to a varying degree depending on the environment [8].

To overcome these problems we propose a principled and general method to alleviate the bias called Adaptively Calibrated Critics (ACC). Our algorithm uses the most recent on-policy rollouts to determine the current bias of the Q-estimates and adjusts a bias controlling parameter accordingly. This parameter adapts the size of the temporal difference (TD) targets such that the bias can be corrected in the subsequent updates. As the parameter changes slower than the rollout returns, our method still benefits from stable and low-variance temporal difference targets, while it incorporates the information from unbiased but high variance samples from the recent policy to reduce the bias.

We apply ACC to Truncated Quantile Critics (TQC) [1], which is a recent off-policy actor-critic algorithm for continuous control showing strong performance on various tasks. In TQC the bias can be controlled in a finegrained way with the help of a hyperparameter that has to be tuned for every environment. ACC allows to automatically adjusts this parameter online during the training in the environment. As a result, it eliminates the need to tune this hyperparameter in a new environment, which is very expensive or even infeasible for many applications.

We evaluate our algorithm on a range of continuous control tasks from OpenAI gym [9] and robotic tasks from the meta world benchmark [10] and exceed the current state-of-the-art results among all algorithms that do not need tuning of environment-specific hyperparameters. For each environment, ACC matches the performance of TQC with the optimal hyperparameter for that environment. Further, we show that the automatic bias correction allows to increase the number of value function updates performed per environment step, which results in even larger performance gains in the sample-efficient regime. We additionally apply ACC to the TD3 algorithm [2] where it also leads to notably improved performance, underscoring the generality of our proposed method. To summarize, the main contributions of this work are:

1) We propose Adaptively Calibrated Critics, a new general algorithm that reduces the bias of value estimates in a principled fashion with the help of the most recent unbiased on-policy rollouts.
2) As a practical implementation we describe how ACC can be applied to learn a bias-controlling hyperparameter of the TQC algorithm and show that the resulting algorithm sets a new state of the art on the OpenAI continuous control benchmark suite.
3) ACC achieves strong performance on robotics tasks.
4) We demonstrate that ACC is a general algorithm with respect to the adjusted parameter by additionally applying it successfully to TD3.

To allow for reproducibility of our results we describe our

Authors are with: [1]University of Freiburg and [2] University of Technology Nuremberg, Germany. dorka@cs.uni-freiburg.de.

This work was supported by the European Union's Horizon 2020 research and innovation program under grant agreement No 871449-OpenDR.

algorithm in detail, report all hyperparameters, use a large number of random seeds for evaluation, and made the source code publicly available[1].

II. Background

We consider model-free reinforcement learning for episodic tasks with continuous state and action spaces $\mathcal{S}$ and $\mathcal{A}$. An agent interacts with its environment by selecting an action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$ for every discrete time step t. The agent receives a scalar reward r_t and transitions to a new state s_{t+1}. To model this in a mathematical framework we use a Markov decision process, defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. Given an action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$ the unknown state transition density $\mathcal{P}$ defines a distribution over the next state. Rewards come from the reward function $\mathcal{R}$ and future rewards are discounted via the discount factor $\gamma \in [0, 1]$.

The goal is to learn a policy π mapping a state s to a distribution over actions such that the sum of future discounted rewards $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$ is maximized. We use the term π_ϕ for the policy with parameters ϕ trained to maximize the expected return $J(\phi) = \mathbb{E}_{s_i \sim \mathcal{P}, a_i \sim \pi}[R_0]$. The value function for a given state-action pair (s, a) is defined as $Q^\pi(s, a) = \mathbb{E}_{s_i \sim \mathcal{P}, a_i \sim \pi}[R_t | s, a]$, which is the expected return when executing action a in state s and following π afterwards.

A. Soft Actor Critic

TQC extends Soft Actor-Critic (SAC) [3], which is a strong off-policy algorithm for continuous control using entropy regularization. While in the end we are interested in maximizing the performance with respect to the total amount of reward collected in the environment, SAC maximizes for an auxiliary objective that augments the original reward with the entropy of the policy $J(\phi) = \mathbb{E}_{s_t \sim \mathcal{P}, a_t \sim \pi}[\sum_t \gamma^t (r_t + \alpha \mathcal{H}(\pi(\cdot|s_t)))]$, where $\mathcal{H}$ denotes the entropy.

A critic is learned that evaluates the policy π in terms of its Q-value of the entropy augmented reward. The policy—called actor—is trained to choose actions such that the Q-function is maximized with an additional entropy regularization

$$J_\pi(\phi) = \mathop{\mathbb{E}}_{s_t \sim \mathcal{D}, a_t \sim \pi_\phi} [Q_\theta(s_t, a_t) - \alpha \log \pi_\phi(a_t|s_t)]. \quad (1)$$

The weighting parameter α of the entropy term can be automatically adjusted during the training [11]. Both the training of actor and critic happen off-policy with transitions sampled from a replay buffer.

B. Truncated Quantile Critics

The TQC algorithm uses distributional reinforcement learning [12] to learn a distribution over the future augmented reward instead of a Q-function which is a point estimate for the expectation of this quantity. To do so TQC utilizes quantile regression [13] to approximate the distribution with Dirac delta functions $Z_\theta(s_t, a_t) = \frac{1}{M} \sum_{m=1}^{M} \delta(\theta^m(s_t, a_t))$. The Diracs are located at the quantile locations for fractions $\tau_m = \frac{2m-1}{m}, m \in \{1, \ldots, M\}$. The network is trained to learn the quantile locations $\theta^m(s, a)$ by regressing the predictions $\theta^m(s_t, a_t)$ onto the Bellman targets $y_m(s_t, a_t) = r_t + \gamma(\theta^m(s_{t+1}, a_{t+1}) - \alpha \log \pi_\phi(a_{t+1}|s_{t+1}))$ via the Huber quantile loss.

TQC uses an ensemble of N networks $(\theta_1, \cdots, \theta_N)$ where each network θ_n predicts the distribution $Z_{\theta_n}(s_t, a_t) = \frac{1}{M} \sum_{m=1}^{M} \delta(\theta_n^m(s_t, a_t))$. A single Bellman target distribution is computed for all networks. This happens by first computing all targets for all networks, pooling all targets together in one set and sorting them in ascending order. Let $k \in \{1, \ldots, M\}$, then the kN smallest of these targets y_i are used to define the target distribution $Y(s_t, a_t) = \frac{1}{kN} \sum_{i=1}^{kN} \delta(y_i(s_t, a_t))$. The networks are trained by minimizing the quantile Huber loss which in this case is given by

$$L(s_t, a_t; \theta_n) = \frac{1}{kNM} \sum_{m,i=1}^{M,kN} \rho_{\tau_m}^H (y_i(s_t, a_t) - \theta_n^m(s_t, a_t)) \quad (2)$$

where $\rho_\tau^H(u) = |\tau - \mathbf{1}(u < 0)| \mathcal{L}_H^1(u)$ and $\mathcal{L}_H^1(u)$ is the Huber loss with parameter 1.

The rationale behind truncating some quantiles from the target distribution is to prevent overestimation bias. In TQC the number of dropped targets per network $d = M - k$ is a hyperparameter that has to be tuned per environment but allows for a finegrained control of the bias.

The policy is trained as in SAC by maximizing the entropy penalized estimate of the Q-value which is the expectation over the distribution obtained from the critic

$$J(\phi) = \mathop{\mathbb{E}}_{\substack{s \sim \mathcal{D} \\ a \sim \pi}} \left[\frac{1}{NM} \sum_{m,n=1}^{M,N} \theta_n^m(s, a) - \alpha \log \pi_\phi(a|s) \right]. \quad (3)$$

III. Adaptively Calibrated Critics

In this section, we will introduce the problem of estimation bias in TD learning, present our method ACC and demonstrate how it can be applied to TQC.

A. Over- and Underestimation Bias

The problem of overestimation bias in temporal difference learning with function approximation has been known for a long time [6]. In Q-learning [14] the predicted Q-value $Q(s_t, a_t)$ is regressed onto the target given by $y = r_t + \gamma \max_a Q(s_{t+1}, a)$. In the tabular case and under mild assumptions the Q-values converge to that of the optimal policy [14] with this update rule. However, using a function approximator to generate the Q-value introduces an approximation error. Even under the assumption of zero mean noise corruption of the Q-value $\mathbb{E}[\epsilon_a] = 0$, an overestimation bias occurs in the computation of the target value because of Jensen's inequality

$$\begin{aligned} \max_a Q(s_{t+1}, a) &= \max_a \mathbb{E}[Q(s_{t+1}, a) + \epsilon_a] \\ &\leq \mathbb{E}\left[\max_a \{Q(s_{t+1}, a) + \epsilon_a\}\right]. \quad (4) \end{aligned}$$

In continuous action spaces it is impossible to take the maximum over all actions. The most successful algorithms rely on an actor-critic structure where the actor is trained to choose actions that maximize the Q-value [2], [3], [15]. So the actor can be interpreted an approximation to the argmax of the Q-value.

With deep neural networks as function approximators other problems such as over-generalization [5], [16] can occur

[1] https://github.com/Nicolinho/ACC

where the updates to $Q(s_t, a_t)$ also increases the target through $Q(s_{t+1}, a)$ for all a which could lead to divergence. There are many other potential sources for overestimation bias such as stochasticity of the environment [17] or computing the Q-target from actions that lie outside of the current training data distribution [18].

While for discrete action spaces the overestimation can be controlled with the double estimator [7], [17], it was shown that this estimator does not prevent overestimation when the action space is continuous [2]. As a solution the TD3 algorithm [2] uses the minimum of two separate estimators to compute the critic target. This approach was shown to prevent overestimation but can introduce an underestimation bias. In TQC [1] the problem is handled by dropping some targets from the pooled set of all targets of an ensemble of distributional critics. This allows for more finegrained control of over- or underestimation by choosing how many targets are dropped. TQC is able to achieve an impressive performance but the parameter d determining the number of dropped targets has to be set for each environment individually. This is highly undesirable for many applications since the hyperparameter sweep to determine a good choice of the parameter increases the actual number of environment interactions proportional to the number of hyperparameters tested. For many applications like robotics this makes the training prohibitively expensive.

B. Dynamically Adjusting the Bias

In the following we present a new general approach to adaptively control bias emerging in TD targets regardless of the source of the bias. Let $R^\pi(s,a)$ be the random variable denoting the sum of future discounted rewards when the agent starts in state s, executes action a and follows policy π afterwards. This means that the Q-value is defined as its expectation $Q^\pi(s,a) = \mathbb{E}[R^\pi(s,a)]$. For notational convenience we will drop the dependency on the policy π in the following. We start with the tabular case. Suppose for each state-action pair (s,a) we have a family $\{\hat{Q}_\beta(s,a)\}_{\beta\in[\beta_{min},\beta_{max}]\subset\mathbb{R}}$ of estimators for $Q(s,a)$ with the property that $\hat{Q}_{\beta_{min}}(s,a) \leq Q(s,a) \leq \hat{Q}_{\beta_{max}}(s,a)$, where $Q(s,a)$ is the true Q-value of the policy π and Q_β a continuous monotone increasing function in β .

If we have samples $R_i(s,a)$ of the discounted returns $R(s,a)$, an unbiased estimator for $Q(s,a)$ is given by the average of the R_i through Monte Carlo estimation [19]. We further define the estimator $\hat{Q}_{\beta^*}(s,a)$, where β^* is given by

$$\beta^*(s,a) = \underset{\beta\in[\beta_{min},\beta_{max}]}{\arg\min} \left| \hat{Q}_\beta(s,a) - \frac{1}{N}\sum_{i=1}^{N} R_i(s,a) \right|. \quad (5)$$

The following Theorem, which we prove in the appendix, shows that the estimator is unbiased under some assumptions.

Theorem 1: Let $Q_\beta(s,a)$ be a continuous monotone increasing function in β and assume that for all (s,a) it holds $\hat{Q}_{\beta_{min}}(s,a) \leq Q(s,a) \leq \hat{Q}_{\beta_{max}}(s,a)$, the returns $R(s,a)$ follow a symmetric probability distribution and that $\hat{Q}_{\beta_{min}}(s,a)$ and $\hat{Q}_{\beta_{max}}(s,a)$ have the same distance to $Q(s,a)$. Then Q_{β^*} from Equation 5 is an unbiased estimator for the true value Q for all (s,a).

Algorithm 1 ACC - General

```
Initialize: bias controlling parameter $\beta$, steps between $\beta$ updates $T_\beta$, $t_\beta = 0$
for $t = 1$ to total number of environment steps do
    Interact with environment according to $\pi$, store transitions in replay buffer $\mathcal{B}$ and store observed returns $R(s,a)$, increment $t_\beta$ += 1
    if episode ended and $t_\beta >= T_\beta$ then
        Update $\beta$ with Eq. 6 using the most recent experience and set $t_\beta = 0$
    end if
    Sample mini-batch $b$ from $\mathcal{B}$
    Update $Q$ with target computed from $Q_\beta$ and $b$
end for
```

The symmetry and same distance assumption can be replaced by assuming that $\hat{Q}_{\beta_{min}}(s,a) \leq R_i \leq \hat{Q}_{\beta_{max}}(s,a)$ with probability one. In this case the proof is straightforward since Q_β can take any value for which R_i has positive mass.

We are interested in the case where $\hat{Q}$ is given by a function approximator such that there is generalization between state-action pairs and that it is possible to generate estimates for pairs for which there are no samples of the return available. Consider off-policy TD learning where the samples for updates of the Q-function are sampled from a replay buffer of past experience. While the above assumptions might not hold anymore in this case, we have an estimator for all state-action pairs and not just the ones for which we have samples of the return. Also in practice rolling out the policy several times from each state action pair is undesirable and so we set $N = 1$ which allows the use of the actual exploration rollouts. Our proposed algorithm starts by initializing the bias-controlling parameter β to some value. After a number of environment steps and when the next episode is finished, the Q-value estimates and actual observed returns are compared. Depending on the difference β is adjusted according to

$$\beta_{new} = \beta_{old} + \alpha \sum_{t=1}^{T_\beta} \Big[R(s_t,a_t) - \hat{Q}(s_t,a_t) \Big], \quad (6)$$

where α is a step size parameter and $(s_t,a_t)_{t=1}^{T_\beta}$ are the $T_\beta \in \mathbb{N}$ most recent state-action pairs. As a result β is decreased in the case of overestimation, where the Q-estimates are larger than the actual observed returns, and increased in the case of underestimation. We assumed that Q_β is continuous and monotonically increasing in β. Hence, increasing β increases Q_β and vice versa. For updating the Q-function the target will be computed from Q_β.

Only performing one update step and not the complete minimization from Equation 5 has the advantage that β is changing relatively slow which means the targets are more stable. Through this mechanism our method can incorporate the high variance on-policy samples to correct for under- or overestimation bias. At the same time our method can benefit from the low variance TD targets. ACC in its general form is summarized in Algorithm 1.

Other algorithms that attempt to control the bias arising in

TD learning with non-linear function approximators usually use some kind of heuristic that includes more than one estimator. Some approaches use them to decouple the choice of the maximizing action and the evaluation of the maximum in the computation of the TD targets [7]. Alternative approaches take the minimum, maximum or a combination of both over the different estimators [2], [8], [20], [21]. All of these have in common that the same level of bias correction is done for every environment and for all time steps during training. In the deep case there are many different sources that can influence the tendency of TD learning building up bias in non-trivial ways. ACC is more principled in the regard that it allows to dynamically adjust the magnitude and direction of bias correction during training. Regardless of the source and amount of bias ACC provides a way to alleviate it. This makes ACC promising to work robustly on a wide range of different environments.

One assumption of ACC is that there is a way to adjust the estimated Q-value with a parameter β such that $\hat{Q}_\beta$ is continuous and monotonically increasing in β. There are many different functions that are in accordance with this assumption. We give one general example of how such a $\hat{Q}_\beta$ can be easily constructed for any algorithm that learns a Q-value. Let $\hat{Q}$ be the current estimate. Then define $\hat{Q}_\beta = \beta|\hat{Q}|/K + \hat{Q}$, where K is a constant (e.g. 100) and $[\beta_{min}, \beta_{max}]$ is some interval around 0. In the following section we will present an application of ACC in a more sophisticated way.

C. Applying ACC to TQC

As a practical instantiation of the general ACC algorithm we apply it to adjust the number of targets dropped from the set of all targets in TQC. Denote with $d_{max} \in \{0, \dots, M\}$ some upper limit of targets to drop per network. Define $\beta_{min} = 0$, $\beta_{max} = d_{max}$ and let $d = d_{max} - \beta$ be the current number of targets dropped for each network. Further, we write Q_β for the TQC estimate with dN targets dropped from the pooled set of all targets. If d_{max} is set high enough the TQC estimate without dropped targets $Q_{\beta_{max}}$ induces overestimation while the TQC estimate with d_{max} dropped targets per net $Q_{\beta_{min}}$ induces underestimation.

In general, $\beta \in [0, d_{max}]$ is continuous and hence also d is a continuous value. As the number of dropped targets from the pooled set of all targets has to be a discrete number in $\{0, \dots, NM\}$ we round the total number of dropped targets dN to the nearest integer in the computation of the TD target. When updating β with Equation 6, we divide the expectation by the moving average of the absolute value of the difference between returns and estimated Q-values for normalization.

IV. Experiments

We evaluate our algorithm on a range of continuous control tasks from OpenAI Gym [9] and the meta world benchmark [10] that both use the physics engine MuJoCo [22] (version 1.5). First, we benchmark ACC against strong methods that do not use environment specific hyerparameters. Then we compare the performance of TQC with a fixed number of dropped targets per network with that of ACC. Next, we evaluate the effect of more critic updates for ACC and show results in the sample efficient regime. Further, we study the effect of ACC on the accuracy of the value estimate, and investigate the generality of ACC by applying it to TD3.

We implemented ACC on top of the PyTorch code published by the authors[2] to ensure a fair comparison. While in general a safe strategy is to use a very high value for d_{max} as it gives ACC more flexibility in choosing the right amount of bias correction we set it to $d_{max} = 5$, which is the maximum value used by TQC for the number of dropped targets in the original publication. At the beginning of the training we initialize $\beta = 2.5$ and set the step size parameter to $\alpha = 0.1$. After $T_\beta = 1000$ steps since the last update and when the next episode finishes, β is updated with a batch that stores the most recent state-action pairs encountered in the environment and their corresponding observed discounted returns. After every update of β the oldest episodes in this stored batch are removed until there are no more than 5000 state-action pairs left. This means that on average β is updated with a batch whose size is a bit over 5000. The updates of β are started after 25000 environment steps and the moving average parameter in the normalization of the $\beta-$update is set to 0.05. The first 5000 environment interactions are generated with a random policy after which learning starts. We did not tune most of these additional hyperparameters and some choices are directly motivated by the environment (e.g. setting T_β to the maximum episode length). Only for α we tested a few different choices but found that for reasonable values it does not have a noticeable influence on performance. All hyperparameters of the underlying TQC algorithm with $N = 5$ critic networks were left unchanged.

Compared to TQC the additional computational overhead caused by ACC is minimal because there is only one update to β that is very cheap compared to one training step of the actor-critic and there are at least $T_\beta = 1000$ training steps in between one update to β.

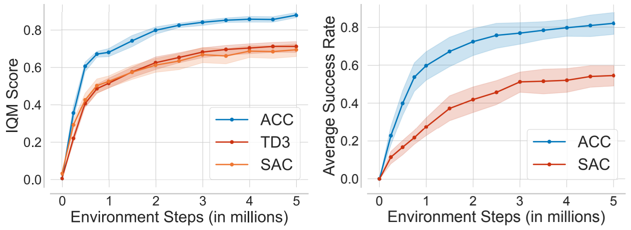
During training, the policy is evaluated every 1,000 environment steps by averaging the episode returns of 10 rollouts with the current policy. For each task and algorithm we run 10 trials each with a different random seed.

A. Comparative Evaluation

We compare ACC to the state of the art continuous control methods SAC [3] (with learned temperature parameter [11]) and TD3 [2] on six OpenAI Gym continuous control environments. To make the different environments comparable we normalize the scores by dividing the achieved return by the best achieved return among all evaluations points of all algorithms for that environment.

Figure 1a) shows the aggregated data efficiency curve over all 6 tasks computed with the method of [23], where the interquantile mean (IQM) ignores the bottom and top 25% of the runs across all games and computes the mean over the remaining. The absolute performance of ACC for each single task can be seen in Figure 2. Overall, ACC reaches a much higer performance than SAC and TD3.

[2] `https://github.com/bayesgroup/tqc_pytorch`



a) b)

Fig. 1. Sample efficiency curves aggregated from the results over several environments. The normalized IQM score and the mean of the success rate respectively is plotted against the number of environment steps. Shaded regions denote pointwise 95% stratified bootstrap confidence intervals according to the method of [23]. **(a)** Aggregated results over the 6 gym continuous control tasks. **(b)** Aggregated results over the 12 metaworld tasks.

