



OpenDR — Open Deep Learning Toolkit for Robotics

Project Start Date: 01.01.2020

Duration: 48 months

Lead contractor: Aristotle University of Thessaloniki

**Deliverable D5.4: Final report on deep robot
action and decision making**

Date of delivery: 29 September 2023

Contributing Partners: TUD, ALU-FR, TAU, AUTH, AU
Version: v2.0



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

Title	D5.4: Final 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), Avramelou Loukia (AUTH), Kakaletsis Efstratios (AUTH), Passalis Nikolaos (AUTH), Kirtas Emmanouil (AUTH), Nousi Paraskevi (AUTH), Tzelepi Maria (AUTH), Symeonidis Charalampos (AUTH), Spanos Dimitrios (AUTH), Tefas Anastasios (AUTH), Nikolaidis Nikolaos (AUTH), Halil Ibrahim Ugurlu (AU), Amir Ramezani Dooraki (AU), Erdal Kayacan (AU), Alexandros Iosifidis (AU), Daniel Honerkamp (ALU-FR), Nikolai Dorka (ALU-FR), Tim Welschehold (ALU-FR), Abhinav Valada (ALU-FR), Roel Pieters (TAU), Akif Ekrekli (TAU), Mikael Petäjä (TAU)
Lead Beneficiary	TUD (Technische Universiteit Delft)
WP	5
Doc ID:	OPENDR_D5.4.pdf

Document History

Version	Date	Reason of change
v1.0	22/9/2023	Draft ready for review
v2.0	25/9/2023	Final version

Contents

1	Introduction	7
1.1	Deep Planning (T5.1)	7
1.1.1	Objectives	7
1.1.2	Innovations and achieved results	7
1.1.3	Ongoing and future work	7
1.2	Deep Navigation (T5.2)	7
1.2.1	Objectives	7
1.2.2	Innovations and achieved results	7
1.2.3	Ongoing and future work	8
1.3	Deep Action and Control (T5.3)	8
1.3.1	Objectives	8
1.3.2	Innovations and achieved results	8
1.3.3	Ongoing and future work	8
1.4	Human Robot Interaction (T5.4)	9
1.4.1	Objectives	9
1.4.2	Innovations and achieved results	9
1.4.3	Ongoing and future work	9
1.5	Connection to Project Objectives	10
2	Deep Planning	11
2.1	Lyapunov-inspired deep reinforcement learning for obstacle avoidance	11
2.1.1	Introduction and objectives	11
2.1.2	Description of work performed so far	12
2.1.3	Future work	12
2.2	Curiosity-Driven Reinforcement Learning based Low-Level Flight Control	12
2.2.1	Introduction and objectives	12
2.2.2	Description of work performed so far	13
2.2.3	Future work	13
3	Deep Navigation	13
3.1	Learning Hierarchical Interactive Multi-Object Search for Mobile Manipulation	13
3.1.1	Introduction and objectives	13
3.1.2	Description of work performed so far	14
3.1.3	Future work	14
3.2	Deep Reinforcement Learning with Action Masking for Differential-drive Robot Navigation using Low-Cost Sensors	14
3.2.1	Introduction and objectives	14
3.2.2	Description of work performed so far	15
3.2.3	Future Work	16
3.3	Improving Inertial-based UAV Localization using Data-efficient Deep Reinforcement Learning	16
3.3.1	Introduction and work performed so far	16
3.3.2	Future Work	16

4	Deep action and control	16
4.1	EAGERx: Graph-Based Framework for Sim2real Robot Learning	16
4.1.1	Introduction and objectives	16
4.1.2	Description of work performed so far	17
4.1.3	Future work	17
4.2	Prioritizing States with Action Sensitive Return in Experience Replay	17
4.2.1	Introduction and objectives	17
4.2.2	Description of work performed so far	17
4.2.3	Future work	18
5	Human robot interaction	18
5.1	EValueAction: a proposal for policy evaluation in simulation to support inter- active imitation learning	18
5.1.1	Introduction and objectives	18
5.1.2	Description of work performed so far	18
5.1.3	Future work	19
5.2	Sensor-based Human-Robot Collaboration for Industrial Tasks	19
5.2.1	Introduction and objectives	19
5.2.2	Description of work performed so far	19
5.2.3	Future work	20
5.3	Co-speech Gestures for Human-Robot Collaboration	20
5.3.1	Introduction and objectives	20
5.3.2	Description of work performed so far	20
5.3.3	Future work	20
6	Conclusions	21
A	Lyapunov-inspired deep reinforcement learning for robot navigation in obstacle environments	25
B	Curiosity-Driven Reinforcement Learning based Low-Level Flight Control	32
C	Deep Reinforcement Learning with Action Masking for Differential-drive Robot Navigation using Low-Cost Sensors	45
D	Data efficient Deep Reinforcement Learning for Robust Inertial-based UAV Local- ization	52
E	Learning Hierarchical Interactive Multi-Object Search for Mobile Manipulation	73
F	EAGERx: Graph-Based Framework for Sim2real Robot Learning	84
G	Prioritizing States with Action Sensitive Return in Experience Replay	93
H	EValueAction: a proposal for policy evaluation in simulation to support interactive imitation learning	106
I	Sensor-Based Human-Robot Collaboration for Industrial Tasks	113