B. Robotics Benchmark

To investigate, if ACCs strong performance also translates into robotics environments, we evaluate ACC and SAC on 12 of the more challenging tasks in the Meta-World benchmark [10], which consists of several manipulation tasks with a Sawyer arm. We use version V2 and use the following 12 tasks: sweep, stick-pull, dial-turn, door-open, peg-insert-side, push, pick-out-of-hole, push-wall, faucet-open, hammer, stick-push, soccer. We evaluate the single tasks in the in the MT1 version of the benchmark, where the goal and object positions change across episodes. Different to the gym environments, β is updated every 500 environment steps as this is the episode length for these tasks. Figure 1b) shows the aggregated data efficiency curve in terms of success rate over all 12 tasks computed with the method of [23].

The curves demonstrate that ACC achieves drastically stronger results than SAC both in terms of data efficiency and asymptotic performance. After 2 million steps ACC already achieves a close to optimal task success rate which is even considerably higher than what SAC achieves at the end of the training. This shows, that ACC is a promising approach for real world robotics applications.

C. Fixing the Number of Dropped Targets

In this experiment we evaluate how well ACC performs when compared to TQC where the number of dropped targets per network d is fixed to some value. Since in the original publication for each environment the optimal value was one of the three values 0, 2, and 5, we evaluated TQC with d fixed to one of these values for each environment. To ensure comparability we used the same codebase as for ACC. The results in Figure 2 show that it is not possible to find one value for d that performs well on all environments. With $d = 0$, TQC is substantially worse on three environments and unstable on the *Ant* environment. Setting $d = 2$ is overall the best choice but still performs clearly worse for two environments and is also slightly worse for *Humanoid*. Dropping $d = 5$ targets per network leads to an algorithm that can compete with ACC only on two of the six environments. Furthermore, even if there would be one tuned parameter that performs equally well as ACC on a given set of environments we hypothesize there are likely very different environments for which the specific parameter choice will not perform well.

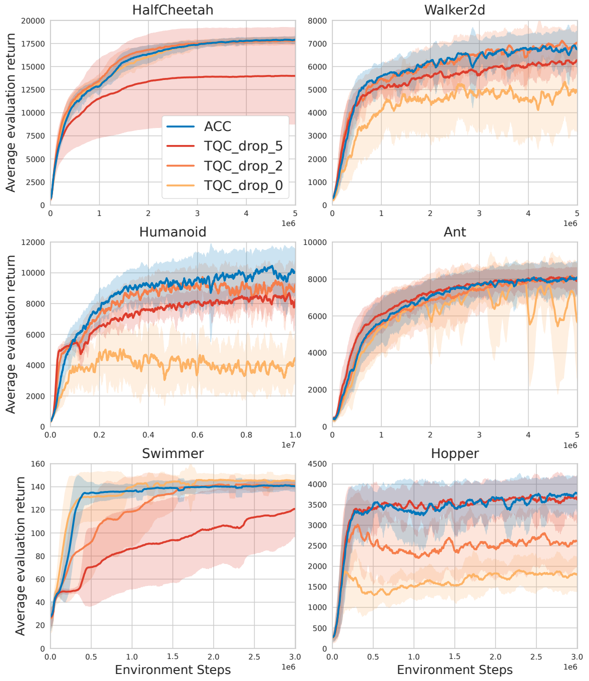


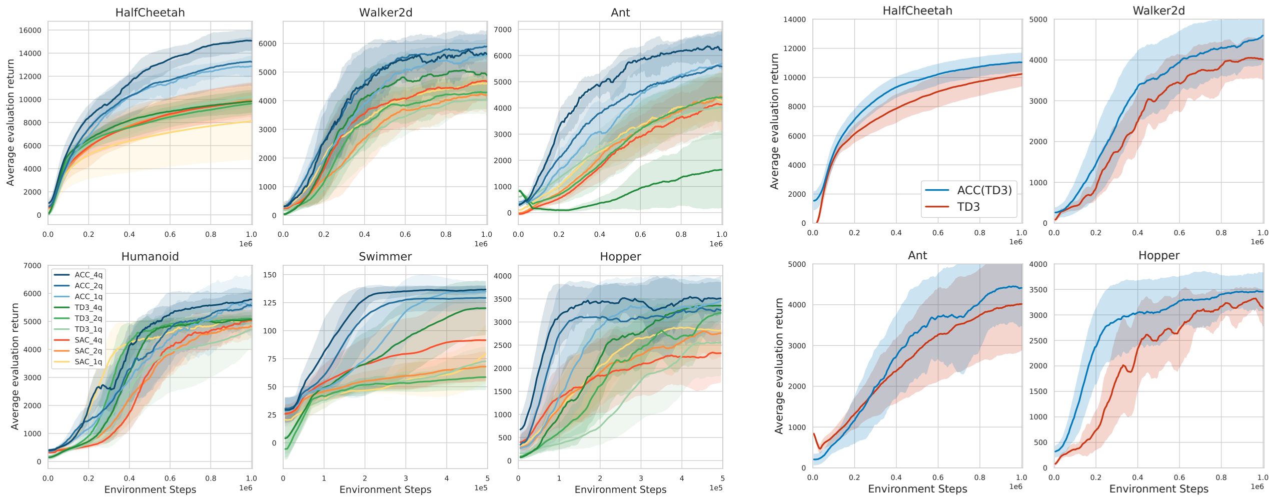
Fig. 2. Learning curves of ACC applied to TQC and TQC with different fixed choices for the number of dropped atoms d on six OpenAi gym environments. We used version *v3*. The shaded area represents mean ± standard deviation over the 10 trials. For readability the curves showing the mean are filtered with a uniform filter of size 15.

The principled nature of ACC on the other hand provides reason to believe that it can perform robustly on a wide range of different environments. This is supported by the robust performance on all considered environments.

D. Evaluation of Sample Efficient Variant

In principle more critic updates per environment step should make learning faster. However, because of the bootstrapping in the target computation this can easily become unstable. The problem is that as targets are changing faster, bias can build up easier and divergence becomes more likely. ACC provides a way to detect upbuilding bias in the TD targets and to correct the bias accordingly. This motivates to increase the number of gradient updates of the critic. In TD3, SAC and TQC one critic update is performed per environment step. We conducted an experiment to study the effect of increasing this rate up to 4. ACC using 4, 2 and 1 updates are denoted with ACC_4q, ACC_2q and ACC_1q respectively. ACC_1q is equal to ACC from the previous experiments. We use the same notation also for TD3 and SAC.

Scaling the number of critic updates by a factor of 4 increases the computation time by a factor of 4. But this can be worthwhile in the sample efficient regime, where a huge number of environment interactions is not possible or the interaction cost dominate the computational costs as it is the case when training robots in the real world. The results in Figure 3a) show that in the sample efficient regime ACC4q further increases over plain ACC. ACC4q reaches the final performance of TD3 and SAC in less than a third of the number of steps for five environments and for



a) b)

Fig. 3. The mean ± standard deviation over 10 trials. **(a)** Results in the sample efficient regime where tuning of hyperparameters in an inner loop is too costly with different choices for the number of value function updates per environment step. **(b)** Results for ACC applied to TD3 compared to pure TD3.

Humanoid in roughly half the number of steps. Increasing the number of critic updates for TD3 and SAC shows mixed results, increasing performance for some environments while decreasing it for others. Only ACC benefits from more updates on all environments, which supports the hypothesis that ACC is successful at calibrating the critic estimate.

E. Analysis of ACC

To evaluate the effect of ACC on the bias of the value estimate, we analyze the difference between the value estimate and the corresponding observed return when ACC is applied to TQC. For each state-action pair encountered during exploration, we compute its value estimate at that time and at the end of the episode compare it with the actual discounted return from that state onwards. Hence, the state-action pair was not used to update the value function at the point when the value estimate has been computed. If an episode ends because the maximum number of episode time-steps has been reached, which is 1,000 for the considered environments, we ignore the last 100 state-action pairs. The reason is that in TQC the value estimator is trained to ignore the episode timeout and uses a bootstrapped target also at the end of the episode. We normalize for different value scales by computing the absolute error between the value estimate and the observed discounted return and divide that by the absolute value of the discounted return. Every 1,000 steps, the average over the errors of the last 1,000 state-action pairs is computed. The aggregated results in Figure 4b) show that averaged over all environments ACC indeed achieves a lower value error than TQC with the a fixed number of dropped atoms d. This supports our hypothesis that the strong performance of ACC applied to TQC indeed stems from better values estimates.

To better understand the hidden training dynamics of ACC we show in Figure 4a) how the number of dropped targets per network $d = d_{max} - \beta$ evolves during training. Interestingly, the relatively low standard deviation indicates a similar behaviour across runs for a specific environment. However, there are large differences between the environments which indicates that it might not be possible to find a single hyperparameter that works well on a wide variety of different environments. Further, the experiments shows that the optimal amount of overestimation correction might change over time during the training even on a single environment.

F. Beyond TQC: Improving TD3 with ACC

To demonstrate the generality of ACC, we additionally applied it to the actor-critic style TD3 algorithm [2], which uses two critics. These are initialized differently but trained with the same target value, which is the minimum over the two targets computed from the two critics. While this successfully prevents the overestimation bias, using the minimum of the two target estimates is very coarse and can instead lead to an underestimation bias. We applied ACC to TD3 by defining the target for each critic network to be a convex combination between its own target and the minimum over both targets. Let $Q_i = Q_{\bar{\theta}_i}(s_{t+1}, \pi_{\bar{\phi}}(s_{t+1}))$, we define the k-th critic target

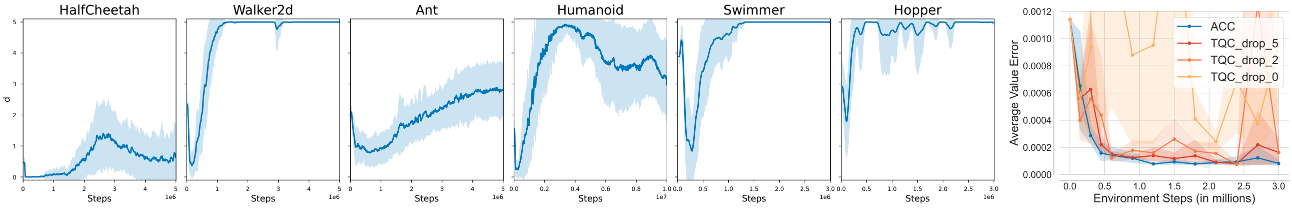
$$y_k = r + \gamma\Big(\beta\; Q_k + (1-\beta)\min_{i=1,2} Q_i\Big),$$

where $\beta \in [0,1]$ is the ACC parameter that is adjusted to balance between under- and overestimation. The results are displayed in Figure 3b) and show that ACC also improves the performance of TD3.

V. Related Work

A. Overestimation in Reinforcement Learning

The problem of overestimation in Q-learning with function approximation was introduced by [6]. For discrete actions the double estimator has been proposed [17] where two Q-functions are learned and one is used to determine the maximizing action, while the other evaluates the Q-function for that action. The Double DQN algorithm extended this to neural networks [7]. However, Zhang *et al.* [24] observed that the double estimator sometimes underestimates the Q-value and propose to use a weighted average of the single and the double estimator as target. This work is similar to



a) b)

Fig. 4. **(a)** Mean (thick line) and standard deviation (shaded area) over 10 trials of the number of dropped targets per network $d = d_{max} - \beta$ in ACC over time for different environments with a uniform filter of size 15. **(b)** The normalized absolute error of the value estimate aggregated over the 6 environments. Shown are the mean with stratified bootstrapped confidence intervals computed from the results of 5 trials per environment. We used a uniform filter of size 401 for readability.

ours in the regard that depending on the parameter over- or underestimation could be corrected. A major difference to our algorithm is that the weighting parameter is computed from the maximum and minimum of the estimated Q-value and does not use unbiased rollouts. Similarly, the weighted estimator [25], [26] estimates the maximum over actions in the TD target as the sum of values weighted by their probability of being the maximum. In continuous action spaces this can be done through Gaussian process regression [26] and for discrete actions via dropout variational inference [25]. Different to ACC the weighting is computed from the same off-policy data used to compute the single quantities while ACC adjusts the weighting parameter β in a separate process using the latest on-policy rollouts. Lv *et al.* [27] use a similar weighting but suggest a stochastic selection of either the single or double estimator. The probability of choosing one or the other follows a predefined schedule. Other approaches compute the weighted average of the minimum and maximum over different Q-value estimates [21], [18]. However, the weighting parameter is a fixed hyperparameter. The TD3 algorithm [2] uses the minimum over two Q-value estimates as TD target. Maxmin Q-learning is another approach for discrete action spaces using an ensemble of Q-functions. For the TD target, first the minimum of over all Q-functions is computed followed by maximization with respect to the action [8]. Decreasing the ensemble size increases the estimated targets while increasing the size decreases the targets. Similarly to TQC this provides a way to control the bias in a more fine-grained way; the respective hyperparameter has to be set before the start of the training for each environment, however. Cetin *et al.* [28] propose to learn a pessimistic penalty to overcome the overestimation bias.

What sets ACC apart from the previously mentioned works is that unbiased on-policy rollouts are used to adjust a term that controls the bias correction instead of using some predefined heuristic.

B. Combining On- and Off-Policy Learning

There are many approaches that combine on- and off-policy learning by combining policy gradients with off-policy samples [29], [30], [31]. In [32] an actor-critic is used where the critic is updated off-policy and the actor is updated with a mixture of policy gradient and Q-gradient. This differs from our work in that we are interested only in better critic estimates through the information of on-policy samples. To learn better value estimates by combining on- and off-policy data prior works proposed the use of some form of importance sampling [33], [34]. In [35] the TD target is computed by mixing Monte Carlo samples with the bootstrap estimator. These methods provide a tradeoff between variance and bias. They differ from our work in using the actual returns directly in the TD targets while we incorporate the returns indirectly via another parameter. Bhatt *et al.* [36] propose the use of a mixture of on- and off-policy transitions to generate a feature normalization that can be used in off-policy TD learning. Applied to TD3, learning becomes more stable eliminating the need to use a delayed target network.

C. Hyperparameter Tuning for Reinforcement Learning

Most algorithms that tune hyperparameters of RL algorithms use many different instances of the environment to find a good setting [37], [38], [39]. There is, however, also work that adjusts a hyperparameter online during training [40]. In this work the meta-gradient (i.e., the gradient of the update rule) is used to adjust the discount factor and the length of bootstrapping intervals. However, it would not be straightforward to apply this method to control the bias of the value estimate. Their method also differs from ours in that they do not use a combination of on- and off-policy data.

VI. Conclusion

We present Adaptively Calibrated Critics (ACC), a general off-policy algorithm that learns a Q-value function with bias calibrated TD targets. The bias correction in the targets is determined via a parameter that is adjusted by comparing the current value estimates with the most recently observed on-policy returns. Our method incorporates information from the unbiased sample returns into the TD targets while keeping the high variance of the samples out. We apply ACC to TQC, a recent off-policy continuous control algorithm that allows fine-grained control of the TD target scale through a hyperparameter tuned per environment. With ACC, this parameter can automatically be adjusted during training, obviating the need for extensive tuning. The strong experimental results suggest that our method provides an efficient and general way to control the bias occurring in TD learning.

Interesting directions for future research are to evaluate the effectiveness of ACC applied to algorithms that work with discrete action spaces and when learning on a real robot where tuning of hyperparameters is very costly.

References

[1] A. Kuznetsov, P. Shvechikov, A. Grishin, and D. Vetrov, "Controlling overestimation bias with truncated mixture of continuous distributional quantile critics," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5556–5566.

[2] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*, 2018, pp. 1582–1591.

[3] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 1861–1870.

[4] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[6] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning," in *Proceedings of the 1993 Connectionist Models Summer School*, 1993, pp. 255–263.

[7] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[8] Q. Lan, Y. Pan, A. Fyshe, and M. White, "Maxmin q-learning: Controlling the estimation bias of q-learning," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=Bkg0u3Etwr

[9] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[10] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on Robot Learning*. PMLR, 2020, pp. 1094–1100.

[11] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," *CoRR*, vol. abs/1812.05905, 2018. [Online]. Available: http://arxiv.org/abs/1812.05905

[12] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *International Conference on Machine Learning*, 2017, pp. 449–458.

[13] W. Dabney, M. Rowland, M. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[14] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[16] I. Durugkar and P. Stone, "Td learning with constrained gradients," in *Proceedings of the Deep Reinforcement Learning Symposium, NIPS 2017*, Long Beach, CA, USA, December 2017. [Online]. Available: http://www.cs.utexas.edu/users/ai-lab?NIPS17-ishand

[17] H. V. Hasselt, "Double q-learning," in *Advances in Neural Information Processing Systems*, 2010, pp. 2613–2621.

[18] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, "Stabilizing off-policy q-learning via bootstrapping error reduction," in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 11 784–11 794.

[19] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, 2018.

[20] R. Agarwal, D. Schuurmans, and M. Norouzi, "An optimistic perspective on offline reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 104–114.

[21] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2052–2062.

[22] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control." in *IROS*. IEEE, 2012, pp. 5026–5033.

[23] R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare, "Deep reinforcement learning at the edge of the statistical precipice," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[24] Z. Zhang, Z. Pan, and M. J. Kochenderfer, "Weighted double q-learning," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. IJCAI'17. AAAI Press, 2017, pp. 3455–3461. [Online]. Available: http://dl.acm.org/citation.cfm?id=3172077.3172372

[25] A. Cini, C. D'Eramo, J. Peters, and C. Alippi, "Deep reinforcement learning with weighted q-learning," *arXiv preprint arXiv:2003.09280*, 2020.

[26] C. D'Eramo, A. Nuara, M. Pirotta, and M. Restelli, "Estimating the maximum expected value in continuous reinforcement learning problems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[27] P. Lv, X. Wang, Y. Cheng, and Z. Duan, "Stochastic double deep q-network," *IEEE Access*, vol. 7, pp. 79 446–79 454, 2019.

[28] E. Cetin and O. Celiktutan, "Learning pessimism for robust and efficient off-policy reinforcement learning," *arXiv preprint arXiv:2110.03375*, 2021.

[29] T. Degris, M. White, and R. Sutton, "Off-policy actor-critic," in *International Conference on Machine Learning*, 2012.

[30] T. Jie and P. Abbeel, "On a connection between importance sampling and the likelihood ratio policy gradient," in *Advances in Neural Information Processing Systems*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds., vol. 23, 2010, pp. 1000–1008.

[31] B. O'Donoghue, R. Munos, K. Kavukcuoglu, and V. Mnih, "Combining policy gradient and q-learning," in *ICLR*, 2016.

[32] S. S. Gu, T. Lillicrap, R. E. Turner, Z. Ghahramani, B. Schölkopf, and S. Levine, "Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 3846–3855.

[33] A. R. Mahmood, H. P. van Hasselt, and R. S. Sutton, "Weighted importance sampling for off-policy learning with linear function approximation," in *Advances in Neural Information Processing Systems*, vol. 27, 2014, pp. 3014–3022.

[34] D. Precup, "Eligibility traces for off-policy policy evaluation," *Computer Science Department Faculty Publication Series*, p. 80, 2000.

[35] M. Hausknecht and P. Stone, "On-policy vs. off-policy updates for deep reinforcement learning," in *Deep Reinforcement Learning: Frontiers and Challenges, IJCAI 2016 Workshop*, 2016.

[36] A. Bhatt, M. Argus, A. Amiranashvili, and T. Brox, "Crossnorm: Normalization for off-policy td reinforcement learning," *arXiv preprint arXiv:1902.05605*, 2019.

[37] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end with autorl," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2007–2014, 2019.

[38] S. Falkner, A. Klein, and F. Hutter, "BOHB: Robust and efficient hyperparameter optimization at scale," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 1437–1446.

[39] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, *et al.*, "Population based training of neural networks," *arXiv preprint arXiv:1711.09846*, 2017.

[40] Z. Xu, H. P. van Hasselt, and D. Silver, "Meta-gradient reinforcement learning," in *NeurIPS*, 2018.

H

Dynamic Update-to-Data Ratio: Minimizing World Model Overfitting

Nicolai Dorka
University of Freiburg
dorka@cs.uni-freiburg.de

Tim Welschehold
University of Freiburg

Wolfram Burgard
Technical University of Nuremberg

Abstract

Early stopping based on the validation set performance is a popular approach to find the right balance between under- and overfitting in the context of supervised learning. However, in reinforcement learning, even for supervised sub-problems such as world model learning, early stopping is not applicable as the dataset is continually evolving. As a solution, we propose a new general method that dynamically adjusts the update to data (UTD) ratio during training based on under- and overfitting detection on a small subset of the continuously collected experience not used for training. We apply our method to DreamerV2, a state-of-the-art model-based reinforcement learning algorithm, and evaluate it on the DeepMind Control Suite and the Atari 100k benchmark. The results demonstrate that one can better balance under- and overestimation by adjusting the UTD ratio with our approach compared to the default setting in DreamerV2 and that it is competitive with an extensive hyperparameter search which is not feasible for many applications. Our method eliminates the need to set the UTD hyperparameter by hand and even leads to a higher robustness with regard to other learning-related hyperparameters further reducing the amount of necessary tuning.

1 Introduction

In model-based reinforcement learning (RL) the agent learns a predictive world model to derive the policy for the given task through interaction with its environment. Previous work has shown that model-based approaches can achieve equal or even better results than their model-free counterparts [7, 12, 20, 23]. An additional advantage of using a world model is, that once it has been learned for one task, it can directly or after some finetuning be used for different tasks in the same environment potentially making the training of multiple skills for the agent considerably cheaper. Learning a world model is in principle a supervised learning problem. However, in contrast to the standard supervised learning setting, in model-based RL the dataset is not fixed and given at the beginning of training but is gathered over time through the interaction with the environment which raises additional challenges.

A typical problem in supervised learning is overfitting on a limited amount of data. This is well studied and besides several kinds of regularizations a common solution is to use a validation set that is not used for training but for continual evaluation of the trained model during training. By considering the learning curve on the validation set it is easy to detect if the model is under- or overfitting the training data. For neural networks a typical behavior is that too few updates lead to underfitting while too many updates lead to overfitting. In this context, the validation loss is a great tool to balance those two and to achieve a small error on unseen data.

For learning a world model on a dynamic dataset there unfortunately is no established method to determine if the model is under- or overfitting the training data available at the given point in time. Additionally, in model-based RL a poorly fit model can have a dramatic effect onto the learning result

Decision Awareness in Reinforcement Learning Workshop at the 39th International Conference on Machine Learning (ICML), Baltimore, Maryland, USA, 2022. Copyright 2022 by the author(s)

as from it the agent derives the policy, which influences the future collected experience which again influences the learning of the world model. So far, in model-based RL this is commonly addressed with some form of regularization and by setting an update-to-data (UTD) ratio that specifies how many update steps the model does per newly collected experience, similar to selecting the total number of parameter updates in supervised learning. Analogously to supervised learning, a higher UTD ratio is more prone to overfit the data and a lower one to underfit it. State-of-the-art methods set the UTD ratio at the beginning of the training and do not base the selection on a dynamic performance metric. Unfortunately, tuning this parameter is very costly as the complete training process has to be traversed several times. Furthermore, a fixed UTD ratio is often sub-optimal because different values for this parameter might be preferable at different stages of the training process.

In this paper, we propose a general method – called **D**ynamic **U**pdate-**t**o-**D**ata ratio (**DUTD**) – that adjusts the UTD ratio during training. DUTD is inspired by using early stopping to balance under- and overfitting. It stores a small portion of the collected experience in a separate validation buffer not used for training but instead used to track the development of the world models accuracy in order to detect under- and overfitting. Based on this, we then dynamically adjust the UTD ratio.

We evaluate DUTD applied to DreamerV2 [12] on the DeepMind Control Suite and the Atari100k benchmark. The results show that DUTD improves the world model and as a result increases the overall performance relative to the default DreamerV2 configuration. Most importantly, DUTD makes searching for the best UTD rate obsolete and is competitive with the best value found through extensive hyperparameter tuning of DreamerV2. Further, our experiments show that with DUTD the world model becomes considerably more robust with respect to the choice of the learning rate.

In summary, this paper makes the following contributions: i) we introduce a method to detect under- and overfitting of the world model online by evaluating it on hold-out data; ii) We use this information to dynamically adjust the UTD ratio to optimize world model performance; iii) Our method makes tuning the UTD hyperparameter obsolete; iv) We exemplarily apply our method to a state-of-the-art model-based RL method and show that it leads to an improved overall performance and higher robustness. We will make the source code of our implementation publicly available upon publication.

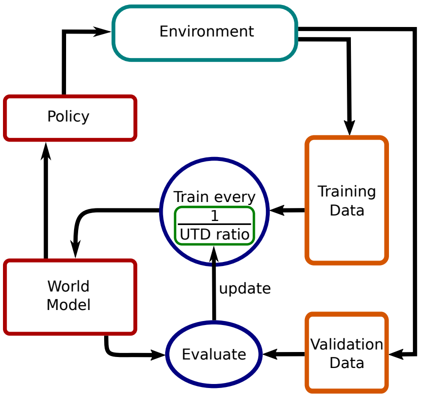


Figure 1: Overview of DUTD. A small subset of the experience collected from the environment is stored in a validation set not used for training. The world model is trained for one update after every $\frac{1}{\mathit{UTD\ ratio}}$ many environment steps. From time to time, *e.g.*, after an episode ended, the *UTD ratio* is adjusted depending on the detection of under- or overfitting of the world model on the validation data. The policy is obtained from the world model either by planning or learning and collects new data in the environment.

2 Related Work

In reinforcement learning there are two forms of generalization and overfitting. Inter-task overfitting describes overfitting to a specific environment such that performance on slightly different environments drops significantly. This appears in the context of sim-to-real, where the simulation is different from the target environment on which a well performing policy is desired, or when the environment changes slightly, for example, because of a different visual appearance [16, 18, 24, 30, 32]. In contrast, intra-task overfitting appears in the context of learning from limited data in a fixed environment when the model fits the data too perfectly and generalizes poorly to new data. We consider intra-task opposed to inter-task generalization.

In model-based reinforcement learning, there is also the problem of policy overfitting on an inaccurate dynamics model. As a result, the policy optimizes over the inaccuracies of the model and finds exploits that do not work on the actual environment. One approach is to use uncertainty estimates coming from an ensemble of dynamics models to be more conservative when the estimated uncertainty is high [7]. Another approach to prevent the policy from exploiting the model is to use different kinds of regularization on the plans the policy considers [3]. In contrast to these previous works, we directly tackle the source of the problem by learning a better dynamics model. Consequently, our method is orthogonal to and can easily be combined with the just mentioned line of work.

Directly targeting the overfitting of the dynamics model can be done through the usage of a Bayesian dynamics model and the uncertainties that come with such a model. Gaussian processes have been used successfully in this context [8] although it is difficult to scale this to high-dimensional problems. Another way to reduce overfitting of the dynamics model is to use techniques from supervised learning. This includes for example regularization of the weights, dropout [25], or data augmentation [14, 22]. All of these are also orthogonal to our method and can be combined with it to learn an even better dynamics model. Another popular approach is early stopping [2, 15, 26], where the training is stopped before the training loss converges. Our method can be regarded as the analogy of early stopping in a dynamic dataset scenario.

It is also possible to reduce the amount of model parameters for less overfitting but this comes with the risk of reduced performance if the right amount of training steps would have been performed. Our method overcomes this problem by automatically choosing the right amount of training steps for a given network.

Hyperparameter optimization for RL algorithms is also related to our work. For example, AlphaStar [23] has been improved by using Bayesian optimization [6]. Zhang et al. [31] demonstrated that model-based RL algorithms can be greatly improved through automatic hyperparameter optimization. A recent overview on automated RL is given by Parker-Holder et al. [17]. However, most of these approaches improve hyperparameters by training the RL agent on the environment in an inner loop while keeping the hyperparameters fixed during each run. Our work deviates from that by adapting a hyperparameter online during training of a single run. An approach that also falls into this category is from Schaul et al. [19], where behavior-related parameters such as stochasticity and optimism are dynamically adapted. Similarly, the algorithm Agent57 [4] adaptively chooses from a set of policies with different exploration strategies and achievs human level performance on all 57 Atari games [5]. Another approach adapts a hyperparameter that controls under- and overestimation of the value function online resulting in a model-free RL algorithm with strong performance on continuous control tasks [9].

In contrast to these approaches, we propose a method that directly learns a better world model by detecting under- and overfitting online on a validation set and dynamically adjusts the number of update steps accordingly. This renders the need to tune the UTD ratio hyperparameter unnecessary and further allows to automatically have it's value being adapted to the needs of the different training stages.

3 The DUTD Algorithm

In this section, we will first introduce the general setup, explain early stopping in the context of finding the right data fit and propose a new method that transfers this technique to the online learning setting. Lastly, we explain how the method can be applied to DreamerV2.

3.1 Model-Based Reinforcement Learning

We use the classical reinforcement learning framework [27] assuming a Markov decision process $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$. In this framework, the agent sequentially observes the current state $s_t \in \mathcal{S}$ in which it executes an action $a_t \in \mathcal{A}$, receives a scalar reward r_t according to the reward function $\mathcal{R}$, and transitions to a next state s_{t+1} generated by the unknown transition dynamics $\mathcal{P}$. The goal is to learn a policy that selects actions in each state such that the total expected return $\sum_{i=t}^{T} r_i$ is maximized.

Model-based reinforcement learning approaches learn a world model $\hat{\mathcal{P}}(s_{t+1} \mid s_t, a_t)$ – also called dynamics model – and a reward model $\hat{R}(r_t \mid s_t)$ that attempt to reflect their real but unknown counterparts. These models can then be used to learn a good policy by differentiating through the world models or by generating imaginary rollouts on which a reinforcement learning algorithm can be trained. Alternatively, the learned model can be used in a planning algorithm to select an action in the environment.

3.2 Under- and Overfitting

A well-known problem in supervised learning is that of overfitting, which typically corresponds to a low error on the training data and a high error on test data not seen during training. Usually, this happens if the model fits the training data too perfectly. In contrast to this, underfitting corresponds to the situation in which the model even poorly fits the training data and is characterized by both a high training and test error. To measure the performance of the model on unseen data, the available data is often split into a training and a validation set. Generally, only the training set is used to train the model while the validation set is used to evaluate its performance on new data.

For iterative training methods – like gradient descent based methods – overfitting is often detected by observing the learning curves for training and validation error against the number of training steps. A typical behavior is that in the beginning of the training both training and validation loss are decreasing. This is the region where the model is still underfitting. At some point, when the model starts overfitting the training data, only the training loss decreases further while the validation loss starts to increase. The aforementioned early stopping method balances under- and overfitting by stopping the training once the validation loss starts to increase.

While in supervised learning one can easily select a well fit model by using the validation loss, in reinforcement learning one cannot apply this technique as the dataset is not fixed but dynamic and is constantly growing or changing. Furthermore, the quality of the current policy influences the quality of the data collected in the future. Even though learning a world model is in principle a supervised task, this problem also occurs in the model-based RL framework.

3.3 Dynamic Update-to-Data Ratio

A typical hyperparameter in many RL algorithms is the update-to-data (UTD) ratio which specifies the number of update steps performed per environment step (i.e., per new data point). This ratio can in principle be used to balance under- and overfitting as one can control it in a way that not too few or too many updates steps are done on the currently available data. However, several problems arise while optimizing this parameter. First, it is very costly to tune this parameter as it requires to run the complete RL training several times making it infeasible for many potential applications. Second, the assumption that one fixed value is the optimal choice during the entire training duration does not necessarily hold. For example, if data from a newly explored region of the environment is added to the replay buffer it might be beneficial to increase the number of update steps.

To address these problems, we propose – DUTD – a new method that dynamically adjusts the UTD ratio during training. It is inspired by the early stopping criterion and targets at automatically balancing under- and overfitting online by adjusting the number of update steps. As part of the method, we store some of the experience in a separate validation buffer not used for training. Precisely, every d environment steps we collect s consecutive transitions from a few separate episodes dedicated to validation and every k environment steps the world model is evaluated on the validation buffer, where k should be much smaller than d. As the world model learning task is supervised this is easily done by recording the loss of the world model on the given validation sequences. The current validation loss is then compared to the validation loss of the previous evaluation. If the loss has decreased, we assume the model is still in the underfitting regime and increase the UTD rate by a specified amount. If the loss has increased, we assume the model to be in an overfitting regime and hence reduce the UTD rate. To allow for a finer resolution at the high-update side of the allowed interval we adjust the UTD rate in log-space, meaning it is increased or decreased by multiplying it with a value of c or $1/c$ respectively, where c is slightly larger than 1. The update formula at time step t then becomes

$$utd\_ratio_t = utd\_ratio_{t-k} \cdot b; \quad b = \begin{cases} c, & \text{if validation\_loss}_t < \text{validation\_loss}_{t-k}, \\ \frac{1}{c}, & \text{if validation\_loss}_t \geq \text{validation\_loss}_{t-k}. \end{cases} \tag{1}$$

DUTD is a general method that can be applied to any model-based RL algorithm that learns a world model in a supervised way. The implementation can be either in terms of the UTD ratio or the data-to-update ratio which is its inverse and which we call **IUTD** (i.e., the number of environment steps per update step). It is more convenient to use the UTD ratio if several updates are performed per environment step and the IUTD if an update step is only performed after some environment steps. Methodologically, the two settings are the same as the two ratios describe the same quantity and are just the inverse of each other.

A high-level overview of DUTD is shown in Figure 1 and the pseudocode is described in Algorithm 1, both explained in terms of the IUTD ratio as we will apply DUTD to an algorithm for which several update steps per environment steps becomes computationally very costly. However, in both framework both scenarios can be addressed by letting the ratio be a fractional. In the next section we will explain how DUTD can be applied specifically to the DreamerV2 algorithm [12].

3.4 Applying DUTD to DreamerV2

We apply DUTD to DreamerV2 [12], which is a model-based RL algorithm that builds on Dreamer [11] which again builds on PlaNet [10]. DreamerV2 learns a world model through latent imagination. The policy is learned purely in the latent space of this world model through an actor-critic framework. It is trained on imaginary roll-outs generated by the world model. The critic is regressed onto λ-targets [21, 27] and the actor is trained by a combination of Reinforce [29] and a dynamics backpropagation loss. The world model learns an image encoder that maps the input to a categorical latent state on which a Recurrent State-Space Model [10] learns the dynamics. Three predictors for image, reward, and discount factor are learned on the latent state. The total loss for the world model is a combination of losses for all three predictors and a Kullback–Leibler loss between the latents predicted by the dynamics and the latents from the encoder.

To apply DUTD we evaluate the image reconstruction loss on the validation set. Other choices are also possible but we speculate that the image prediction is the most difficult and important part of the world model. One could also use a combination of different losses but then one would potentially need a scaling factor for the different losses. As we want to keep our method simple and prevent the need of hyperparameter tuning for our method, we employ the single image loss.

Algorithm 1 DUTD (in terms of inverted UTD ratio)

```
Input: Initial inverted UTD ratio iutd_ratio; number of steps after which additional validation data is collected $d$, number of validation transitions collected $s$, steps after which the iutd_ratio is updated $k$, iutd update increment $c$
for $t = 1$ to total_num_of_env_steps do
    Act according to policy $\pi(a \mid s)$ and observe next state
    if $t$ mod $d == 0$ then
        Collect $s$ transitions and store experience in a separate validation buffer; increment $t = t + s$
    end if
    if $t$ mod $iutd\_ratio == 0$ then
        Perform one training step of the transition model
    end if
    if $t$ mod $k == 0$ then
        Compute model loss $L$ on validation dataset
        if $L \geq L_{previous}$ then                # Overfitting
            $iutd\_ratio = iutd\_ratio \cdot c$
        else                                         # Underfitting
            $iutd\_ratio = iutd\_ratio / c$
        end if
        $L_{previous} \ = \ L$
    end if
end for
```

4 Experiments

We evaluate DUTD applied to DreamerV2 on the Atari 100k benchmark [13] and the DeepMind Control Suite [28]. For each of the two benchmarks we use the respective hyperparameters provided by the authors in their original code base. Accordingly, the baseline IUTD ratio is set to a value of 5 for the control suite and 16 for Atari which we also use as initial value for our method. This means an update step is performed every 5 and 16 environment steps respectively. For both benchmarks we set the increment value of DUTD to $c = 1.3$ and the IUTD ratio is updated every 500 steps which corresponds to the length of one episode in the control suite (with a frame-skip of 2). Every $100,000$ steps DUTD collects $3,000$ transitions of additional validation data. We cap the IUTD ratio in the interval $[1, 15]$ for the control suite and in $[1, 32]$ for Atari. This is in principle not necessary and we find that most of the time the boundaries, especially the upper one, is not reached. A boundary below 1 would be possible by using fractions and doing several updates per environment step, but this would be computationally very expensive for DreamerV2. All other hyperparameters are reported in the Appendix. They were not extensively tuned and we observed that the performance of our method is robust with respect to the specific choices. The environment steps in all reported plots also include the data collected for the validation set.

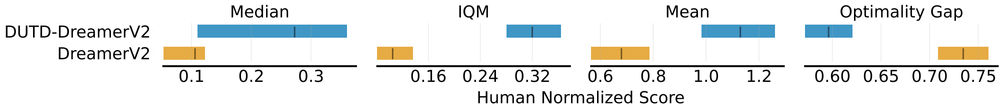


Figure 2: Aggregated metrics over 5 random seeds on the 26 games of Atari 100k with 95% confidence intervals according to the method presented in Agarwal et al. [1]. The intervals are estimated by the percentile bootstrap with statified sampling. Higher mean, median, interquantile mean (IQM) and lower optimality gap are better.

The Atari 100k benchmark [13] includes 26 games from the Arcade Learning Environment [5] and the agent is only allowed $100,000$ steps of environment interaction per game, which are $400,000$ frames with a frame-skip of 4 and corresponds to roughly two hours of real-time gameplay. The final performance per run is obtained by averaging the scores of 100 rollouts with the final policy after training has ended. We compute the human normalized score of each run as $\frac{\text{agent score}-\text{random score}}{\text{human score}-\text{random score}}$. The DeepMind Control Suite provides several environments for continuous control. Agents receive pixel inputs and operate with a frame-skip of 2 as in the original DreamerV2. We trained for 2 million frames on most environments and to save computation cost for 1 million frames if standard DreamerV2 already achieves its asymptotic performance well before that mark. The policy is evaluated every $10,000$ frames for 10 episodes. For both benchmarks, each algorithm is trained with 5 different seeds on every environment.

Our experiments are designed to demonstrate the following:

- The UTD ratio can be automatically adjusted using our DUTD approach
- DUTD generally increases performance (up to 300% on Atari100k) by learning an improved world model compared to the default version of DreamerV2
- DUTD increases the robustness of the RL agent with regard to learning-related hyperparameters
- DUTD is competitive with the best UTD hyperparameter found by an extensive grid search

4.1 Performance of DUTD compared to Standard DreamerV2

For Atari100k, Figure 2 shows results aggregated over the 26 games with the method of Agarwal et al. [1], where the interquantile mean (IQM) ignores the bottom and top 25% of the runs across all games and computes the mean over the remaining. The optimality gap describes the amount by which a minimal value of human level performance has not been achieved. In Figure 10 of the Appendix we present the learning curves for each environment. The results show that DUTD increases the performance of DreamerV2 drastically on all considered metrics. It increases the interquantile mean (IQM) score by roughly 300% and outperforms the human baseline in terms of mean score without any data augmentation.

Figure 3 shows the aggregated results for two million frames over ten environments of the Control Suite, which we list in the Appendix. The curves per environment are presented in Figure 11 of the Appendix further including results for ten more environments on which the algorithms run until one million frames. Compared to the manually set default UTD ratio, DUTD matches or improves the performance on every environment. Overall, DUTD improves the performance significantly although its average IUTD rate over all games and checkpoints is 5.84 similar to the default rate of 5 showing that DUTD better exploits the performed updates.

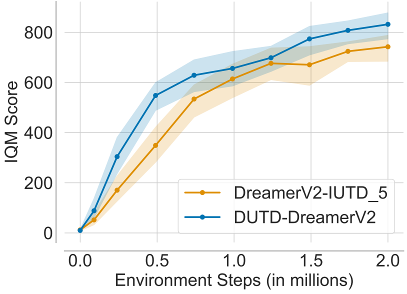


Figure 3: Sample efficiency curves aggregated from the results for ten environments of the DeepMind Control Suite for DreamerV2 with the default UTD ratio and when it is adjusted with DUTD. The IQM score at different training steps is plotted against the number of environment steps. Shaded regions denote pointwise 95% stratified bootstrap confidence intervals according to the method by Agarwal et al. [1].