D5.4: Final report on deep robot action and decision making

5/131

J Co-speech Gestures for Human-Robot Collaboration

127

Executive Summary

This document presents the final update of the work performed between M36 and M45 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 fourth 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.

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

Planning in low and high dimensional state spaces is one of the important directions for developing solutions in a wide range of robotics tasks. Using learning-based methods such as reinforcement learning it is possible to develop end to end planning methods.

1.1.2 Innovations and achieved results

Within this year, AU proposed a novel methodology for obstacle avoidance based on Lyapunov-inspired deep reinforcement learning. Moreover, AU proposed a curiosity-driven reinforcement learning based method for low-level flight control, which is capable of learning to control and navigate a quad-copter UAV towards the desired position where the inputs to the learned model are odometry data, and the output is motor speeds.

1.1.3 Ongoing and future work

We are further extending our work on is the machine imagination, and specifically exploring the effect of combining machine imagination in low and high level UAV control methods.

1.2 Deep Navigation (T5.2)

1.2.1 Objectives

Navigation is required for a vast range of robotic tasks. As pure navigation approaches mature, challenges move more towards unexplored environments and integration into more complex robotic systems such as mobile manipulators.

1.2.2 Innovations and achieved results

Within this year, ALU-FR has proposed a novel interactive multi-object search task that requires robots to manipulate their surroundings to achieve their goal by opening doors that are blocking their path or looking inside drawers and cabinets. Furthermore, ALU-FR has proposed a hierarchical reinforcement learning agent that learns to reason on an object and instance level to combine exploration, navigation and manipulation. Building on previous work from the OpenDR project for low-level subpolicies. This agent achieves impressive results on this novel task. Also, AUTH developed a novel DRL-based end-to-end trainable agent for differential-drive wheeled robot navigation, while also developing the appropriate techniques to improve

learning efficiency. Finally, AUTH also further developed a data efficient DRL approach for robust inertial-based UAV localization.

1.2.3 Ongoing and future work

We are further extending our work on exploration and mobile manipulation in unexplored indoor environments. We plan to incorporate more complex manipulation actions within the hierarchical framework as well as to connect the current reinforcement learning based reasoning with the semantic knowledge that is embedded in large language models.

1.3 Deep Action and Control (T5.3)

1.3.1 Objectives

TUD’s key objective was to address the challenges associated with transferring control policies from simulated environments to real-world robotics, commonly known as the sim2real problem. We aimed to develop a unified framework, EAGERx, which could harmonize the different aspects of both simulated and real robotic learning. The framework sought to overcome issues such as model discrepancies, asynchronous control, and inaccuracies in physical phenomena.

Also TUD worked towards enhancing the sample efficiency and stabilize the training process in off-policy reinforcement learning. To that end, TUD focused on addressing the issue of irrelevant samples in experience replay buffers that hamper an agent’s performance. TUD introduced a novel method, Action Sensitive Experience Replay (ASER), designed to prioritize relevant states in the replay buffer, particularly those where non-optimal actions significantly affect the return.

1.3.2 Innovations and achieved results

TUD introduced EAGERx, a novel framework with a unified software pipeline designed to mitigate the sim2real gap. The framework is compatible with various simulators and incorporates features like integrated delay simulation, domain randomization, and a unique synchronization algorithm. TUD evaluated EAGERx on two benchmark robotic tasks, demonstrating its effectiveness in enabling consistent behavior across both simulated and real-world scenarios. The results were summarized in a paper currently under review.

TUD developed ASER, a novel experience replay method that reallocates modeling resources to prioritize states where the return is especially sensitive to action choice. This innovation led to substantial improvements in sample efficiency, stability, and overall performance. Moreover, TUD demonstrated that the approach enables smaller function approximators, like neural networks with fewer neurons, to perform well in environments where they would usually struggle. The results were summarized in an accepted workshop paper.

1.3.3 Ongoing and future work

As future work, TUD will maintain EAGERx as an open source project, and showcase its usefulness in the agile production use-case.

In addition, TUD aims to extend the evaluation of ASER across a larger array of environments to better understand its implications and limitations. Potential directions include the

implementation of ASER in discrete action-space algorithms like DQN and exploring its applicability in transfer learning, particularly in sim-to-real scenarios.

1.4 Human Robot Interaction (T5.4)

1.4.1 Objectives

The objective of the TUD was to simplify the interaction between humans and robots in industrial settings. We aimed to develop a system, EValueAction (EVA), that reduces the cost and complexity associated with collecting interactive demonstrations for learning from demonstration techniques. The system is designed to pre-train policies using human demonstrations and refine them interactively, providing a safe and efficient framework for human-robot collaboration. The ultimate goal was to achieve a balance between exploration and exploitation in interactive imitation learning (IIL), thereby enabling the robot to generalize well in various scenarios.

Interaction and collaboration between human and robot requires effective modes of communication to assign robot tasks and coordinate activities. As communication can utilize different modalities, a multi-modal approach can be more expressive than single modal models alone. The objective of TAU was to utilize different sensor inputs either individually and combined in human-robot collaborative tasks.

1.4.2 Innovations and achieved results

TUD introduced EVA, a novel system to streamline the process of collecting interactive demonstrations for IIL. EVA leverages simulations to preemptively identify and mitigate failures, thus ensuring a safer environment for human-robot interaction. TUD's approach resulted in an initial policy that could be iteratively refined, making the system more adaptive and robust. A case study validated the effectiveness of EVA by highlighting the crucial role of informative demonstrations for achieving good generalization. The work was summarized in an accepted conference paper.

TAU has demonstrated 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. In addition, TAU has developed a co-speech gesture model that can assign robot tasks for human-robot collaboration, by utilizing speech commands, gestures and object perception.

1.4.3 Ongoing and future work

TUD plans to fully implementing the EVA system and rigorously evaluate its real-world performance. This will lay the groundwork for implementing EVA as a standard solution for enhancing human-robot interaction in Industry 5.0.

TAU plans to extend the human-robot collaboration scenario to long-term tasks and repetitive actions. Human commands and perception coordinates the shared tasks, thereby offering lower workload and better ergonomics for the human operator.

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.

- * AU proposed a novel methodology for obstacle avoidance based on Lyapunov-inspired deep reinforcement learning, presented in Section 2.1.
- * AU proposed a curiosity-driven reinforcement learning based method for learning low-level control of quad-copter UAV exploring a newly introduced new curiosity approach called High Level Curiosity (HCM), presented in Section 2.2. In this approach, the algorithm trains a policy using intrinsic and extrinsic rewards to control the UAV, avoid Obstacles, and control the Yaw Direction toward the desired position.
- * The EAGERx toolkit presented by TUD in Section 4.1 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.
- * The experience replay method presented by TUD in Section 4.2 enables users to efficiently learn control policies with off-policy RL algorithms.
- * The hierarchical interactive multi-object search approach introduced by ALU-FR develops a high-level reinforcement learning agent to coordinate low-level subpolicies across complex long-horizon tasks.

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

- * AUTH developed a DRL-based end-to-end trainable agent for differential-drive wheeled robot navigation, while also developing the appropriate techniques to improve learning efficiency (Section 3.2). AUTH also continued working on the data efficient DRL approach for robust inertial-based UAV localization developed in D5.3, providing additional evaluation experiments (Section 3.3).
- * ALU-FR developed a hierarchical multi-object search approach, described in Section 3.1. This method uses a high-level reinforcement learning agent to coordinate low-level subpolicies, resulting in autonomous exploration and interaction of unexplored environments over very long-horizons.

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 Section 4.1 with the graph structure of environments in the EAGERx toolkit. This functionality allows the user to use the perception algorithms in WP3 as nodes.
- * ALU-FR extended its multi-object search to a hierarchical method, centered around a semantic map that serves as central memory component across high- and low-level policies.

- * TAU has contributed to this objectives with different methods that utilize perception (speech, visual and multi-modal) to coordinate robot actions in industrial human-robot collaborative tasks, presented in Section 5.3

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

- * TUD has contributed to this objective as described in Section 5.1 with a novel system to streamline the process of collecting interactive demonstrations for interactive imitation learning. EVA uniquely leverages simulations to preemptively identify and mitigate failures, thus ensuring a safer environment for human-robot interaction.
- * TAU has contributed to this objectives with different methods that utilize perception for enabling efficient and effective human robot collaboration, presented in Section 5.2. Both individual perception tools and tools that fuse multiple modalities are evaluated for industrial human-robot collaborative tasks.