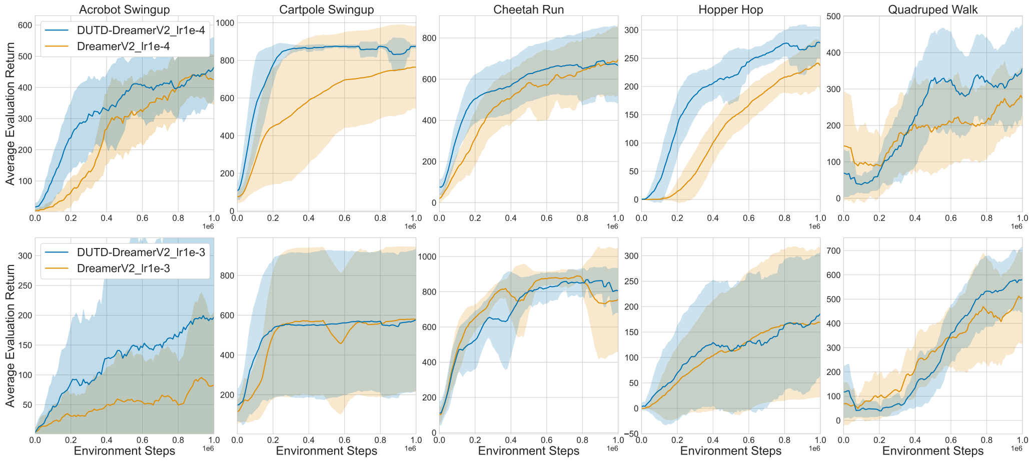


Figure 4: Learning curves for five environments of the Control Suite for DUTD-DreamerV2 and standard DreamerV2 when non-default learning rates are used. The first row shows the results for a lower than default learning rate of 0.0001 and the second row for a higher one of 0.001. The default learning rate is 0.003 and its results are shown in Figure 11. The solid line represents the mean and the shaded region a pointwise standard deviation in each direction computed over 5 runs.

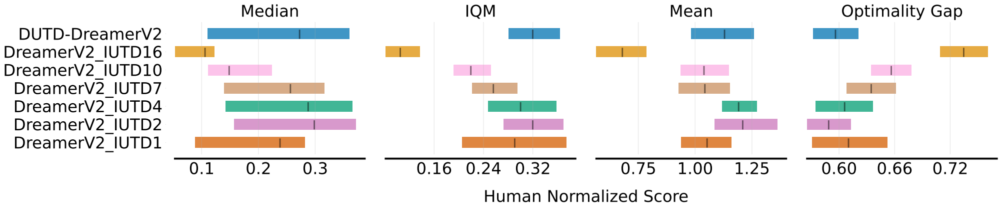


Figure 5: Aggregated metrics over 5 random seeds on the 26 games of Atari 100k, cf. Figure 2 for the methodology. DUTD is compared to Dreamer with different choices for the IUTD rate.

4.2 Increased Robustness with DUTD

As DUTD dynamically adjusts the UTD ratio which allows to modify the training process online, we formed the hypothesis that with DUTD the underlying RL algorithm is more robust to suboptimal learning hyperparameters. Similar to supervised learning on a fixed dataset the optimal number of updates to tradeoff between under- and overfitting will be highly dependent on hyperparameters like the learning rate. To investigate this, we evaluated DreamerV2 with and without our method for different learning rates of the dynamics model. The standard learning rate on the control suite is 0.0003. Hence, we trained with both a higher learning rate of 0.001 and a lower one of 0.0001 on a subset of the environments. The resulting learning curves are displayed in Figure 4. While the performance for a learning rate of 0.001 is overall rather similar, for a learning rate of 0.0001 the performance decreases substantially with the standard fixed IUTD ratio of 5. However, using DUTD the algorithm achieves considerably stronger results. This shows that using DUTD the algorithm is more robust to the learning rate, which is an important property when the algorithm is applied in real world settings such as robotic manipulation tasks, since multiple hyperparameter sweeps are often infeasible in such scenarios. The need for more robustness as offered by DUTD is demonstrated by the performance drop of DreamerV2 with a learning rate differing by a factor of 3 and the fact that on Atari a different learning rate is used.

4.3 Comparing DUTD with Extensive Hyperparameter Tuning

In the previous sections, we showed that DUTD improves the performance of DreamerV2 with its default IUTD rate significantly. Now we want to investigate the question of how well DUTD compares

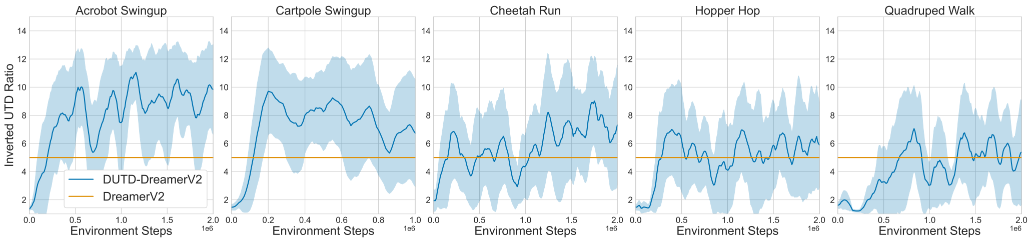


Figure 7: IUTD ratio against environment steps for DUTD and the standard DreamerV2 on five environments. For each environment the mean over 5 runs is plotted as the solid line and the shaded region represents one pointwise standard deviation in each direction.

to the best hyperparameter value for IUTD that can be found through an extensive grid search on each benchmark. While for many applications such a search is not feasible we are interested in what can be expected of DUTD relative to what can be regarded as the highest achievable performance.

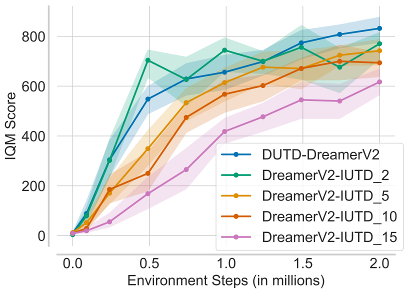


Figure 6: Sample efficiency curves showing the IQM score aggregated from the results for ten environments of the DeepMind Control Suite for DreamerV2 with different choices for the IUTD ratio. Shaded regions denote pointwise 95% stratified bootstrap confidence intervals.

On the Atari 100k benchmark we evaluate DreamerV2 with IUTD rates of $1, 2, 4, 7, 10$ and 16 (the default value) and denote the algorithms with DreamerV2-IUTD_1, DreamerV2-IUTD_2, etc. The aggregated results over all games and seeds in Figure 5 show an increase in performance when the number of updates increases up to an IUTD rate of 2. Increasing it further to 1 leads to declining results. Thus, there is a sweet spot and one can not simply set the IUTD rate very low and expect good results. Averaged over all runs and checkpoints the IUTD rate of DUTD is at 3.91 which is in the region of the best performing hyperparameters of 2 and 4. This is also reflected by the fact that DUTD achieves similar performance to these two optimal choices.

We further evaluate DreamerV2 with IUTD rates of $2, 5$ (the default one), 10, and 15 on ten environments of the control suite. An IUTD value below 2 is not possible as a single run would take roughly two weeks to run on our hardware. The aggregated sample efficiency curves in Figure 6 further support the hypothesis that DUTD is competitive with the results of an extensive grid search. Only an IUTD choice of 2 gives slightly better sample efficiency but reaches a lower final performance. To further investigate the behaviour of DUTD we report the adjusted **inverted** UTD ratio over time for five environments in Figure 7, and for all environments in Figure 12 in the Appendix. Interestingly, the behavior is similar for all the environments. At the start of the training, the ratio is very low and then it quickly oscillates around a value of roughly 5 for most environments and an even higher value for a few others. On cheetah_run and hopper_hop, the IUTD oscillates around the default value of 5 most of the time and still, DUTD reaches a higher performance than Dreamer as can be seen in the single environment plot in Figure 11 of the Appendix. This result supports the hypothesis that a static IUTD rate can be suboptimal for some environments and that DUTD successfully balances over- and underfitting during the training process.

4.4 Evaluating World Model Performance

Next we want to explore if the improved performance with DUTD actually originates from an improved world model that better generalizes to new data. To this end, we designed an experiment where the accuracy of the world model is constantly evaluated on a held out test set that is neither used for training nor validation. We evaluate each algorithm on 5 environments for 5 random seeds. For every combination of environment and seed we collect a test set with 20 episodes of random

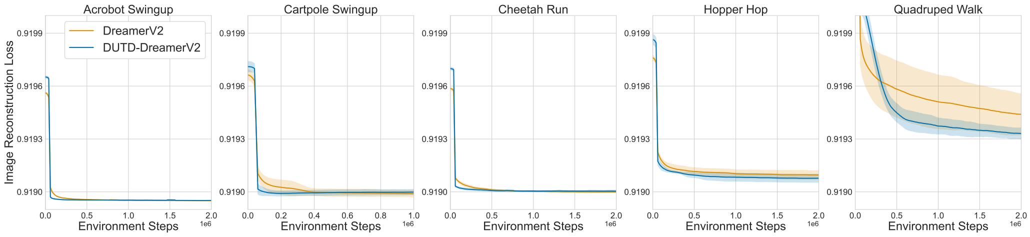


Figure 8: Image reconstruction loss of the world model on a held out test set for five environments of the DeepMind Control Suite. The solid line represents the mean and the shaded region a pointwise standard deviation in each direction computed over 5 runs.

interactions before the training starts. This test set is the same for our approach and the baseline to ensure the data selection procedure avoids bias in the test data induced through the respective world models and policies. In Figure 8 we plot the image prediction loss on the test set averaged over the 5 seeds for DreamerV2 with the default UTD rate and when using DUTD. The curves validate our hypothesis that with DUTD the world model achieves on average a better accuracy and hence that the validation data is indeed useful for balancing under- and overfitting. For the acrobot, cartpole, and cheetah environment, the accuracy with DUTD is better in the beginning and then becomes similar which, according to our opinion, is due to the fact that the asymptotic performance of the corresponding policy is also reached quickly on those environments. On hopper, the accuracy is better during the whole learning process which is also reflected in a better performing policy (cf. Figure 11). These results indicate that the performance increase originates indeed from an improved world model.

5 Discussion

We presented a novel and general method denoted as DUTD that is designed to detect under- and overfitting on evolving datasets and is able to dynamically adjust the typically hand-set UTD ratio in an automated fashion. As in early stopping, the underlying rationale is that too many updates can lead to overfitting while too few updates can lead to underfitting. DUTD quickly identifies such trends by tracking the development of the world model performance on a validation set. It then accordingly increases or decreases the UTD ratio in the case of underfitting or overfitting.

In our experiments, we demonstrated how to successfully apply DUTD to a model-based RL algorithm like DreamerV2. The experiments show that DUTD can automatically balance between the under- and overfitting of the world model by adjusting the UTD ratio. As a result, DUTD removes the burden of manually setting the UTD ratio, which otherwise needs to be tuned for new environments making it prohibitively expensive to apply such algorithms in many domains. At the same time, DUTD increases the performance of DreamerV2 significantly compared to its default UTD rate and is competitive with the best hyperparameter found for each domain through an extensive hyperparameter search. Moreover, a notable property of DUTD-DreamerV2 is its robustness to changes in the learning rate. This is important, as the learning rate often has to be tuned for new environments. For example, in DreamerV2 the default learning rate differs between Atari and the DeepMind Control Suite. In the context of real world problems such tuning is undesirable and often too costly. At the same time, the hyperparameters of DUTD can easily be set and do not have a big influence on the final performance. We recommend updating the UTD rate after a fixed time interval that is similar to the average episode length. The data used for validation should not exceed 10% of all data.

An interesting avenue for future work would be to explore non-supervised objectives for model-free RL algorithms that can be used for evaluation on the validation set. This would allow the usage of DUTD to adjust the UTD ratio of such algorithms.

We are convinced that DUTD is a further step in the direction of autonomy and the easy applicability of RL algorithms to new real world problems without the need to tune any hyperparameters in an inner loop.

References

[1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 34, 2021.

[2] RS Anderssen and PM Prenter. A formal comparison of methods proposed for the numerical solution of first kind integral equations. *The ANZIAM Journal*, 22(4):488–500, 1981.

[3] Dilip Arumugam, David Abel, Kavosh Asadi, Nakul Gopalan, Christopher Grimm, Jun Ki Lee, Lucas Lehnert, and Michael L Littman. Mitigating planner overfitting in model-based reinforcement learning. *arXiv preprint arXiv:1812.01129*, 2018.

[4] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning*, pages 507–517. PMLR, 2020.

[5] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[6] Yutian Chen, Aja Huang, Ziyu Wang, Ioannis Antonoglou, Julian Schrittwieser, David Silver, and Nando de Freitas. Bayesian optimization in alphago. *arXiv preprint arXiv:1812.06855*, 2018.

[7] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in Neural Information Processing Systems*, 31, 2018.

[8] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472. Citeseer, 2011.

[9] Nicolai Dorka, Joschka Boedecker, and Wolfram Burgard. Adaptively calibrated critic estimates for deep reinforcement learning. In *Deep RL Workshop NeurIPS 2021*, 2021.

[10] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

[11] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=S1lOTC4tDS`.

[12] Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=0oabwyZbOu`.

[13] Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model based reinforcement learning for atari. In *International Conference on Learning Representations*, 2019.

[14] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in Neural Information Processing Systems*, 33, 2020.

[15] Nelson Morgan and Hervé Bourlard. Generalization and parameter estimation in feedforward nets: Some experiments. *Advances in neural information processing systems*, 2:630–637, 1989.

[16] Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.

[17] Jack Parker-Holder, Raghu Rajan, Xingyou Song, André Biedenkapp, Yingjie Miao, Theresa Eimer, Baohe Zhang, Vu Nguyen, Roberto Calandra, Aleksandra Faust, et al. Automated reinforcement learning (autorl): A survey and open problems. *arXiv preprint arXiv:2201.03916*, 2022.

[18] Roberta Raileanu, Maxwell Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in reinforcement learning. 2020.

[19] Tom Schaul, Diana Borsa, David Ding, David Szepesvari, Georg Ostrovski, Will Dabney, and Simon Osindero. Adapting behaviour for learning progress. *arXiv preprint arXiv:1912.06910*, 2019.

[20] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[21] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[22] Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=uCQfPZwRaUu`.

[23] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[24] Xingyou Song, Yiding Jiang, Stephen Tu, Yilun Du, and Behnam Neyshabur. Observational overfitting in reinforcement learning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HJli2hNKDH`.

[25] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

[26] Otto Neall Strand. Theory and methods related to the singular-function expansion and landweber's iteration for integral equations of the first kind. *SIAM Journal on Numerical Analysis*, 11 (4):798–825, 1974.

[27] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction.* MIT Press, 2018. ISBN 78-0262039246.

[28] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[29] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[30] Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, 2018.

[31] Baohe Zhang, Raghu Rajan, Luis Pineda, Nathan Lambert, André Biedenkapp, Kurtland Chua, Frank Hutter, and Roberto Calandra. On the importance of hyperparameter optimization for model-based reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 4015–4023. PMLR, 2021.

[32] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.

I

Sensor-based Human-Robot Collaboration for Industrial Tasks

Alexandre Angleraud[a], Akif Ekrekli[a], Kulunu Samarawickrama[a], Gaurang Sharma[a] and Roel Pieters[a]

[a] *Unit of Automation Technology and Mechanical Engineering, Tampere University, Korkeakoulunkatu 6, Tampere, Finland*

ARTICLE INFO

Keywords:
Human-robot collaboration
Visual perception
Deep learning

ABSTRACT

Collaboration between human and robot requires interaction modalities that suit the context of the shared tasks and the environment in which it takes place. While an industrial environment can be tailored to favor certain conditions (e.g., lighting), some limitations cannot so easily be addressed (e.g., noise, dirt). In addition, operators are typically continuously active and cannot spare long time instances away from their tasks engaging with physical user interfaces. Sensor-based approaches that recognize humans and their actions to interact with a robot have therefor great potential. This work demonstrates how human-robot collaboration can be supported by visual perception models, for the detection of objects, targets, humans and their actions. For each model we present details with respect to the required data, the training of a model and its inference on real images. Moreover, we provide all developments for the integration of the models to an industrially relevant use case, in terms of software for training data generation and human-robot collaboration experiments. These are available open-source in the OpenDR toolkit at https://github.com/opendr-eu/opendr. Results are discussed in terms of performance and robustness of the models, and their limitations. Although the results are promising, learning-based models are not trivial to apply to new situations or tasks. Therefore, we discuss the challenges identified, when integrating them into an industrially relevant environment.

1. Introduction

Collaborative robots (co-bots) can improve the safety, work efficiency and productivity of industrial processes by acting as flexible and reconfigurable tool to human operators. Within Industry 4.0, co-bots have a core role to contribute to the transition from traditional manufacturing to digital manufacturing [3, 13]. Co-bots can be easily programmed and reconfigured, and are safe for interaction, due to their small form-factor and incorporated sensor systems that can detect collisions [55]. Co-bots are also to be found in high-payload form, where protective covering can be complemented by sensor-based safety features. Human-robot collaboration (HRC) is typically possible in two ways [60]: 1. Off-line programming of robot tasks by demonstration (also known as hand-guiding or kinesthetic teaching), and 2. On-line interaction between human and robot, enabled by external sensor systems. While off-line programming is an established method of collaboration, on-line interaction still typically requires great efforts in development and its success depends highly on the sensor system. That is, if the external sensor system is not robust or has high latency, this reflects negatively on the performance of the collaboration.

Nevertheless, the role of humans and industrial robots in smart factories is often emphasized [13] and future roadmaps state clear benefits on utilizing collaboration between humans and robots [55]. The practical requirements and tools needed, however, are often underestimated or given little attention, resulting in great interest from industry and SMEs, but not many practical implementations [60]. To be realistic, successful integration of perception tools in human-robot collaboration requires considerable effort towards the selection of suitable detection tools, the preparation of suitable data for training and the actual training of a detection model, followed by its implementation in the robotic system. In this work we address these issues, and present the following contributions:

1. Identification of challenges for deep learning-based visual perception in HRC
2. Practical integration details for three deep learning-based visual perception tools in HRC
3. Open-source software templates for sensor-based HRC
4. Validation of the sensor-based HRC framework with an industrial use case

The problems we aim to address in this work are the current limitations in perception models and situational awareness for industrial human-robot collaboration. Perception and situational awareness of robot systems can be enhanced, such that fluent and responsive collaboration between human and robot is possible. We believe that perception models, based on deep learning, are ideal for this, as they can be accurate, reliable and fast to execute. These can then provide the required sensory input for interaction, such as the human body and its pose, human actions or gestures, and the pose of objects and targets in the scene. Developing and integrating such models for robotics in industry are hard tasks, often requiring expertise from many different areas [47]. Therefore, we additionally provide a general HRC software framework, based on ROS [39], which can be utilized to replicate our developments. The framework is build around OpenDR [37], a deep learning toolkit for robotics, and has the perception tools integrated for a practical and industrially relevant use case in agile production. The visual perception tools are human skeleton detection, human action recognition and the detection and pose estimation of objects and targets in the scene.

In the following section, the current challenges of perception for HRC are identified, when considering deployment in industrial environments.

1.1. Challenges for sensor-based HRC

The first two identified challenges relate to typical and well-known issues of learning-based perception [28], i.e., perception model selection and training data collection. The last two identified challenges relate to the applicaton and integration of such models to an industrial environment.

1. **Model selection and training -** The choice of perception model depends mostly on what needs to be detected. Many well-performing models exist, e.g. for common objects households objects [25] or humans [33, 54]. However, simply selecting the model with the highest accuracy is usually not the best approach. For example, a model that detects humans in an automotive scenario would not perform well in industrial scenario. All relevant context and properties of the model needs to be considered, as it will affect the performance with respect to the intended use case. Moreover, properties such as model size and inference time are of practical importance for human-robot collaboration where delay and responsiveness of the interaction matter greatly.

2. **Data collection -** The performance of a detection model is directly influenced by the quality and quantity of the data used for training. Data and its annotation need to include enough variability that could occur in the real use case, without enlarging the dataset unnecessarily. While in certain areas large datasets exist (e.g., household objects [22]), in other cases the dataset needs to be collected or generated from scratch. Collecting real data is usually preferred, as it captures the realistic content of the target object as well as the sensor, however, synthetic data has also shown suitable performance in many cases [35]. One additional problem for data collection is the annotation of the data with the ground truth, for example, object classes or 6D object poses. For real data, annotation is difficult and time-consuming, and in some cases near impossible (e.g., object poses). In this case simulation and the generation of synthetic data has the benefit of knowing exactly where an object is rendered in the virtual world [48].

3. **Reliability and safety -** Deep neural networks (DNN) are known as black-box models, implying that their inner workings cannot (easily) be understood [5]. Explainable AI aims to provide explanations to models, even though there is no general consensus of what is meant by explainable and/or interpretable [20]. In case of safety-critical applications (e.g. autonomous driving or human-robot collaboration), DNN cannot provide required reliability and safety levels [18]. Moreover, model performance, failure probability and their uncertainty are difficult to determine and can drift during long-term operation. While continual learning might prove useful in this regard, developments are still in early stages [30].