2 Deep Planning

2.1 Lyapunov-inspired deep reinforcement learning for obstacle avoidance

2.1.1 Introduction and objectives

Despite the remarkable success of learning-based policies in various robot platforms and tasks [21, 20, 23], they have been subject to criticism due to their limited interpretability, particularly in terms of safety and stability. Ensuring stability in robotic systems is of utmost importance to mitigate undesirable behaviors and potential hazards. To tackle these concerns, researchers have explored various approaches at different levels. While advancements in more sophisticated algorithms have demonstrated improved safety experimentally across several domains [12], there has also been a concerted effort to integrate concepts from conventional control theory in an attempt to bridge this interpretability gap [3]. For instance, recent advancements include the synthesis of Lyapunov-stable neural network controllers for non-linear state feedback control [4], offering promising avenues for achieving stability in learning-based control systems. However, these methods are computationally expensive, and challenging to scale in higher state-space dimensions.

We proposed a novel approach for robotic navigation with obstacle presence using a deep reinforcement learning (DRL) strategy inspired by the principles of Lyapunov theory to enhance stability and safety. To achieve this, we formalize the robot planning problem as a state-space control problem, integrating obstacle locations into the state representation. The objective of the control is to achieve constant velocity movement. A Lyapunov function is then designed to provide conditions for the safe travel of the robot. A reward-shaping strategy is introduced, leveraging the Lyapunov function designed for the environment, guiding the learning process. Furthermore, we introduced a constrained exploration strategy for DRL training, incorporating the Lyapunov condition to improve exploration efficiency and accelerate the training. By adopting the proposed strategy, the policy learns to satisfy the Lyapunov conditions, resulting in improved stability, convergence, and overall performance of the robotic navigation system in complex environments.

2.1.2 Description of work performed so far

Details of this work can be found in the pre-print listed below, which is also provided in Appendix A:

- H. I. Ugurlu and E. Kayacan “*Lyapunov-inspired deep reinforcement learning for robot navigation in obstacle environments*” (to be submitted to American Control Conference - 2024)

After its success in games and simulated control tasks, deep reinforcement learning is studied extensively for robotics to learn neural network-based planners or controllers. However, in contrast to conventional control-theoretic methods, neural network controllers lack an understanding of safety or stability due to their black-box nature. We propose a deep reinforcement learning (DRL) strategy inspired by Lyapunov theory for addressing safe robot navigation problems with obstacle presence. The robot planning problem is formulated as a state-space control problem, incorporating obstacle locations as part of the state representation. A reward-shaping strategy is introduced, leveraging a Lyapunov function designed for the environment. Additionally, a constrained exploration method is proposed to guide the DRL training process. Experimental results demonstrate that the proposed method trains faster than a vanilla DRL policy and achieves better exploration, leading to the learning of superior policies in terms of completion rates compromising maintenance of speeds closer to the desired target speed. The findings highlight the potential of incorporating Lyapunov theory into DRL approaches for improving robot navigation.

2.1.3 Future work

In the future, there are several directions for further exploration and improvement based on the findings of this research. Firstly, it would be valuable to extend the proposed DRL training strategy to more complex and dynamic environments, where the robot needs to adapt to changing obstacles or varying task requirements. Additionally, investigating the scalability of the approach to large-scale robotic systems would be crucial. Lastly, it would be interesting to explore the transferability and generalization capabilities of the trained policies to real-world environments or tasks. Addressing these aspects would contribute to a deeper understanding and practical applicability of the proposed DRL approach for robot navigation problems.

2.2 Curiosity-Driven Reinforcement Learning based Low-Level Flight Control

2.2.1 Introduction and objectives

Curiosity is one of the main motives in many of the natural creatures with measurable levels of intelligence for exploration and, as a result, more efficient learning. It makes it possible for humans and many animals to explore efficiently by searching for being in states that make them surprised with the goal of learning more about what they do not know. As a result, while being curious, they learn better. In the machine learning literature, curiosity is mostly combined with reinforcement learning-based algorithms as an intrinsic reward.

We proposed an algorithm based on the drive of curiosity for autonomous learning to control by generating proper motor speeds from odometry data. The quadcopter controlled by our

proposed algorithm can pass through obstacles while controlling the Yaw direction of the quadcopter toward the desired location. To achieve that, we also proposed a new curiosity approach based on prediction error. We ran tests using on-policy, off-policy, on-policy plus curiosity, and the proposed algorithm and visualized the effect of curiosity in evolving exploration patterns. Results show the capability of the proposed algorithm to learn optimal policy and maximize reward where other algorithms fail to do so.

2.2.2 Description of work performed so far

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

- Ramezani Dooraki, A., Iosifidis, A., 2023. Curiosity-Driven Reinforcement Learning based Low-Level Flight Control. <https://arxiv.org/abs/2307.15724>.

This work proposes using a policy gradient-based reinforcement learning algorithm to learn an optimal policy. However, the off-the-shelf policy gradient algorithms such as Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) cannot learn an optimal policy for our problem. To solve the problem, we change the PPO algorithm by adding 1) A new state value function network and 2) Introducing a new curiosity approach for motivating meaningful exploration. To implement and test the capability of our algorithm in a real work problem. We design and implement a new simulation-based environment in Gazebo simulator. Moreover, we implement our algorithm in Python, and for the connection between our algorithm and the environment, we use Robot Operating System (ROS).