4. **Integration -** Deploying DNNs to a real environment requires integration efforts that depend on the model and its intended outcome. Clear differences can be identified between models that provide input for on-line decision making and models that provide diagnostics for off-line monitoring [52]. For example, in manufacturing environments, the detection of obstacles and humans needs to provide timely input to machinery for halting processes. As such, the operating equipment needs to be shut down and tested extensively to ensure reliable working of the developed tools [44]. Predictive maintenance, on the other hand, only provides recommendations and does not interfere with running processes. Data collection and installation of models can, therefore, often be done while machinery is in operation or without rigorous testing protocols [57]. One additional challenge is the availability of state-of-the-art DNN tools. While most developments are open-source available and can even be commercialized, there is no guarantee for code-quality and its maintenance [21]. Support for the software is typically not offered by the tool developers, and tools quickly become obsolete due to, for example, general software updates. As industrial systems are operational for extended time periods (years), investment in upgrading is not a regular occurrence.

These identified challenges are broad research topics, and cannot be tackled by individual research efforts, but require community effort to push boundaries forward. We therefore do not claim in this work that we provide a solution to these challenges but offer directions in the specific area of human-robot collaboration how the challenges can be taken into account. The remainder of this paper is organized as follows. In Section 2 we provide an overview of related work in human-robot collaboration and relevant perception tools. As a result of this overview, several perception tools are selected for implementation and explained in further detail in Section 3. Section 4 describes the industrial assembly use case, the software framework as well as integration details needed to replicate the research developments. The results of the perception modules and the human-robot collaboration experiments are presented and discussed in Section 5. Finally, Section 6 concludes the work.

2. Related work

2.1. Human-robot collaboration

Collaboration between human and robot has been an ongoing trend since the advent of smart manufacturing [13] and Industry 4.0 [55]. Formal definitions of collaboration, working zones and operating modes are common [51] and standards provide requirements and design guidelines to ensure safety for operators. [53] provides an overview of symbiotic human-robot collaborative assembly and highlights future research directions. Methods presented include voice processing, gesture recognition, haptic interaction, and even brainwave perception. In most cases deep learning is used for classification, recognition and context awareness identification. Computer vision-based approaches are the most popular, as presented in [14]. This reports a systematic review of computer vision-based holistic scene understanding in HRC scenarios, which mainly takes into account the cognition of object, human, and environment. Subsequently, visual reasoning can be used to gather and compile visual information into semantic knowledge for robot decision-making and proactive collaboration.

Particular examples of vision-based HRC are as follows. Visual recognition of repetitive assembly actions is presented in [8] and utilizes object detection with Yolo [40] and human pose estimation with OpenPose [6] as separate methods. Real-time human-robot interaction has been demonstrated in [32], with hand gesture detection utilized for robot programming. An overview of approaches is presented in [42, 44], where again vision-based methods are high-lighted as most popular, in combination with machine learning.

2.2. Human detection

The detection of humans, individual body parts and their actions based on visual information has been a long-standing problem in computer vision [33].

Human presence detection - Detecting the presence of a person in the robot work space has been an active area of research, mainly to ensure safety of the human [61]. Different visual modalities can be used to detect humans [23]. In [34], a depth sensor is utilized, producing data in the form of a point cloud. From this, a convex hull of the human point cloud is created and background removal detects any moving objects/subjects in the scene. Similar is the work in [15], where a depth map is utilized to detect a person's presence, but also to allow interaction with a projected graphical user interface. A dynamically updated workspace model is, therefore, required. Depth cameras are also used in [27] for the detection of a person in the work space and to compute their distance to the robot. In addition, laser scanners at leg-level are included to detect an operator's presence. It is noted that both sensing systems work in parallel and do not fuse information together, allowing a redundancy for safety. 3D LiDAR-based detection of humans is presented in [59], which utilizes a learning-based approach for human classification. The work, however, targets large indoor public spaces and a mobile service robot. In [23] a comparison is made between the performance of state-of-the-art person detectors for 2D range data, 3D LiDAR, and RGB-D data, as well as selected combinations thereof, in a challenging industrial use case. Multi-modal approaches have also gained interest [38], however, most works only consider larger environments for mobile robots (or cars) [19], making their suitability for small and dense industrial environments questionable. Human pose estimation goes beyond human detection by estimating 3D poses of humans and their individual skeleton joints. Well-known approaches are OpenPose [6] and VoxelPose [49], which can utilizes single as well as multiple cameras.

Gesture detection - Detection and recognition of human gestures has also been of interest to robotics. In [24], a comprehensive review is given of different gesture recognition approaches for human-robot collaboration. Besides visual perception, the review also includes non-image based approaches, such as wearables. Related to our work, is [32], which demonstrates a real-time human-robot interaction framework with robust background invariant hand gesture detection. The approach presents a method to collect a training dataset for static hand gestures, taken from letters and numbers from American sign language.

Human action recognition - As an extension to the detection of humans and their gestures, the methods of human action recognition consider the behavior of a person, i.e., their actions or motions, to be detected [54, 46]. This implies an image sequence to be used for recognition, as compared to single images in e.g., human detection. Recent progress has been achieved by deep learning approaches that take as input an image sequence in RGB-D format, extracts the 2D or 3D skeleton pose and performs action classification [58]. In relation to human-robot collaboration, research on action recognition has also focused on industrial activities [10] and pose forecasting [43], including actions such as picking, placing, assembling, polishing, etc.

2.3. Object detection and pose estimation

State of the art deep neural networks have shown impressive performance for generic object categories [25]. Real-time object detection is an active research problem to allow adoption to robotics applications, and many works can be found that ave utilized detectors for tasks such as robot grasping [11]. Popular approaches are for example, Faster R-CNN [41], Yolo [40] and SSD [26]. Pose estimation of objects considers to estimate the 6D pose of an object. Similar to object detection, different approaches exist, such as correspondence-based methods 3DMatch [62], template-based methods such as PoseCNN [7] and voting based methods such as DenseFusion [53]. For both object detection and pose estimation, datasets can be found, for example, Pascal VOC [12] and COCO [22] for 2D object detection, and, more recently, Objectron [1] and T-LESS [16] for 3D objects and 6D pose estimation. It is important to mention a crucial difference between these methods of object detection and pose estimation, as compared to human detection and pose estimation. In general, most human perception approaches are successful with a large variety in humans. That means existing dataset are sufficient to be used in new areas with new humans. In contrast, most object perception approaches do not scale well to novel objects and additional data should be generated to train a model and achieve successful detection. In this work, results were achieved in a similar manner.

2.4. Other interaction modalities

Speech - Utilizing speech as interaction modality has the benefit of not requiring physical actions for the human, allowing work-related tasks to be uninterrupted [31]. As a research field, the maturity has increased significantly recently, due to advancements of speech recognition technologies, with respect to recognition performance and robustness against noise [50]. However, despite the maturity in speech recognition performance, the connection of speech commands to robot actions and/or higher-level goals requires internal representations that need to be developed as well [29]. For tasks that are low in complexity (e.g., pick-and-place, hand-overs) such knowledge representation is manageable [4], but with increasing conversational capabilities in natural language perception, knowledge representation requires careful and extensive modelling.

Graphical user interface - The most common modality for programming industrial robots is a graphical user interface (GUI) [51]. Robot tasks and motions can be achieved by either robot hand-guiding and a teaching pendant, or by low-level programming with suitable programming language and software toolbox. In both cases a GUI is utilized to assist in the programming and/or teaching of robot tasks. GUIs are typically developed with ease-of-use in mind and, recently, user perceptions such as user experience, user effort and understanding are actively taken into account as well [9]. As a graphical tool, GUIs offer great capabilities, such as visualization and simulation, integrated as apart of the robot programming stage. Limitations, however, have been identified as well, such as a higher cognitive burden needed for end-users [2]. While GUIs are beneficial for the programming of robots, they are not well suited for interaction during task execution. Human-robot collaboration requires responsiveness of the robot to human cues, which is difficult to achieve with a GUI alone.

From this brief overview of related work, a few observations can be made. Most perception tools are developed and presented without robotics in mind, aiming for general target groups. This implies that specific characteristics relevant for human-robot collaboration in industrial environments are not included or tested, making their suitability for this questionable. For example, manufacturing environments can be dirty and noisy, and specific conditions, such as lighting, can be difficult to adjust, in contrast to laboratory and domestic environments. In addition, while the utilization (and replication) of perception tools is often possible by open-source software, details on integration are usually limited to just the tool itself and not to a common framework for robotics (e.g., ROS). Exceptions to this can be found, however, these are typically individual tools with specific robots [32]. Our work aims to fill this gap, by detailing three different visual perception tools with respect to human-robot collaboration. For all three, we provide detailed explanations on how to replicate our work, from dataset generation and training tools, to code examples (Python and ROS) and integrated experiments with a collaborative robot. All developments can found open-source in the OpenDR toolkit [37].

3. Visual recognition modules

All three integrated visual recognition modules utilize color images only for detection and recognition. Depth perception was intentionally excluded to aim that perception models run at high update rate, ideally in real-time (i.e., 20 FPS or higher). Especially for the detection of a person and their gestures this is needed to have a responsive system with low delay time.

3.1. Human skeleton detection

Method - Detection of a human in the scene is done with OpenPose [6], a real-time multi-person human pose detector. OpenPose is capable of detecting up to a total of 135 human body, foot, hand, and facial key points, from a single or multiple image/camera sources. To achieve detections in realtime, the lightweight version of OpenPose is used, as reported in [36]. For a successful detected human pose the method returns a list 18 2D image key points of the human skeleton with associated key point abbreviation.

Data generation and model training - The method in this work utilizes the pretrained MobileNet model as explained in [36], which was trained and evaluated with the COCO 2017 dataset [22] under default training parameters.

3.2. Human action recognition

Method - Recognition of human actions is done with ST-GCN [58], a real-time skeleton-based human action recognition framework. Depending on the dataset the method can detect a large number of different human actions, ranging from daily activities to complex actions with interactions. It utilizes the lightweight OpenPose model, to estimate the location of the human joints in every image, to generate a sequence of detected human skeleton graphs, connected both spatially and temporally.

Data generation and model training - The utilized training dataset is NTU-RGB+D [45], which contains 56,000 human action clips in 60 human action classes. For each image human skeleton joints are annotated in 3D, with respect to the camera coordinate system. The pretrained model from the original authors, with default training parameters, is used for inference.

3.3. Assembly object and target detection

Method - Detection of objects and targets in the scene utilizes the Mask R-CNN model from Detectron2 [56]. Mask R-CNN combines a Region Proposal Network (RPN) with the CNN model, to simultaneously predict object bounds and objectness scores at each position. RPN and Fast R-CNN are then merged into a single network. After objects and targets are detected, their orientation is estimated in each bounding box by the second order moment from a segmented object or target.

Data generation and model training - As the assembly objects and targets are novel with respect to existing datasets, a custom dataset needed to be generated. For this, 200 images of eight object and target classes were annotated with segmentation polygons, as depicted in Fig. 1. The object classes included rocker arms, bolts and pushrods, and the target classes included the Diesel engine, small and big pushrod holes, bolt holes and rocker arm locations. This data was augmented to include a broad variation in noise and lighting conditions, to form the custom dataset of <300,000 images. The methods for data generation and annotation are available in the OpenDR toolkit[1].

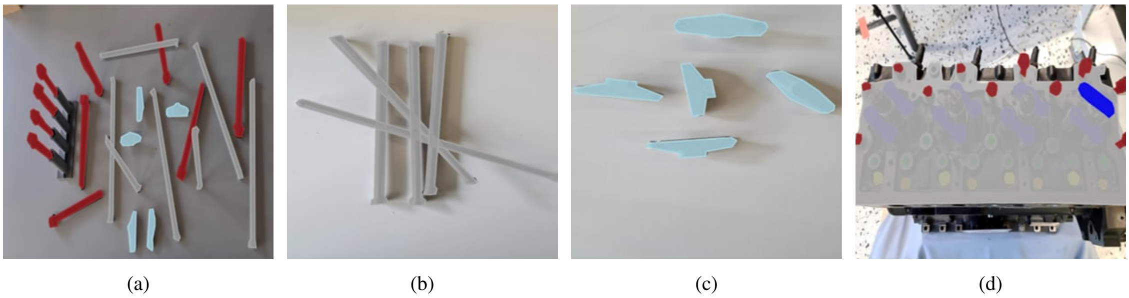


Figure 1: Image annotations for assembly objects, including bolts (red), pushrods (grey) and rocker arms (light blue); (a), (b) and (c), and targets objects, including Diesel engine (grey), small (yellow) and big (orange) pushrod holes, bolt holes (green) and rocker arm locations (dark blue); (d). Annotations are done with segmentation polygons in different colors, for different object classes. A total of 200 images with eight object and target classes were utilized for augmentation and dataset generation.

4. Industrial assembly use case

4.1. Diesel engine assembly

The manufacturing of Diesel engines involves assembly steps that are hard to automate, such as contact placement and manipulation of parts with various degrees of freedom. For example, rocker arm placement, push rod insertion and bolt fastening all have different constraints with respect to the final manipulation of the part to the engine. Rocker arms can be moved freely in 3D task space before placements, push rod insertion requires vertical motion into a pushrod hole and bolt fastening requires rotational motion and compliance orthogonal to vertical motion. In addition, parts to assemble are complex in shape, metallic and require lubricant for assembly and for operation. This means traditional robotic operations for picking and placing are not suitable for assembly and manual actions are the standard approach for manufacturing. A promising alternative, however, is to utilize the robot as assistant and assign tasks to it that support the assembly procedure and the ergonomy of the human operator. These are easy, but repetitive tasks, such as pick and placement, and actions for operator assistance such as hand-overs of parts and tools.

The scenario for human-robot collaboration is depicted in Fig. 2 and includes the Diesel engine, a table with parts and tools, the human operator and a collaborative robot. To demonstrate and validate our developments, we constructed a use case in which the robot picks and places parts from the table to the engine (push rods) and hands-over parts from the table to the operator (rocker arms and bolts). Visual perception is used as input to robot actions, i.e., to detect the position and orientation of objects and targets. At the same time, human skeleton detection and human action recognition is used as input to coordinate the human-robot collaborative tasks and to introduce safety for the human operator, i.e., only allowing robot actions to be executed when a person's hands are away from the engine.

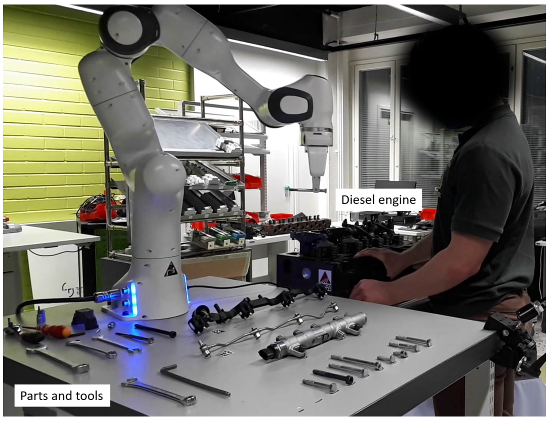


Figure 2: Experimental setup with a collaborative robot (Franka Emika), Diesel engine and parts for assembly tasks.

4.2. Integration

All developments are integrated in the OpenDR[1] toolkit [37] with ROS/ROS2[2] nodes of the perception tools and ROS moveit2[3] scripts for the human-robot collaboration scenarios, enabling to easily replicate (and extend) our work. For robot and perception hardware, we utilize the Franka Emika collaborative robot[4] and an Intel Realsense D435 RGB-D camera.

A Python script example of a visual recognition module is shown in Listing 1, demonstrating its usage. Here, a pretrained model for Detectron2 is loaded and the model inference is run on an input image. The prediction results of the model are drawn as boxes on the image as well. It should be noted that other tools of the OpenDR toolkit, i.e., human skeleton detection, human action recognition, as well as other perception tools, datasets and trained models, can be

[1] https://github.com/opendr-eu/opendr

[2] https://www.ros.org/

[3] https://moveit.picknik.ai/

[4] https://franka.de/

utilized in a similar manner [37]. For example, in the case of object and target detection, a custom dataset was generated, as explained briefly in Section 3.3. This included image annotation and augmentation, with the open-source tools Label Studio[5] and Albumentations[6], respectively. These are also integrated into the OpenDR perception tools, in form of Python scripts and Jupyter notebooks[7].

Listing 1: Object and target detections script in OpenDR[1]

```
from opendr.engine.data import Image
from opendr.perception.object_detection_2d import
    ↪ Detectron2Learner

# load model and run inference on image
detectron2 = Detectron2Learner(device="cpu")
detectron2.download(".", mode="pretrained")
detectron2.load("./detectron2_default")
img = Image.open("input_image.jpg")
predictions = detectron2.infer(img)

# draw bounding boxes of predictions on image
boxes = BoundingBoxList([box for kp,box in predictions])
draw_bounding_boxes(img.opencv(), boxes, class_names=
    ↪ detectron2.classes, show=True)
```

A python script example of robot actions is shown in Listing 2, demonstrating how to define a pick and place task with several concatenated actions. These low-level actions are based on Moveit2[3] and therefore robot-agnostic. In the example, motions are defined in task space as 2D planar motion parallel to the table (`2D_action`) and 1D motion vertical to the table (`1D_action`), to perform a grasping action. Additional actions include end-effector rotations (`rotate_EE`) and gripper actions (`move_gripper`). Actions can take input from a visual recognition module, as shown by the inclusion of `object` and `place`.

Listing 2: Robot actions script in OpenDR[1]

```
def Pick_and_Place(object,place):
    # Move and align robot above object
    2D_action(pose=[object.x, object.y], slow=False)
    rotate_EE(angle=object.angle)
    # Move robot down and grasp object
    1D_action(z_pose=0.35, slow=True)
    move_gripper(speed=20.0, width=0.02)
    # Move robot to place and release object
    1D_action(z_pose=0.2, slow=True)
    2D_action(pose=[place.x, place.y], slow=False)
    1D_action(z_pose=0.35, slow=True)
    move_gripper(speed=20.0, width=0.08)
```

A python script example for human-robot collaboration is shown in Listing 3, demonstrating how to combine the visual recognition modules and the robot actions. In the example, whenever a visual recognition module publishes a message, i.e., when a successful detection is made, a callback function is called with successive robot actions. This can therefore be used for human coordination of the assembly process, by triggering, halting and/or resuming robot actions.

Listing 3: Human-robot collaboration script in OpenDR[1]

```
def AR_callback(AR_data):
    if AR_data.id == 37 and AR_data.score > 0.80:
    # Stop robot motion when 'salute' is detected
        stopAction()
    elif AR_data.id == 39 and AR_data.score > 0.80:
    # Continue when 'cross hands in front' is detected
        continueAction()

def OD_callback(OD_detections):
    # Get bolt and bolt_hole pose
    bolt_id = detections.find_object("bolt")
    bolt_pose = detections.get_pose(bolt_id)
    bolt_hole_id = detections.find_object("bolt_hole")
    bolt_hole_pose = detections.get_pose(bolt_hole_id)
    # Call pick and place action
    Pick_and_Place(bolt_pose,bolt_hole_pose)

if __name__ == '__main__':
    # subscribe to action_recognition topic
    rospy.Subscriber("/opendr/action_recognition",
         ↪ ObjectHypothesis, AR_callback)
    # subscribe to object_detection topic
    rospy.Subscriber("/opendr/object_detection",
         ↪ ObjectHypothesisWithPose, OD_callback)

    rospy.spin()
```

5. Results and Discussion

Results are described for each individual visual recognition module and for the utilization of the modules in human-robot collaboration experiments. Integration, limitations and future work are described in the discussion as well.

5.1. Visual recognition performance

Table 1 provides details of the different perception modules and their corresponding datasets for training and inference. In the case of human skeleton detection and human action recognition, pre-generated datasets were utilized, as these provided sufficient performance for detection. A disadvantage, however, was that the features for detection could not be changed and no additional classes could be added. We explain this in more detail for each recognition module.

Human skeleton detection

The human skeleton detection method (LightWeight OpenPose [36]) with pretrained model [22] achieves satisfactory detection performance, when assessed with a single person in the image. Fig. 3 depicts the skeleton detection and draws it over the person in the scene.

[5] https://labelstud.io/
[6] https://albumentations.ai/
[7] https://jupyter.org/

Table 1
Perception models and datasets utilized to enable human-robot collaboration. Performance is reported in terms of frames per second (FPS) and reported prediction accuracy on the dataset, from the original research papers.

	Training			Inference (GTX 1080 Ti)			
Perception module	**Method**	**Dataset**	**Dataset size**	**Model size**	**Image size**	**FPS**	**Prediction accuracy (%)**
Human skeleton detection	Lightweight OpenPose [36]	COCO 2017 [22]	25 GB	1.2 GB	1920x1080 1280x720 960x540	30 30 60	88
Human action recognition	ST-GCN [58]	NTU-RGB+D [45]	1.3 TB	47 MB	1920x1080 1280x720 960x540	24 30 31	83
Object and target detection	Detectron2 [56]	Custom	65 GB	0.5 GB	1920x1080 1280x720 960x540	2.6 4.5 6.0	-

In terms of computational performance, the module achieves 30 frames per second, for high resolution camera image input (1920 × 1080) and even higher for lower resolution images (see Table 1).

The industrial environment and the scenario of engine assembly leaves practical limitations on how the human skeleton detection tool can be utilized. For example, the camera cannot capture the human in full, but only the upper body. For human-robot collaborative tasks the detection of a person's left and right wrist was therefore chosen for the interaction, as these could be detected reliably, while allowing free motion in the entire camera view. The detection of both wrists in predefined areas in the image can then be utilized to trigger robot actions, and to halt and resume them. Requiring both detections simultaneously in both areas increased the robustness to false positive detection with a single wrist, when the person was doing assembly actions on the engine. A sequence of screenshots of human skeleton and wrist detection can be seen in Fig. 3 and Fig. 5.

Human action recognition

The human action recognition method (ST-GCN [58]) with pretrained model [45] achieves satisfactory recognition performance, when assessed with a single person performing actions in the image. Fig. 3 depicts the actions recognized and their confidence score printed on the image. As the action recognition tool utilizes skeleton detection, this is drawn over the image as well.

In terms of computational performance, the module achieves 24 frames per second, for high resolution camera image input (1920 × 1080) and even higher for lower resolution images (see Table 1).

Similar to human skeleton detection, the industrial scenario imposed limitations as the datasets for human action recognition mostly covers daily actions [45], not relevant for industrial tasks. However, few action classes were general enough to be also useful in human-robot collaborative scenarios and robust enough for reliable detection. These were the human actions of '*salute*' (ID:37), '*put the palms together*' (ID:38) and '*cross hands in front*' (ID:39), as can be seen in Fig. 3c and Fig. 3d.

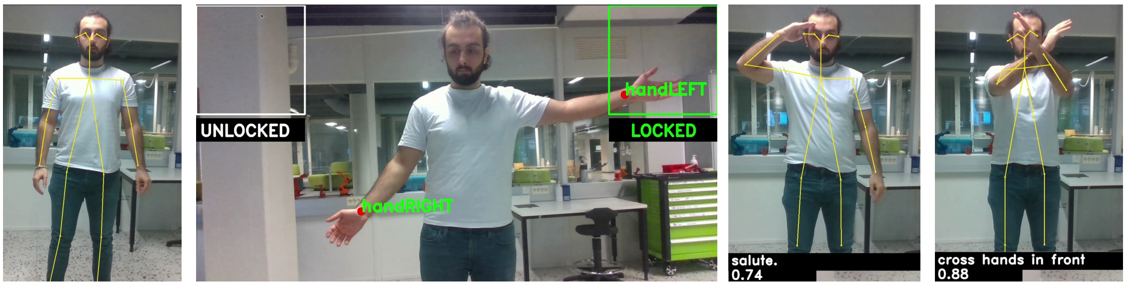
Object and target detection

The object and target detection method (Detectron2 [56]) with custom trained model achieves satisfactory performance, for non-overlapping objects. Fig. 4 depicts the objects detected on the table (a) and the targets detected on the engine (b). To create the dataset, 200 images of the eight objects and targets, in various configurations, were recorded and all objects and targets in the images were annotated with segmentation polygons in their correct class. Distractor objects, such as Diesel fuel lines, common rails and other tools, were included, as would be expected in a real scene. This data was then expanded with augmentations to a full datatset of around 300,000 images. Training of the model was done until convergence of the loss function (sum of losses due to classification and bounding box regression), which took around 20,000 epochs. With this method, the trained model achieved detection confidences for real camera images of more than 90%. While more data could be added and more training could be done, results are sufficient to perform reliable experiments for picking and placing, and human-robot collaboration.

In terms of computational performance, the module cannot run in real-time, but achieves 2.6 frames per second for high resolution camera input (1920 × 1080). As the objects and targets are static in the scene, real-time performance is not required. The implemented object and target detection tool enables both continuous detection (images are processed consecutively) and detection requests from a single image, with a function call. In the human-robot collaboration scenario a detection request is utilized to save computational performance of the GPU machine. It is expected, though, that both approaches would work equally well in terms of object pick and placement performance.

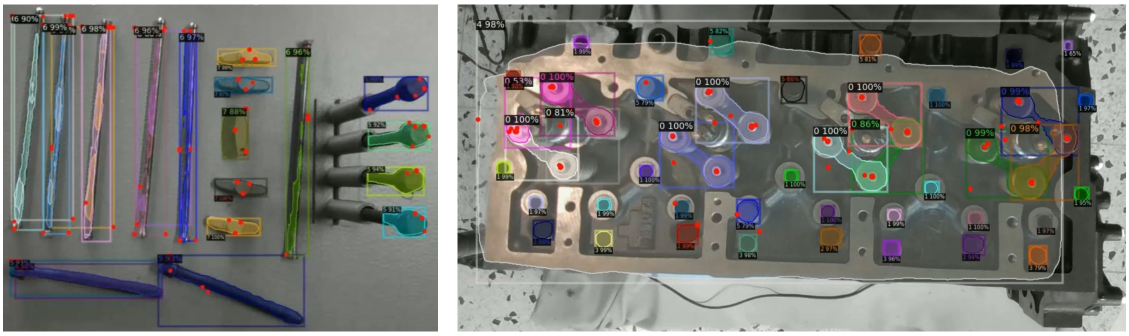
5.2. Human-robot collaboration

The visual perception modules were utilized to enable human-robot collaboration, in several different ways, with the detection modules utilized as interaction tool. Certain tools are more suited to specific tasks, due to their detection or computational performance. For example, human skeleton detection is very reliable and fast, while human



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

Figure 3: Human visual recognition modules. (a) and (b) depict results of human skeleton detection with the skeleton-based tracker Lightweight OpenPose [36]. (b) demonstrates that skeleton detection can be used for human-robot collaboration by detecting human wrists (handLeft and handRight) in certain image areas. (c) and (d) depict results of human action recognition with the real-time skeleton-based human action recognition framework ST-GCN [58]. Recognized actions are '*salute*' (c) and '*cross hands in front*' (d), with their corresponding confidence score.



(a) (b)

Figure 4: Results of visual perception for object and target detection utilizes Detectron2 [56]. (a) depicts detection of objects (three classes): rocker arms, bolts and pushrods, and (b) depicts detection of targets (five classes): engine, bolt holes, pushrod holes and rocker arm location. Each detection is labeled with the detected class and their corresponding confidence score.

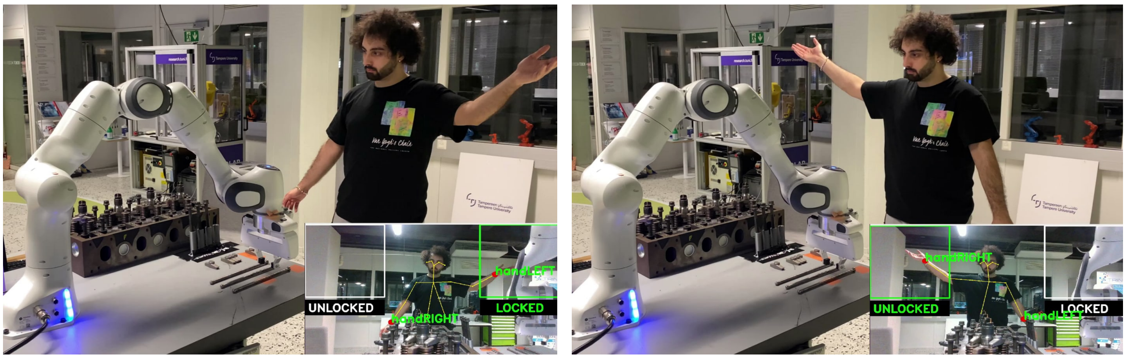
action recognition is less reliable and slower. This time performance difference is due to the fact that human action recognition relies on the human skeleton detection as input and requires a considerable number of detected frames (300) for successful recognition. In practise this means that human action recognition has more false detections as well. The following experiments were tested in detail.

Human safety

The functionality of human safety can be enabled by only allowing robot motion when the hands of a person are removed from the scene. As the human skeleton detection tools returns a set of skeleton nodes, the wrist nodes' locations in the image can be utilized for this. The detection of the human wrists in certain image areas is demonstrated in Fig. 3 and Fig 5. While the system is responsive after a true positive detection (delay of $\approx 33ms$), performance is sensitive to false negative detections, causing unnecessary delays and an unsafe system. Moreover, requiring the hands of an operator to be held up for extended time periods, many times per day, might be straining and uncomfortable.

Human task coordination

The shared assembly task can easily be coordinated by the human with visual perception. Human skeleton detection (i.e., wrists in certain location) or human actions can be used for starting and/or stopping robot actions, thereby setting the pace for the assembly task and performing corrective actions, in case a robot has misplaced a part. Human visual perception is not required to have high performance for this, as the detection tools can be run at a high rate (i.e., >30 FPS). This implies that few false negative detections have no significant negative impact in the collaboration. For the object and target detection tool, real-time performance is not required either, as pick and place actions are called on request. These coordination experiments, by human wrist detection, are depicted in Fig. 5 and Fig. 6. Robot actions are the assembly (pick and placement) of pushrods and bolts (six in total) to the Diesel engine and human actions are the placement of rocker arms, after their hand-over from the robot. A video of the experiments can be found here: `https://youtu.be/3z3yiLdznrY`



(a)

(b)

Figure 5: Results of human-robot collaboration experiments. (a) and (b) depict human task coordination by visual detection of the left wrist (handLeft), for halting the robot and performing manual assembly actions (a), followed by right wrist detection (handRight) for resuming robot actions (b).

Robot-human hand-overs

As explained in Section 4, certain tasks for assembling a Diesel engine are too difficult for a robot to execute. However, as assistive tool, the robot can hand-over parts located on a table to the person executing complex assembly tasks. This is demonstrated in Fig. 6a and Fig. 6b for the assembly tasks of rocker arm placement. The objects are detected with the same detection model and all detected parts are handed over in sequence to a hand-over point, close to the human. By human gestures (visual perception tools) the person can request for the initiation of the hand-over task (i.e., pick an object and move to the hand-over location) and trigger the actual hand-over action. After the rocker arm is handed over, the human can continue the assembly action, while the robot fetches another part.

In theory, human-robot collaboration by human coordination can improve the fluency of collaboration fluency measures [17]. This implies the reduction of idle time for both human and robot, as well as the robot's functional delay, leading to higher task efficiency. While this work serves to demonstrate the functionality of the visual perception modules, a thorough analysis and evaluation for fluency measures has not been carried out.

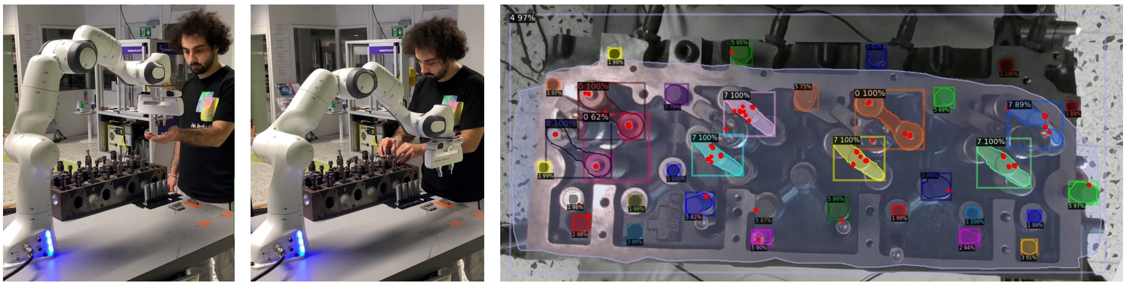
Assembly progress tracking

Besides enabling human-robot collaboration, the visual perception tools can also be used to track the progress of the Diesel engine assembly task. This means to track how many objects are placed in the correct location or whether some objects are missing. While there are many ways how this could be implemented, a simple but effective implementation was done as follows. As the entire engine block is detected as well, it can be easily checked whether certain assembly objects (rocker arms, pushrods and bolts) are detected inside the detected engine bounding box. For this, the image dataset included the images of assembly objects assembled on the engine. Output of the assembly progress tracking tool then returns the number of objects assembled and/or whether the task is completed or not. Fig. 6c depicts the detection of different objects (rocker arms, bolts) inside the detected Diesel engine bounding box. In this time instance, five of the eight rocker arms are placed and detected (class 7), while three rocker arm locations are detected (class 0) and thus empty. In addition, ten bolts are placed and detected (class 5), while eleven bolts holes are detected (class 1) and thus empty. It should be noted, however, that visual detection is not a fool-proof way to track assembly progress. In total, 22 bolts should be assembled to the engine block, meaning one bolt hole or bolt was not detected. This misdetection of the bolt can be seen in the center of image (Fig. 6c).

5.3. Discussion

Limitations - The first limitation of the explored perception modules relates to the relevance of the (training) data for industrial context. As most tools are developed for humans and objects in domestic or outdoor environments, success in other areas is not guaranteed. In certain cases this is not a major issues (e.g., humans look similar in a broad context), but in some cases it can be a problem, as classes are unsuitable (e.g., multi-human actions in a single human use case) or simply do not exist (e.g., novel objects or human actions to detect). One obvious solution to this would be to extend an existing dataset or create a new dataset from scratch, however, this is not a trivial task. Collecting data is complex, and expensive in resources and equipment, even when synthetic data generation approaches exist [48]. In this work, the data generation tools for object and target detection are open-source available through the OpenDR toolkit.

Utilizing perception tools for human safety, in particular by DNN-based visual perception models, is not recommended. The reaction time of a safety system, in order to stop robot motion, should be small, which cannot always be guaranteed. Some models used in this work can be executed in real-time (see Table 1), and even faster (60 FPS), meaning that it takes at least 17ms for a detection, assuming



(a) (b) (c)

Figure 6: Results of robot-human hand-over and assembly tracking experiments. (a) depicts the hand-over of a rocker arm from robot to human. (b) depicts the human assembly action of the rocker arm by the human, while the robot fetches another rocker arm. (c) depicts the assembly tracking results, with several objects (rocker arms, class 7; bolts, class 5) and their locations (rocker arm location, class 0; bolt holes, class 1) detected inside the detected Diesel engine bounding box (class 4). One bolt, however, is not detected at all. Each detection is labeled with the detected class and their corresponding confidence score.

a prediction is accurately made. Other models are simply not suited for fast detection or recognition, as they require a set of images, instead of single images (e.g., 300 in the case of [58]) and/or rely on another detection tool as input (e.g., skeleton detection in the case of [58]). In addition, as reported in [18], quantifying the reliability of machine learning and DNN-based perception tools is still a challenge and performance might drift over time. The time-delay of perception and its performance uncertainty should then be taken into account when calculating the minimum separation distance between human and robot.

Integration effort - The resources and effort needed to develop, train and deploy perception models for industrial use, is considerable. Even when robust and reliable pretrained models are to be integrated, still effort is needed to comply tools to existing software frameworks with its own datatypes and formatting. While ROS[2] has taken first steps to enable this for robotics, computer vision tools are typically disconnected from this. OpenDR [37] has made efforts to integrate a variety of perception models into ROS, and examples to specific use cases are presented in this work. In the case when pretrained models are not sufficient, additional effort is needed for data collection and training. As it is difficult to estimate how much effort is needed for different models, we report the effort for our custom dataset for object and target detection. A collection of 200 RGB images where taken as base for the dataset and annotations were needed for eight object and target classes. This annotation took considerable time (2-3 days) for the relatively small set of images. Generation of the complete dataset and training a model is time-consuming as well (2 hours for a single training cycle), and optimizing to good results requires expertise. Naturally, better performance can be obtained with more powerful computational hardware (e.g., computing cluster or cloud computing), however, these are not always available, and come with additional cost.

Future work - The results of our work demonstrate that deep learning-based perception models can be easily trained and deployed to robotic environments and achieve reliable detection and recognition results. Results also demonstrated that multiple perception models can be utilized simultaneously, enabling the fusion of different sensors or utilizing different detection modules in parallel. As such, this work has established a baseline for future directions. These include the fusion of different sensor information, from similar or dissimilar modalities. This sensor fusion would enable a higher robustness then single sensor models and introduces a redundancy of sensing, for example, in case one sensor fails or is occluded. Exploration of these topics will be done as future work.

6. Conclusions

Visual perception is a common tool for enabling human-robot collaboration, by detection or recognition of relevant objects, features and actions in the scene. The performance and maturity of such tools are usually evaluated by scenarios not related to robotics or manufacturing, limiting their direct utilization in industrial environments. Moreover, in some cases visual perception tools need to be tailored to suit the context of the human-robot collaboration scenario. This means collecting, annotating and augmenting visual data and the training of a perception model.

In this work we have identified these common issues and provide the practical integration details for three different deep learning-based visual perception tools. These are human skeleton detection, human action recognition, and object and target detection in context of the industrial use case of Diesel engine assembly. The tools are integrated open-source in the OpenDR toolkit, with ROS as software platform, providing templates for perception, robot actions and human-robot collaboration, thereby enabling to easily replicate and extend our work.

Declaration of competing interest

There is no known conflict of interest.

Acknowledgements

Project funding was received from European Union's Horizon 2020 research and innovation programme, grant no. 871449 (OpenDR) and no. 825196 (TRINITY).

References

[1] Ahmadyan, A., Zhang, L., Ablavatski, A., Wei, J., Grundmann, M., 2021. Objectron: A large scale dataset of object-centric videos in the wild with pose annotations, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7822–7831.

[2] Ajaykumar, G., Steele, M., Huang, C.M., 2022. A survey on end-user robot programming. ACM Computing Surveys 54, 1–36. doi:10.1145/3466819.

[3] Alćer, V., Cruz-Machado, V., 2019. Scanning the industry 4.0: A literature review on technologies for manufacturing systems. Engineering Science and Technology, an International Journal 22, 899–919. doi:10.1016/j.jestch.2019.01.006.

[4] Angleraud, A., Sefat, A.M., Netzev, M., Pieters, R., 2021. Coordinating shared tasks in human-robot collaboration by commands. Frontiers in Robotics and AI 8. doi:10.3389/frobt.2021.734548.

[5] Arrieta, A.B., Díaz-Rodríguez, N., Ser, J.D., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., Herrera, F., 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion 58, 82–115. doi:10.1016/j.inffus.2019.12.012.

[6] Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., Sheikh, Y.A., 2019. OpenPose: Realtime multi-person 2D pose estimation using part affinity fields. IEEE Transactions on Pattern Analysis and Machine Intelligence .

[7] Capellen, C., Schwarz, M., Behnke, S., 2019. ConvPoseCNN: Dense convolutional 6D object pose estimation. arXiv preprint arXiv:1912.07333 .

[8] Chen, C., Wang, T., Li, D., Hong, J., 2020. Repetitive assembly action recognition based on object detection and pose estimation. Journal of Manufacturing Systems 55, 325–333. doi:10.1016/j.jmsy.2020.04.018.

[9] Chowdhury, A., Ahtinen, A., Pieters, R., Vaananen, K., 2020. User experience goals for designing industrial human-cobot collaboration, in: Proceedings of the 11th Nordic Conference on Human-Computer Interaction: Shaping Experiences, Shaping Society, ACM. pp. 1–13. doi:10.1145/3419249.3420161.

[10] Dallel, M., Havard, V., Baudry, D., Savatier, X., 2020. Inhard - industrial human action recognition dataset in the context of industrial collaborative robotics, in: IEEE International Conference on Human-Machine Systems (ICHMS), pp. 1–6. doi:10.1109/ICHMS49158.2020.9209531.

[11] Du, G., Wang, K., Lian, S., Zhao, K., 2021. Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review. Artificial Intelligence Review 54, 1677–1734. doi:10.1007/s10462-020-09888-5.

[12] Everingham, M., Gool, L.V., Williams, C.K.I., Winn, J., Zisserman, A., 2010. The Pascal visual object classes (VOC) challenge. International Journal of Computer Vision 88, 303–338. doi:10.1007/s11263-009-0275-4.

[13] Evjemo, L.D., Gjerstad, T., Grøtli, E.I., Sziebig, G., 2020. Trends in smart manufacturing: Role of humans and industrial robots in smart factories. Current Robotics Reports 1, 35–41. doi:10.1007/s43154-020-00006-5.

[14] Fan, J., Zheng, P., Li, S., 2022. Vision-based holistic scene understanding towards proactive human–robot collaboration. Robotics and Computer-Integrated Manufacturing 75, 102304. doi:10.1016/j.rcim.2021.102304.

[15] Hietanen, A., Changizi, A., Lanz, M., Kamarainen, J., Ganguly, P., Pieters, R., Latokartano, J., 2019. Proof of concept of a projection-based safety system for human-robot collaborative engine assembly, in: IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), pp. 1–7. doi:10.1109/RO-MAN46459.2019.8956446.

[16] Hodan, T., Haluza, P., Obdrzalek, S., Matas, J., Lourakis, M., Zabulis, X., 2017. T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects, in: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE. pp. 880–888. doi:10.1109/WACV.2017.103.