2.2.3 Future work

In the future work, we plan to increase the speed and efficiency of learning-based flight controllers based on machine imagination (MI). We expect MI to allow the learning-based controller calculate the effect of a trajectory of interaction with the environment using the model learned from the environment (without executing the trajectory in the actual environment). There are different methodologies for implementing machine imagination, such as RNN-based and Transformer-based. We plan to implement both of the mentioned methods and measure their similarities and differences.

3 Deep Navigation

3.1 Learning Hierarchical Interactive Multi-Object Search for Mobile Manipulation

3.1.1 Introduction and objectives

We propose an interactive multi-object search task in which the agent as to manipulate the environment to find the objects of interest as doors may block its path and objects may be hidden within articulated objects such as cabinets and drawers. We then introduce a hierarchical reinforcement learning approach that learns to compose exploration, navigation, and manipulation skills. To achieve this, we design an abstract high-level action space around a semantic map memory and leverage the explored environment as instance navigation points. This is enabled

by our previous work on mobile manipulation [7, 8] and exploration [18] within the OpenDR project, which now serve as powerful low-level subpolicies for the agent.

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 E.

- [19] F. Schmalstieg, D. Honerkamp, T. Welschehold and A. Valada, “*Learning Hierarchical Interactive Multi-Object Search for Mobile Manipulation*”, Jul. 2023, DOI: 10.48550/ARXIV.2307.06125.

In this work, we propose Hierarchical Interactive Multi-Object Search (HIMOS), a hierarchical reinforcement learning approach to learn both exploration and manipulation skills and to reason at a high level about the required steps. We combine learned motions for local exploration in continuous action spaces and frontier exploration for long-horizon exploration together with mobile manipulation skills for object interactions. We use semantic maps as the central memory component, which have shown to be an expressive and sample-efficient representation for these tasks and design a high-level action space that exploits the acquired knowledge about the environment. By leveraging explored object instance locations as navigation waypoints, our approach efficiently learns these complex tasks from little data and consistently achieves success rates above 90% even as the number of target objects increases. By equipping all the low-level skills with mobility, we remove the “hand-off” problem in which subpolicies have to terminate in the initial set of the following skill. Lastly, we transfer the trained agent to the real world and demonstrate that it successfully accomplishes these tasks in a real office environment. In particular, we replace the subpolicies from simulation with unseen real-world variations and find that the policy is able to generalize to these unseen subpolicies and is robust to failures in their execution, making it highly modular and flexible for transfer. Finally, we present ablation studies to evaluate the impact of the main design decisions.

3.1.3 Future work

In the future, we plan to investigate the benefits of jointly training the high- and low-behavior and integrate more sophisticated mapping modules to build a semantic map directly from the robot sensors. Further, additional low-level behaviors could extend environment interaction options or perform more goal-oriented active perception actions.

3.2 Deep Reinforcement Learning with Action Masking for Differential-drive Robot Navigation using Low-Cost Sensors

3.2.1 Introduction and objectives

Autonomous navigation of mobile robots has been a popular research topic in the field of robotics for decades. The ability to navigate in complex and dynamic environments is essential for many applications, such as search and rescue, logistics, and home care. One of the most common and versatile types of mobile robots is the differential-drive wheeled robot. These robots are easy to build and control, and their differential-drive system allows them to turn on the spot and move in any direction. At the same time, Deep Reinforcement Learning (DRL) [2]

has also emerged as a promising technique for training autonomous agents to perform complex tasks, leading to several robotics applications [16], including navigation, manipulation, and control.

However, recent research mainly targets robots that use multiple sensors and relatively expensive configurations with LiDAR, depth cameras and others. One such example is in [5], where the authors used a Jetson Nano for obstacle avoidance using a monocular camera. In another work [1], the authors used the Turtlebot 3 Waffle Pi, and trained a Double DQN [22] agent to navigate to a target. In a more recent work using the same robot [24], a mapless local path planning approach was presented, that used variants of Deep Q-Network [14] to increase success rates, proposing the n-Step Dueling Double DQN with Reward-Based ϵ -Greedy (RND3QN) algorithm. In [17], the authors successfully used Deep Deterministic Policy Gradient [11] to train an agent to drive a differential-drive robot to a target, improving on this paper’s authors’ earlier work in [10]. They used curriculum learning [15] to gradually train the agent on a small set of increasingly difficult maps. Another more generic example is the established Robot Operating System (ROS)¹ navigation stack, which while it can work with inexpensive ultrasonic distance sensors, it is more well-suited to work with LiDAR and depth sensors. As a result, many DRL stacks are designed exclusively for high-end robotic hardware, which limits their potential impact in a wide range of applications, from education to low-cost mass production robots. At the same time, training DRL agents with low-fidelity and noisy sensors can worsen sample efficiency. This necessitates the use of more sample-efficient paradigms in such applications.