[17] Hoffman, G., 2019. Evaluating fluency in human–robot collaboration. IEEE Transactions on Human-Machine Systems 49, 209–218.

[18] Jourdan, N., Sen, S., Husom, E.J., Garcia-Ceja, E., Biegel, T., Metternich, J., 2021. On the reliability of machine learning applications in manufacturing environments. arXiv preprint arXiv:2112.06986 .

[19] Ku, J., Mozifian, M., Lee, J., Harakeh, A., Waslander, S.L., 2018. Joint 3D proposal generation and object detection from view aggregation, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1–8. doi:10.1109/IROS.2018.8594049.

[20] Langer, M., Oster, D., Speith, T., Hermanns, H., Kästner, L., Schmidt, E., Sesing, A., Baum, K., 2021. What do we want from explainable artificial intelligence (XAI)? – a stakeholder perspective on XAI and a conceptual model guiding interdisciplinary XAI research. Artificial Intelligence 296, 103473. doi:10.1016/j.artint.2021.103473.

[21] Lavin, A., Gilligan-Lee, C.M., Visnjic, A., Ganju, S., Newman, D., Ganguly, S., Lange, D., Baydin, A.G., Sharma, A., Gibson, A., Zheng, S., Xing, E.P., Mattmann, C., Parr, J., Gal, Y., 2022. Technology readiness levels for machine learning systems. Nature Communications 13, 6039. doi:10.1038/s41467-022-33128-9.

[22] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014. Microsoft COCO: Common objects in context, in: European conference on computer vision (ECCV), Springer. pp. 740–755.

[23] Linder, T., Vaskevicius, N., Schirmer, R., Arras, K.O., 2021. Cross-modal analysis of human detection for robotics: An industrial case study, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 971–978. doi:10.1109/IROS51168.2021.9636158.

[24] Liu, H., Wang, L., 2018. Gesture recognition for human-robot collaboration: A review. International Journal of Industrial Ergonomics 68, 355–367. doi:10.1016/j.ergon.2017.02.004.

[25] Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., Pietikäinen, M., 2020. Deep learning for generic object detection: A survey. International Journal of Computer Vision 128, 261–318. doi:10.1007/s11263-019-01247-4.

[26] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C., 2016. SSD: Single shot multibox detector, in: European conference on computer vision, Springer. pp. 21–37.

[27] Magrini, E., Ferraguti, F., Ronga, A.J., Pini, F., Luca, A.D., Leali, F., 2020. Human-robot coexistence and interaction in open industrial cells. Robotics and Computer-Integrated Manufacturing 61, 101846. doi:10.1016/j.rcim.2019.101846.

[28] Marcus, G., 2018. Deep learning: A critical appraisal. arXiv preprint arXiv:1801.00631 .

[29] Marge, M., Espy-Wilson, C., Ward, N.G., Alwan, A., Artzi, Y., Bansal, M., Blankenship, G., Chai, J., Daumé, H., Dey, D., Harper, M., Howard, T., Kennington, C., Kruijff-Korbayová, I., Manocha, D., Matuszek, C., Mead, R., Mooney, R., Moore, R.K., Ostendorf, M., Pon-Barry, H., Rudnicky, A.I., Scheutz, M., Amant, R.S., Sun, T., Tellex, S., Traum, D., Yu, Z., 2022. Spoken language interaction with robots: Recommendations for future research. Computer Speech & Language 71, 101255. doi:10.1016/j.csl.2021.101255.

[30] Maschler, B., Pham, T.T.H., Weyrich, M., 2021. Regularization-based continual learning for anomaly detection in discrete manufacturing. Procedia CIRP 104, 452–457. doi:10.1016/j.procir.2021.11.076.

[31] Mavridis, N., 2015. A review of verbal and non-verbal human–robot interactive communication. Robotics and Autonomous Systems 63, 22–35. doi:10.1016/j.robot.2014.09.031.

[32] Mazhar, O., Navarro, B., Ramdani, S., Passama, R., Cherubini, A., 2019. A real-time human-robot interaction framework with robust background invariant hand gesture detection. Robotics and Computer-Integrated Manufacturing 60, 34–48. doi:10.1016/j.rcim.2019.05.008.

[33] Nguyen, D.T., Li, W., Ogunbona, P.O., 2016. Human detection from images and videos: A survey. Pattern Recognition 51, 148–175. doi:10.1016/j.patcog.2015.08.027.

[34] Nikolakis, N., Maratos, V., Makris, S., 2019. A cyber physical system (*cps*) approach for safe human-robot collaboration in a shared workplace. Robotics and Computer-Integrated Manufacturing 56, 233–243. doi:10.1016/j.rcim.2018.10.003.

[35] Nowruzi, F.E., Kapoor, P., Kolhatkar, D., Hassanat, F.A., Laganiere, R., Rebut, J., 2019. How much real data do we actually need: Analyzing object detection performance using synthetic and real data. arXiv preprint arXiv:1907.07061 .

[36] Osokin, D., 2018. Real-time 2D multi-person pose estimation on cpu: Lightweight openpose. doi:10.48550/ARXIV.1811.12004.

[37] Passalis, N., Pedrazzi, S., Babuska, R., Burgard, W., Dias, D., Ferro, F., Gabbouj, M., Green, O., Iosifidis, A., Kayacan, E., Kober, J., Michel, O., Nikolaidis, N., Nousi, P., Pieters, R., Tzelepi, M., Valada, A., Tefas, A., 2022. OpenDR: An open toolkit for enabling high performance, low footprint deep learning for robotics, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). doi:arXivpreprintarXiv:2203.00403.

[38] Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J., 2018. Frustum pointnets for 3D object detection from RGB-D data, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[39] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y., et al., 2009. ROS: an open-source robot operating system, in: ICRA workshop on open source software, Kobe, Japan. p. 5.

[40] Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[41] Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster R-CNN: Towards real-time object detection with region proposal networks, in: International Conference on Neural Information Processing Systems, p. 91–99.

[42] Robinson, N., Tidd, B., Campbell, D., Kulić, D., Corke, P., 2022. Robotic vision for human-robot interaction and collaboration: A survey and systematic review. Journal of Human-Robot Interaction doi:10.1145/3570731. just Accepted.

[43] Sampieri, A., D'Amely, G., Avogaro, A., Cunico, F., Skenderi, G., Setti, F., Cristani, M., Galasso, F., 2022. Pose forecasting in industrial human-robot collaboration. arXiv preprint arXiv:2208.07308 .

[44] Semeraro, F., Griffiths, A., Cangelosi, A., 2023. Human–robot collaboration and machine learning: A systematic review of recent research. Robotics and Computer-Integrated Manufacturing 79, 102432. doi:10.1016/j.rcim.2022.102432.

[45] Shahroudy, A., Liu, J., Ng, T.T., Wang, G., 2016. NTU RGB+D: A large scale dataset for 3D human activity analysis, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1010–1019.

[46] Sun, Z., Ke, Q., Rahmani, H., Bennamoun, M., Wang, G., Liu, J., 2022. Human action recognition from various data modalities: A review. IEEE Transactions on Pattern Analysis and Machine Intelligence , 1–20doi:10.1109/TPAMI.2022.3183112.

[47] Sünderhauf, N., Brock, O., Scheirer, W., Hadsell, R., Fox, D., Leitner, J., Upcroft, B., Abbeel, P., Burgard, W., Milford, M., Corke, P., 2018. The limits and potentials of deep learning for robotics. The International Journal of Robotics Research 37, 405–420. doi:10.1177/0278364918770733.

[48] Thalhammer, S., Patten, T., Vincze, M., 2019. SyDPose: Object detection and pose estimation in cluttered real-world depth images trained using only synthetic data, in: International Conference on 3D Vision (3DV), IEEE. pp. 106–115. doi:10.1109/3DV.2019.00021.

[49] Tu, H., Wang, C., Zeng, W., 2020. Voxelpose: Towards multi-camera 3D human pose estimation in wild environment, in: European Conference on Computer Vision (ECCV), Springer. pp. 197–212.

[50] Vargas, A.M., Cominelli, L., Dell'Orletta, F., Scilingo, E.P., 2021. Verbal communication in robotics: A study on salient terms, research fields and trends in the last decades based on a computational linguistic analysis. Frontiers in Computer Science 2. doi:10.3389/fcomp.2020.591164.

[51] Villani, V., Pini, F., Leali, F., Secchi, C., 2018. Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. Mechatronics 55, 248–266. doi:10.1016/j.mechatronics.2018.02.009.

[52] Wang, J., Ma, Y., Zhang, L., Gao, R.X., Wu, D., 2018a. Deep learning for smart manufacturing: Methods and applications. Journal of Manufacturing Systems 48, 144–156. doi:10.1016/j.jmsy.2018.01.003.

[53] Wang, L., Gao, R., Váncza, J., Krüger, J., Wang, X., Makris, S., Chryssolouris, G., 2019. Symbiotic human-robot collaborative assembly. CIRP Annals 68, 701–726. doi:10.1016/j.cirp.2019.05.002.

[54] Wang, P., Li, W., Ogunbona, P., Wan, J., Escalera, S., 2018b. Rgb-d-based human motion recognition with deep learning: A survey. Computer Vision and Image Understanding 171, 118–139. doi:10.1016/j.cviu.2018.04.007.

[55] Weiss, A., Wortmeier, A.K., Kubicek, B., 2021. Cobots in Industry 4.0: A roadmap for future practice studies on human–robot collaboration. IEEE Transactions on Human-Machine Systems 51, 335–345. doi:10.1109/THMS.2021.3092684.

[56] Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R., 2019. Detectron2. https://github.com/facebookresearch/detectron2.

[57] Wuest, T., Weimer, D., Irgens, C., Thoben, K.D., 2016. Machine learning in manufacturing: advantages, challenges, and applications. Production & Manufacturing Research 4, 23–45. doi:10.1080/21693277.2016.1192517.

[58] Yan, S., Xiong, Y., Lin, D., 2018. Spatial temporal graph convolutional networks for skeleton-based action recognition, in: Thirty-second AAAI conference on artificial intelligence.

[59] Yan, Z., Duckett, T., Bellotto, N., 2020. Online learning for 3D LiDAR-based human detection: experimental analysis of point cloud clustering and classification methods. Autonomous Robots 44, 147–164. doi:10.1007/s10514-019-09883-y.

[60] Yang, C., Zhu, Y., Chen, Y., 2022. A review of human–machine cooperation in the robotics domain. IEEE Transactions on Human-Machine Systems 52, 12–25. doi:10.1109/THMS.2021.3131684.

[61] Zacharaki, A., Kostavelis, I., Gasteratos, A., Dokas, I., 2020. Safety bounds in human robot interaction: A survey. Safety Science 127, 104667. doi:10.1016/j.ssci.2020.104667.

[62] Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J., Funkhouser, T., 2017. 3DMatch: Learning local geometric descriptors from RGB-D reconstructions, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1802–1811.

J PARTNR: Pick and Place Ambiguity Resolving by Trustworthy iNteractive leaRning

PARTNR: Pick and place Ambiguity Resolving by Trustworthy iNteractive leaRning

Jelle Luijkx   Zlatan Ajanović   Laura Ferranti   Jens Kober
Department of Cognitive Robotics
Delft University of Technology, The Netherlands
{J.D.Luijkx, Z.Ajanovic, L.Ferranti, J.Kober}@tudelft.nl

partnr-learn.github.io

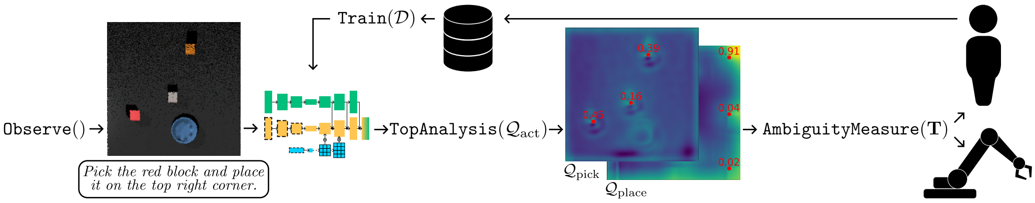
Abstract

Several recent works show impressive results in mapping language-based human commands and image scene observations to direct robot executable policies (e.g., pick and place poses). However, these approaches do not consider the uncertainty of the trained policy and simply always execute actions suggested by the current policy as the most probable ones. This makes them vulnerable to domain shift and inefficient in the number of required demonstrations. We extend previous works and present the PARTNR algorithm that can detect ambiguities in the trained policy by analyzing multiple modalities in the pick and place poses using topological analysis. PARTNR employs an adaptive, sensitivity-based, gating function that decides if additional user demonstrations are required. User demonstrations are aggregated to the dataset and used for subsequent training. In this way, the policy can adapt promptly to domain shift and it can minimize the number of required demonstrations for a well-trained policy. The adaptive threshold enables to achieve the user-acceptable level of ambiguity to execute the policy autonomously and in turn, increase the trustworthiness of our system. We demonstrate the performance of PARTNR in a table-top pick and place task.

1 Introduction

Despite the numerous exciting results in robot learning, only a few methods are actually robust enough to be employed in everyday life. Many manipulation tasks, such as pick-and-place in household scenarios, are challenging for robots, while they are actually easy for humans. To overcome this performance mismatch, we can exploit the human domain knowledge through (interactive) imitation learning [4]. This requires novel methods with an intuitive interface to transfer non-expert user knowledge to robotic systems. The impressive capabilities of recently introduced foundation models can possibly ease this transfer of knowledge. Foundation models can be trained on language data only (e.g., Transformers [17], BERT [7], or GPT-3 [3]) or can be trained on multi-modal data, such as images and their captions (e.g., CLIP [13]). In the field of robotics, language foundation models can be used for task planning [2] and interpreting human commands [16, 1] as well as corrections [15]. In particular, in the setting of (interactive) imitation learning, it is a natural choice to exploit language foundation models, since it allows the user to give instructions or corrections in an intuitive manner. Interactive imitation learning is a subclass of imitation learning in which the human is influencing the learning loop while executing the task [4]. To be practical and trustworthy, the robot should ask for help when it is uncertain about the outcome of its actions. At the same time, humans should not be bothered too much. In this work, we address this problem by introducing Pick and place Ambiguity Resolving by Trustworthy iNteractive leaRning (PARTNR). Our work is related to the seminal work that introduced dataset aggregation (DAgger) for imitation learning [14]. DAgger addresses the

NeurIPS 2022 Workshop on Robot Learning: Trustworthy Robotics, Virtual, Virtual



Train($\mathcal{D}$)
Observe()
Pick the red block and place it on the top right corner.
TopAnalysis($\mathcal{Q}_{\text{act}}$)
0.39
0.16
0.45
0.91
0.04
0.02
$\mathcal{Q}_{\text{pick}}$
$\mathcal{Q}_{\text{place}}$
AmbiguityMeasure($\mathbf{T}$)

Figure 1: PARTNR framework on an example task.

compounding errors in imitation learning caused by covariate shift through the collection of on-policy data. Many variants were introduced afterward, such as Human-Gated DAgger (HG-DAgger) [11], where the expert can take over control if deemed necessary, and Ensemble-DAgger[12], where the robot queries expert input when a novel or risky situation is faced. Regarding the ambiguity resolution, our work is related to LIRA [9] which treats ambiguities in discrete reference frames, while here we focus on actions. PARTNR can be used to interactively train vision-based pick and place models, such as Transporter networks [19] and its extension for language commands CLIPort [16]. PARTNR asks for a human demonstration in case the model predictions are ambiguous. We consider a prediction to be ambiguous if it results in multiple options with similar value estimates. To be trustworthy, the threshold of the gating function is adaptive and allows it to satisfy a user-defined sensitivity, balancing between potential failing and asking the user unnecessarily. PARTNR consists of two main steps: 1) Detecting ambiguity in pick and place heatmaps by finding multiple local maxima using topological persistence and query user demonstrations if needed. 2) Aggregating data from new human demonstrations in DAgger style to learn from feedback and resolve the ambiguity. PARTNR has several advantageous features compared to the other state-of-the-art approaches: 1) By querying a new demonstration based on the level of ambiguity, it avoids gathering demonstrations for situations that are already learned by the agent, therefore reducing the number of required demonstrations. 2) By specifying the desired sensitivity level, the user can set its preferred balance between the frequency of queries by the robot and the failure rate, therefore increasing the system's trustworthiness. 3) By gathering data during execution, PARTNR can adapt to changing environments as well as include *failure states*, i.e., new states visited by making mistakes, in the dataset so it can learn to recover from them. We demonstrate these on a simulated table-top robot pick and place task, where we show an improvement of the performance with respect to the baseline (CLIPort variant).

The rest of the paper is organized as follows. Section 2 presents preliminaries as seen in the works [19, 16] and lays the formal problem formulation for our work. Section 3 presents our method. This is followed by experiments and conclusion sections.

Additional material is available at: partnr-learn.github.io.

2 Preliminaries and Problem Definition

We follow [19, 16] and describe the pick and place problem as finding a mapping from an observation $\mathbf{o}_t$ to a pick and place action $\mathbf{a}_t$ at time step t, that is, $f(\mathbf{o}_t) \rightarrow \mathbf{a}_t = (\mathcal{T}_{\text{pick}}, \mathcal{T}_{\text{place}}) \in \mathcal{A}$, where $\mathcal{A}$ is the set of possible actions and $\mathcal{T}_{\text{pick}}$ and $\mathcal{T}_{\text{place}}$ are the end-effector pick and place poses, respectively. We consider a table-top pick and place problem with $\mathcal{T}_{\text{pick}}, \mathcal{T}_{\text{place}} \in \mathbb{R}^2$. Motion primitives can be used to obtain a sequence of lower-level actions from $\mathcal{T}_{\text{pick}}$ and $\mathcal{T}_{\text{place}}$. Furthermore, we consider vision-based manipulation with $\mathcal{T}_i \sim (u, v), i \in \{\text{pick}, \text{place}\}$, where (u, v) is a pixel location of a (projected) top view image. If we model the action-value functions $\mathcal{Q}_{\text{pick}}$ and $\mathcal{Q}_{\text{place}}$, the optimal pick and place locations according to the model are $\mathcal{T}_{\text{pick}} = \arg\max_{(u,v)} \mathcal{Q}_{\text{pick}}((u, v)|\mathbf{o}_t)$ and $\mathcal{T}_{\text{place}} = \arg\max_{(u,v)} \mathcal{Q}_{\text{place}}((u, v)|\mathbf{o}_t, \mathcal{T}_{\text{pick}})$. Note that $\mathcal{T}_{\text{place}}$ is conditioned on $\mathcal{T}_{\text{pick}}$. Normalized heatmaps correlated with pick and place success can be obtained using the $\mathrm{softmax}$ function, i.e., $\mathcal{V}_{\text{pick}} \in \mathbb{R}^{H \times W} = \mathrm{softmax}(\mathcal{Q}_{\text{pick}}((u, v)|\mathbf{o}_t))$, where H and W are the height and width of the top view image, respectively. The action-value functions can be estimated through imitation learning. We build on the standard imitation learning setting where we have a dataset $\mathcal{D} = \{\zeta_1, \zeta_2, ..., \zeta_n\}$, where n is the number of expert demonstration trajectories consisting of one or more tuples of observations and actions, i.e., $\zeta_i = \{(\mathbf{o}_0, \mathbf{a}_0), (\mathbf{o}_1, \mathbf{a}_1), ...\}$.

In this work, we extend previous problem formulation and consider situations when taking the most probable action by $\arg\max$ is not sufficient, e.g. when there is no single distinctive maximum. We tackle the interactive learning problem where the robot needs to hand over the control back to the human and learn from new human demonstrations.

3 PARTNR: Pick and place Ambiguity Resolving by Trustworthy iNteractive leaRning

PARTNR is an interactive imitation learning algorithm that asks the human to take over control in case it considers the situation to be ambiguous. The situation is ambiguous when the learned policy does not provide a single dominant solution, i.e., there are multiple local maxima with close values in the action space. User demonstrations are aggregated to the dataset $\mathcal{D}$ and used for subsequent training, as shown in Algorithm 1. The robot observes, at each execution step, a human-provided natural language command and the state of the environment (e.g., a top-view image of the table). Based on the observation, the policy provides the heatmap, representing the value of the action (e.g., $\mathcal{Q}_{\text{pick}}$ representing pick location). The heatmap ($\mathcal{Q}_{\text{pick}}$ and subsequently $\mathcal{Q}_{\text{place}}$) is then analyzed to detect multiple local maxima (in `TopAnalysis`). In this work, we rely on computational topology methods for finding local maxima [8]. Specifically we use a persistent homology method [10]. Then, in `AmbiguityMeasure`, the obtained corresponding values of the local maxima $\mathbf{T}$, are normalized using the softmax function and the maximum value $\hat{p}_{\text{act}}$ is then used to decide if the situation is ambiguous. If $\hat{p}_{\text{act}}$ is smaller than the threshold $p_{\text{act}}^{\text{thr}}$, the situation is ambiguous. In case the situation is ambiguous, the robot is not executing the policy but queries the human teacher. The threshold $p_{\text{act}}^{\text{thr}}$ is updated continuously, by the function `UpdateThreshold`, to satisfy a user-defined sensitivity value (more details in Appendix C). Whenever there is a teacher input, the data is aggregated and the policy is updated using the function `Train` (like in [14]).