3.2.2 Description of work performed so far

In this work, we propose a method for training an agent to drive a low-cost differential-drive wheeled robot navigating to a target while avoiding obstacles, using the well-established Proximal Policy Optimization (PPO) [6] RL algorithm. To improve training efficiency we employ invalid action masking [9], also known as Maskable PPO, after appropriately designing two masks that can lead to increased performance. This work a) introduces a more systematic and effective approach to perform action masking, as well as b) paves the way for introducing a state-of-the-art DRL approach using low-cost sensors in the robotic navigation domain. The agent used, apart from its sensor values, takes only the relative angle and distance to its target and not its own or the target’s absolute position, and thus can act as a local path planner and navigate dynamically in unknown environments. We developed a randomized procedural map generation method within the Webots robotics simulator [13], to be able to train and realistically evaluate the agent in complex environments with obstacles of various challenging shapes, using realistic noisy sensors. We experimentally demonstrate that the proposed agent can robustly navigate to a given target even in unknown procedurally generated environments, or even when denying part of its sensor input. Finally, we demonstrated a practical use-case using object detection to dynamically search for, and move to objects within unknown environments.

The proposed method, along with a detailed evaluation in several different scenarios, are provided in Appendix C:

C K. Tsampazis, M. Kirtas, P. Tosidis, N. Passalis, and A. Tefas “*Deep Reinforcement Learning with Action Masking for Differential-Drive Robot Navigation using Low-cost*

¹<https://www.ros.org/>

Sensors”, IEEE International Workshop on Machine Learning for Signal Processing (accepted), 2023.

3.2.3 Future Work

AUTH will continue working on integrating and evaluating OpenDR tools according to the toolkit’s specifications and plan.

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

3.3.1 Introduction and work performed so far

AUTH also continued working on evaluating and further improving the proposed inertial-based UAV localization approach using data-efficient DRL, as initially introduced in D5.3. To this end, among others, additional experiments have been performed to evaluate the impact of the individual components involved in the proposed method, as well as to quantitatively evaluate the effect of the proposed method on different trajectories. AUTH will continue on working on toolkit integration according to the toolkit’s specifications. The updated technical report is provided in Appendix D:

D D. Tsiakmakis, N. Passalis, and A. Tefas. “*Improving Inertial-based UAV Localization using Data-efficient Deep Reinforcement Learning*”, Technical Report (AUTH), 2023.

3.3.2 Future Work

AUTH will continue working on integrating and evaluating OpenDR tools according to the toolkit’s specifications and plan.

4 Deep action and control

4.1 EAGERx: Graph-Based Framework for Sim2real Robot Learning

4.1.1 Introduction and objectives

The main focus of this work revolves around the challenge of transferring control policies from simulation to real-world robotic systems, a problem known as sim2real. Simulations provide a risk-free and cost-effective platform for the development and testing of robotic algorithms. However, the transfer of these algorithms faces hurdles due to the sim2real gap, caused by various factors like inaccurate modeling of physical phenomena, separate software implementations, and asynchronous control in real-world robotics. Addressing these challenges requires the integration of various kinds of abstractions and simulator environments to make the simulation more consistent with the real world. Existing solutions often restrict users to specific simulation environments and fail to synchronize parallel components effectively. Our objective is to present EAGERx, a novel framework designed to bridge these gaps efficiently and flexibly.

4.1.2 Description of work performed so far

Sim2real, that is, the transfer of learned control policies from simulation to real world, is an area of growing interest in robotics due to its potential to efficiently handle complex tasks. The sim2real approach, however, is hampered by discrepancies between simulation and reality, inaccuracies in physical phenomena modeling, and asynchronous control, among others. To this end, we introduce EAGERx, a framework with a unified software pipeline for both real and simulated robot learning. It can support various simulators and aids in integrating state, action and time-scale abstractions to facilitate learning. EAGERx’s integrated delay simulation, domain randomization features, and proposed synchronization algorithm contribute to narrowing the sim2real gap. We demonstrate the efficacy of EAGERx in accommodating diverse robotic systems and maintaining consistent simulation behavior. EAGERx is open source and its code is available at <https://eagerx.readthedocs.io>.

The proposed method, along with a detailed evaluation are provided in Appendix F:

F B. van der Heijden, J. Luijkx, L. Ferranti, J. Kober and R. Babuska “*EAGERx: Graph-Based Framework for Sim2real Robot Learning*”, IEEE Robotics and Automation Letters (under review), 2023.