Figure 1 shows the PARTNR framework in an example where a human asks the robot to pick the red block and place it in the top right corner. As we can see on the heatmaps for pick and place, there are multiple local maxima for this command. In the pick heatmap $\mathcal{Q}_{\text{pick}}$ there are at least three local maxima, each related to one of the blocks. The maximum related to the red block is the highest (0.45). However, the orange block is also relatively close (0.39). In this case, the situation might be ambiguous (depending on the sensitivity level) and the robot might query the teacher for a demonstration.

4 Experiments and Results

We evaluated the performance of the proposed method in a simulated table-top pick and place task, which is shown in Figure 2. This task is very similar to tasks from [16, 19, 18]. The goal of this task

Algorithm 1: PARTNR

```
output : $\mathcal{Q}_{\text{pick}}, \mathcal{Q}_{\text{place}}$                         // pick and place value functions
begin
    $\mathcal{D}, \mathcal{Q}_{\text{pick}}, \mathcal{Q}_{\text{place}}, p_{\text{pick}}^{\text{thr}}, p_{\text{place}}^{\text{thr}} \leftarrow$ init()                         // initialization
    for $t \leftarrow 0$ to $t_{\max}$ do // while experiment runs
        $\mathbf{o}_t \leftarrow$ Observe()
        foreach act $\in \{\text{pick}, \text{place}\}$ do
            $\mathbf{T} = \{(u_1, v_1), \ldots, (u_k, v_k)\} \leftarrow$ TopAnalysis($\mathcal{Q}_{\text{act}}((u, v)|\mathbf{o}_t)$)          // $k$ local maxima
            $\hat{p}_{\text{act}} \leftarrow$ AmbiguityMeasure($\mathbf{T}$)
            if $\hat{p}_{\text{act}} \leq p_{\text{act}}^{\text{thr}}$ then // if ambiguous
                $\mathbf{a}_t \leftarrow$ QueryTeacher($\mathbf{T}$)
                Act($\mathbf{a}_t$), $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{o}_t, \mathbf{a}_t)$                // adding user input to the Dataset
            else // if not ambiguous
                Act($\arg\max_{(u,v)\in\mathbf{T}} \mathcal{Q}_{\text{act}}((u, v)|\mathbf{o}_t)$)
                if $\mathbf{a}_{\text{corr}} \leftarrow$ ObserveCorrection() $\neq \varnothing$ then // if teacher corrects
                    $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{o}_t, \mathbf{a}_{\text{corr}})$
            $p_{\text{act}}^{\text{thr}} \leftarrow$ UpdateThreshold($p_{\text{act}}^{\text{thr}}$), $\mathcal{Q}_{\text{act}} \leftarrow$ Train($\mathcal{D}$)        // update the model with new data
```

is to execute language commands in the form "*Pick the [pick color] box and place it in the [place color] bowl.*", where the pick and place colors are sampled at the beginning of an episode, based on the colors of the objects present in the scene. The task is simulated using the PyBullet simulator [6] and the implementation is adapted from [18]. At the beginning of each episode, three boxes and three bowls are placed at random locations on the table. Similar to [16], the colors of the objects are either sampled from the color set of $\mathcal{C}_{\text{all}} \cup \mathcal{C}_{\text{seen}}$ or the color set $\mathcal{C}_{\text{all}} \cup \mathcal{C}_{\text{unseen}}$, where $\mathcal{C}_{\text{all}} = \{\text{red, blue, green}\}$, $\mathcal{C}_{\text{seen}} = \{\text{yellow, brown, gray, cyan}\}$, and $\mathcal{C}_{\text{unseen}} = \{\text{orange, purple, pink, white}\}$. The set of seen colors is used for offline training, while both sets are used for evaluation and interactive learning, to simulate a domain shift.

As a baseline, the CLIPort variant used in [18, 2] is employed and trained on a dataset consisting of demonstrations from a scripted expert. We also trained CLIPort models using the PARTNR algorithm with the same architecture as the baseline interactively, as described in Algorithm 1. The interactive models are initially trained offline and updated interactively while executing the task. To have a fair comparison between the baseline and the interactive models, both have the same number of total demonstrations and a total number of model updates. Since real-life demonstrations are never perfect, we also evaluated the method with noisy demonstrations.

We follow [16] and evaluate each model in 100 episodes consisting of three pick-and-place commands. The percentage of successfully performed pick and place commands is used as evaluation metric. The results in Table 1 show that the PARTNR algorithm improves the baseline performance, both in the in-distribution and out-of-distribution scenarios (seen and unseen case). The improvement in the unseen case indicates that by collecting on-policy data, the methods improves robustness against domain shifts. The on-policy data also enables to learn to recover from failure states, which is not possible in the offline baseline. Interestingly, both the baseline and PARTNR performance improved substantially when adding noise to the pick and place demonstrations from the scripted expert. To be noted, the final performance is lower than obtained with the original CLIPort model in [16]. Most likely, this is due to the usage of a simplified variant, a lower number of data augmentations and a lower number of camera perspectives. However, this is not relevant as the main focus here is to make a comparison against a non-interactive baseline, and not to obtain optimal performance.

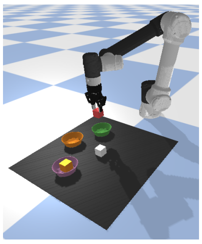


Figure 2: The *put-blocks-in-bowls* task.

algorithm	data split	put-blocks-in-bowls seen-colors 500	1000	1000 noisy	1500	put-blocks-in-bowls unseen-colors 500	1000	1500
Baseline	100% off	28.3	51.7	82.7	62.7	19.0	22.0	16.7
PARTNR	50% off + 50% int	**30.3**	**57.3**	**91.0**	**80.3**	**30.7**	**53.0**	**78.3**
PARTNR (80% data)	50% off + 30% int	28.0	39.3	77.7	68.0	20.3	28.3	57.3

Table 1: The performance of the PARTNR algorithm is evaluated against the performance of the non-interactive baseline (CLIPort variant). Here the success rate (%) is shown for a number of demonstrations, i.e., 500, 1000 and 1500. The data split indicates the percentage of demonstrations that were obtained offline (off) and interactively (int), so in the last row, 20% fewer demonstrations were collected. Since real demonstrations are often noisy, we also evaluated both methods with noise ($\sim \mathcal{N}(0, 3^2)$ pixels) added to the pick and place locations (1000 noisy).

5 Conclusions and Outlook

This work introduced PARTNR, an interactive learning algorithm for resolving ambiguities in pick-and-place tasks. The PARTNR algorithm improves the baseline performance, both in the in-distribution and out-of-distribution scenarios. Furthermore, sampling efficiency is improved (even up to 20% more data-efficient), since demonstrations are only collected when needed, based on the user-specified sensitivity. In the future, we plan to evaluate PARTNR with the original CLIPort baseline as well and to further address the epistemic uncertainty of the model, e.g., through an ensemble approach. Also, we wish to extend the method with sequence prediction and feedback control. Finally, we plan to monitor the human cognitive load in a real-world participant study.

Acknowledgments

This work was supported by the European Union's H2020 project Open Deep Learning Toolkit for Robotics (OpenDR) under grant agreement #871449 and by the ERC Stg TERI, project reference #804907.

References

[1] Hyemin Ahn, Obin Kwon, Kyungdo Kim, Jaeyeon Jeong, Howoong Jun, Hongjung Lee, Dongheui Lee, and Songhwai Oh. Visually Grounding Language Instruction for History-Dependent Manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 675–682, 2022.

[2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, and Mengyuan Yan. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances, 2022.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[4] Carlos Celemin, Rodrigo Pérez-Dattari, Eugenio Chisari, Giovanni Franzese, Leandro de Souza Rosa, Ravi Prakash, Zlatan Ajanović, Marta Ferraz, Abhinav Valada, and Jens Kober. Interactive imitation learning in robotics: A survey. *arXiv preprint arXiv:2211.00600*, 2022.

[5] Kevin Chu. An introduction to sensitivity, specificity, predictive values and likelihood ratios. *Emergency Medicine*, 11(3):175–181, 1999.

[6] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. American Mathematical Soc., 2010.

[9] Giovanni Franzese, CE Celemin, and Jens Kober. Learning interactively to resolve ambiguity in reference frame selection. In *Conference on Robot Learning (CoRL)*, 2020.

[10] Stefan Huber. Persistent homology in data science. In *Data Science–Analytics and Applications*, pages 81–88. Springer, 2021.

[11] Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J. Kochenderfer. HG-DAgger: Interactive Imitation Learning with Human Experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8077–8083, 2019.

[12] Kunal Menda, Katherine Driggs-Campbell, and Mykel J. Kochenderfer. EnsembleDAgger: A Bayesian Approach to Safe Imitation Learning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5041–5048, 2019.

[13] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[14] Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[15] Pratyusha Sharma, Balakumar Sundaralingam, Valts Blukis, Chris Paxton, Tucker Hermans, Antonio Torralba, Jacob Andreas, and Dieter Fox. Correcting robot plans with natural language feedback. *arXiv preprint arXiv:2204.05186*, 2022.

[16] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021.

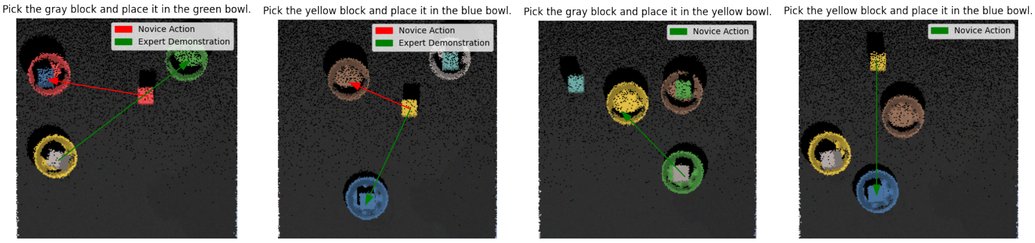
[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[18] Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv*, 2022.

[19] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*, pages 726–747. PMLR, 2021.

A Recovering from mistake

Because the PARTNR algorithm collects data from the state distribution induced by the novice policy, it can learn to recover after mistakes, while this is not the case when learning offline from expert demonstrations. That is to say, the expert does not make any mistakes and therefore failure states are not visited by the expert policy. An example of such a recovery learned interactively is shown in Figure 3.


Pick the gray block and place it in the green bowl.
🟥 Novice Action
🟩 Expert Demonstration
Pick the yellow block and place it in the blue bowl.
🟥 Novice Action
🟩 Expert Demonstration
Pick the gray block and place it in the yellow bowl.
🟩 Novice Action
Pick the yellow block and place it in the blue bowl.
🟩 Novice Action

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

Figure 3: Example of learning how to recover thanks to on-policy data collection. The figures **(a)** and **(b)** show demonstrations for failure states, i.e., in **(a)** the block to be picked is already in another bowl, and in **(b)** there is already a block in the blue bowl. Such demonstrations of failure states are collected only in the interactive case. Figures **(c)** and **(d)** show that the novice can learn to recover from such failure states.

B PARTNR framework: Detailed algorithm

A detailed version of the PARTNR algorithm is shown in Algorithm 2. For computing the sensitivity or optionally specificity [5], we also need to keep track of the true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). The definition of positives and negatives is shown in Table 2.

Table 2: Definition of positives and negatives. In the ideal case, the teacher is only queried in the case that the robot's action would result in a failure (true positive).

		Human input was necessary	
		True	False
Ambiguous	True	True Positive (TP)	False Positive (FP)
	False	False Negative (FN)	True Negative (TN)

C Adaptive Threshold: More details

A detailed version of the adaptive threshold algorithm is shown in Algorithm 3. Here, the number of true positives, true negatives, false positives, and false negatives are counted over a window ($k_{\mathrm{TP}}, k_{\mathrm{TN}}, k_{\mathrm{FP}}, k_{\mathrm{FN}}$, respectively). Subsequently, the sensitivity can be estimated ($\hat{s}$) [5]. Finally, the threshold p^{thr} is updated proportionally to the error between the desired and estimated sensitivity. In our experiments, we used the following values: $p_0^{\mathrm{thr}} = 0.5$, $s_{\mathrm{des}} = 0.9$, $w_n = 50$ and $a = 0.005$.

Algorithm 2: PARTNR - detailed algorithm

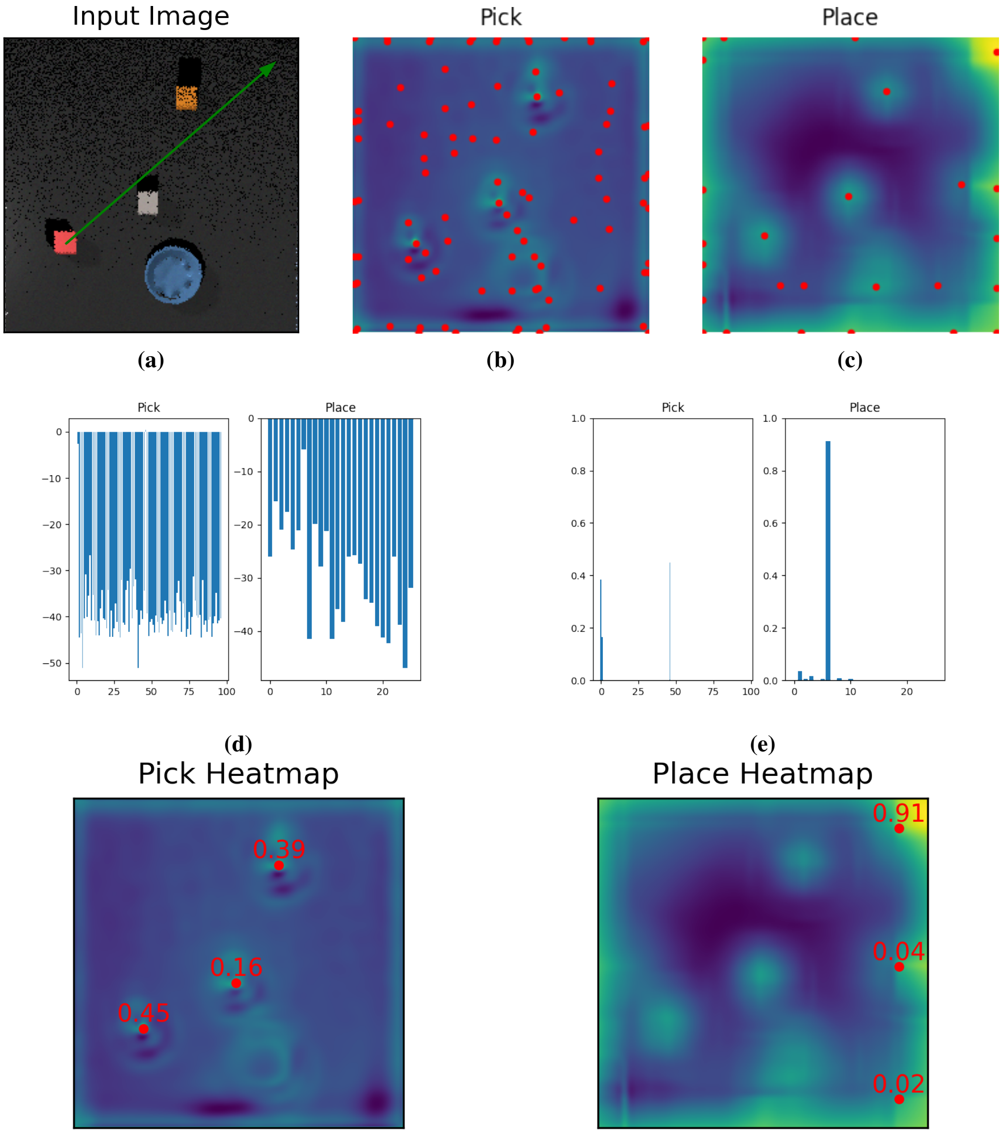
```
input  : $\mathcal{D}^{\text{init}}$                                   // initial demonstrations
output : $\mathcal{Q}_{\text{pick}}, \mathcal{Q}_{\text{place}}$                         // pick and place value functions
begin
    $\mathcal{D} \leftarrow \mathcal{D}^{\text{init}}$                                  // initial Dataset
    $\mathcal{Q}_{\text{pick}}, \mathcal{Q}_{\text{place}} \leftarrow$ Train($\mathcal{D}^{\text{init}}$)
    $\text{TP}, \text{TN}, \text{FP}, \text{FN} \leftarrow \varnothing$
    $p^{\text{thr}}_{\text{pick}}, p^{\text{thr}}_{\text{place}} \leftarrow$ init()
    for $t \leftarrow 0$ to $t_{\max}$ do // while experiment runs
        $\mathbf{o}_t \leftarrow$ Observe()
        foreach act $\in \{\text{pick}, \text{place}\}$ do
            isUpdated $\leftarrow$ false
            $\mathbf{T} = \{(u_1, v_1), \ldots, (u_k, v_k)\} \leftarrow$ TopAnalysis($\mathcal{Q}_{\text{act}}((u, v)|\mathbf{o}_t)$)
            $\mathbf{a}_{\max} \leftarrow \arg\max_{(u,v) \in \mathbf{T}} \mathcal{Q}_{\text{act}}(u, v)$
            $\hat{p}_{\text{act}} \leftarrow$ AmbiguityMeasure($\mathbf{T}$)
            if $\hat{p}_{\text{act}} \leq p^{\text{thr}}_{\text{act}}$ then // if ambiguous
                $\mathbf{a}_t \leftarrow$ QueryTeacher($\mathbf{T}$)
                $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{o}_t, \mathbf{a}_t)$            // adding user input to the Dataset
                isUpdated $\leftarrow$ true
                if $\mathbf{a}_t == \mathbf{a}_{\max}$ then
                    $\text{FP} \leftarrow \text{FP} \cup t$                    // adding False Positive flag
                else
                    $\text{TP} \leftarrow \text{TP} \cup t$                     // adding True Positive flag
                Act($\mathbf{a}_t$)
            else // if not ambiguous
                Act($\mathbf{a}_{\max}$)
                if $\mathbf{a}_{\text{corr}} \leftarrow$ ObserveCorrection() $\neq \varnothing$ then // if teacher corrects
                    $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{o}_t, \mathbf{a}_{\text{corr}})$
                    isUpdated $\leftarrow$ true
                    $\text{FN} \leftarrow \text{FN} \cup t$                    // adding False Negative flag
                else
                    $\text{TN} \leftarrow \text{TN} \cup t$                     // adding True Negative flag
            $p^{\text{thr}}_{\text{act}} \leftarrow$ UpdateThreshold($p^{\text{thr}}_{\text{act}}, \text{TP}, \text{TN}, \text{FP}, \text{FN}$)
            if isUpdated then
                $\mathcal{Q}_{\text{act}} \leftarrow$ Train($\mathcal{D}$)            // update the model with new data
```

Algorithm 3: UpdateThreshold

```
input  : Initial threshold $p^{\text{thr}}_0$, Desired sensitivity $s_{\text{des}}$, Window length $w_n$, Adaptation rate $a$.
output : threshold $p^{\text{thr}}$
begin
    $k_{\text{TP}}, k_{\text{TN}}, k_{\text{FP}}, k_{\text{FN}} \leftarrow$ MovHorCnt($w_n, \text{TP}, \text{TN}, \text{FP}, \text{FN}$)    // Counting occurrence
      in the window $w_n$
    $\hat{s} \leftarrow \frac{k_{\text{TP}}}{k_{\text{TP}} + k_{\text{FN}}}$
    $p^{\text{thr}} \leftarrow p^{\text{thr}}_0 - a \cdot (s_{\text{des}} - \hat{s})$
```

D Ambiguity Measure: Example and visualization

Figure Figure 4 shows a visual example of how the ambiguity measure is obtained.



(a) **(b)** **(c)**

(d) **(e)**

(f) **(g)**

Figure 4: This figure shows a visual example of obtaining the ambiguity measure. The input image together with the correct action (green arrow) is shown in **(a)**. Here, the language command is: '*Pick the red block and place it on the top right corner*". With `TopAnalysis`, we obtain local maxima $\mathbf{T}$, which are shown in **(b)** and **(c)** for the pick and place poses, respectively. The corresponding values are shown in **(d)**. After normalization using the $\mathrm{softmax}$ function, we obtain **(e)**. The local maxima with a normalized value greater than 0.01 are shown in **(f)** and **(g)** for the pick and place poses, respectively. The maximum of the normalized values is used as ambiguity measure.