4.1.3 Future work

As future work, TUD will maintain EAGERx as an open source project, and showcase its usefulness in the agile production use-case.

4.2 Prioritizing States with Action Sensitive Return in Experience Replay

4.2.1 Introduction and objectives

This work is focused on addressing the efficiency of experience replay in reinforcement learning, particularly for algorithms that employ state-action value functions. In the conventional use of experience replay, all state transitions are treated equally, which does not necessarily lead to an optimal policy. Previous works have demonstrated that every sample is not equally relevant for learning a good policy, and this inefficiency is amplified when global function approximators, like neural networks, are used. We introduce Action Sensitive Experience Replay (ASER), a method that seeks to prioritize “decision points” in the replay buffer to enhance learning efficiency. Our key objective is to fine-tune the replay distribution by defining a modeling importance criterion that assesses the sensitivity on return of taking a suboptimal action. The ultimate aim is to optimize the allocation of computational resources and improve both sample efficiency and final policy performance.

4.2.2 Description of work performed so far

Experience replay for off-policy reinforcement learning has been shown to improve sample efficiency and stabilize training. However, typical uniformly sampled replay includes many irrelevant samples for the agent to reach good performance. We introduce Action Sensitive Experience Replay (ASER), a method to prioritize samples in the replay buffer and selectively model parts of the state-space more accurately where choosing sub-optimal actions has a larger effect on the return. We experimentally show that this can make training more sample efficient

and that this allows smaller parametric function approximators – like neural networks with few neurons – to achieve good performance in environments where they would otherwise struggle.

The proposed method, along with a detailed evaluation are provided in Appendix G:

G A. Keijzer, B. van der Heijden, and J. Kober “*Prioritizing States with Action Sensitive Return in Experience Replay*”, Sixteenth European Workshop on Reinforcement Learning (accepted), 2023.

4.2.3 Future work

We aim to extend the evaluation of ASER across a larger array of environments to better understand its implications and limitations. Potential directions include the implementation of ASER in discrete action-space algorithms like DQN and exploring its applicability in transfer learning, particularly in sim-to-real scenarios.

5 Human robot interaction

5.1 EValueAction: a proposal for policy evaluation in simulation to support interactive imitation learning

5.1.1 Introduction and objectives

The development of Industry 4.0 has paved the way for the implementation of Artificial Intelligence (AI) in various industrial settings. In particular, Learning from Demonstration (LfD) emerges as a technique to facilitate human-robot collaboration. However, existing LfD methods encounter issues of dataset bias and overfitting, which limit their generalization capabilities. This project aims to address these challenges by introducing the EValueAction (EVA) framework. The primary objectives are twofold: 1) to reduce the number of demonstrations needed for effective learning and 2) to improve the quality of these demonstrations, thereby minimizing both mental and physical effort on the human side. This work aligns with the larger research agenda towards Industry 5.0, emphasizing sustainable, value-driven, and human-centric production processes.

5.1.2 Description of work performed so far

The up-and-coming concept of Industry 5.0 fore-sees human-centric flexible production lines, where collaborative robots support human workforce. In order to allow a seamless collaboration between intelligent robots and human workers, designing solutions for non-expert users is crucial. Learning from demonstration emerged as the enabling approach to address such a problem. However, more focus should be put on finding safe solutions which optimize the cost associated with the demonstrations collection process. This paper introduces a preliminary outline of a system, namely EValueAction (EVA), designed to assist the human in the process of collecting interactive demonstrations taking advantage of simulation to safely avoid failures. A policy is pre-trained with human-demonstrations and, where needed, new informative data are interactively gathered and aggregated to iteratively improve the initial policy. A trial case study further reinforces the relevance of the work by demonstrating the crucial role of informative demonstrations for generalization.

The proposed method, along with a detailed evaluation are provided in Appendix H:

H F. Sibona, J. Luijkx, B. van der Heijden, L. Ferranti, and M. Indri “*EValueAction: a proposal for policy evaluation in simulation to support interactive imitation learning*”, IEEE 21st International Conference on Industrial Informatics (INDIN) (accepted), 2023.

5.1.3 Future work

As future work, we intend to implement the EVA framework on a real system and benchmark its usefulness in making real-time risk assessments of task failure.

5.2 Sensor-based Human-Robot Collaboration for Industrial Tasks

5.2.1 Introduction and objectives

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 therefore 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. 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.

5.2.2 Description of work performed so far

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

I A. Angleraud, A. Ekrekli, K. Samarawickrama, G. Sharma, R. Pieters “*Sensor-based Human-Robot Collaboration for Industrial Tasks*”, Robotics and Computer-Integrated Manufacturing (accepted), 2023.

In this work the current limitations in perception models and situational awareness for industrial human-robot collaboration is addressed. 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 provided a general HRC software framework, based on ROS, which can be utilized to replicate our developments. The framework is built 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

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 than 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.

5.3 Co-speech Gestures for Human-Robot Collaboration

5.3.1 Introduction and objectives

Fluent interaction between human and robot requires reliable perception to capture the commands of a person. While recent approaches in deep learning have established impressive tools to detect e.g., human pose, gestures and speech, single tools alone can not always convey easily the commands intended. Reasons for this are the limited expressions available for different modes of communication and the limitations in perception performance. Human hand gestures, for example, contain much less information content than speech. On the other hand, gesture detection can be done much quicker than speech recognition, leading to a faster response time. These conflicting properties motivate to combine multiple perception tools into a single multi-modal detection model that utilizes communication from human to robot for assigning tasks and coordinating the collaboration.

5.3.2 Description of work performed so far

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

J. A. Ekrekli, A. Angleraud, G. Sharma, R. Pieters “*Co-speech Gestures for Human-Robot Collaboration*”, under review, 2023.

In this work we compare different perception tools and analyse them with respect to their suitability for human-robot collaboration. A co-speech gesture model is then developed that combines speech, human hand gestures and object detection to achieve effective communication of desired robot tasks, such as picking human-specified objects and robot to human hand-overs. The developments are intended for industrial human-robot collaboration where a collaborative robot shares its tasks, and works in close collaboration with, a human operator.

5.3.3 Future work

In future work we will aim to extend the sensor-fusion approach to multiple perception modalities, with higher-level speech commands and visual-language models. This should lead to more intuitive coordination of the human by more suitable and effective commands.

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 a novel DRL training strategy for addressing robot navigation problems by leveraging the principles of the Lyapunov theory. The robot planning problem in an obstacle environment was formulated as a state-space control problem, where obstacle locations are treated as part of the state representation. To guide the learning process, a reward-shaping strategy was introduced which is based on a Lyapunov function designed specifically for the formulated environment. Additionally, a constrained exploration scenario for DRL training that incorporates the Lyapunov condition was proposed. Experimental results demonstrate that the proposed method achieves improved exploration, leading to the learning of a superior policy. AU also proposed a novel approach for learning the low-level flight control of UAV robots. Using the proposed low-level flight controller, a quadcopter can learn to fly and avoid obstacles while controlling its yaw direction toward the desired position. Reinforcement learning is used in combination with curiosity. Experiments show that using existing curiosity methods could not provide the desired results. Thus, a new curiosity approach called the High Curiosity Module (HCM) was proposed. Using the new curiosity module and combining it with PPO as the reinforcement learning method, the new approach could learn optimal policies where other existing methods failed to do so.

Chapter 3 detailed the status of the work performed for Task 5.2–Deep Navigation. ALU-FR introduced an approach that learns to coordinate and combine previous OpenDR works on exploration and mobile manipulation to autonomously solve a novel interactive multi-object-search task. This method enables autonomous navigation and search in unexplored environments over very long horizons. The method is evaluated in both simulation and the real world and currently under peer-review. Furthermore, AUTH worked towards O2c by developing a DRL-based end-to-end trainable agent for differential-drive wheeled robot navigation, while also developing the appropriate techniques to improve learning efficiency. Furthermore, AUTH also continued working on O2c on the data efficient DRL approach for robust inertial-based UAV localization developed in D5.3, concluding this work.

Chapter 4 highlighted the work performed for Task 5.3–Deep Action and Control. TUD focused on two main aspects: mitigating the sim2real gap and enhancing sample efficiency in off-policy reinforcement learning. EAGERx, a novel framework, was developed to address the sim2real problem by unifying software pipelines for simulated and real robotic learning. In parallel, ASER was introduced to prioritize relevant states in the experience replay buffer, thus improving sample efficiency, stability, and overall performance. Both methods were validated through benchmark tasks and summarized in academic papers.

Finally, Chapter 5 highlighted the work performed for Task 5.4–Human Robot Interaction. TUD’s primary objective was to simplify and make safer the collaboration between humans and robots in an industrial context. TUD developed EValueAction (EVA), a novel system for streamlining the collection of interactive demonstrations for interactive imitation learning (IIL). EVA incorporates simulation to safely avoid failures, enabling a more adaptive and robust policy that can be iteratively refined. The work has shown promising results in a case study and was accepted for publication in a conference paper. TAU’s main objective was to utilize perception for human-robot collaboration, either by individual perception tools or by combining input from

multiple tools into a fused output. Human speech, gestures and object perception provided the input for commanding robot actions and enabling collaboration in shared industrial tasks.

References

- [1] H. Aydemir, M. Gök, and M. Tekerek. Reinforcement learning based local path planning for mobile robot. In *Interdisciplinary Conf. Mechanics, Computers and Electrics*, 11 2021.
- [2] C. Berner et al. Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680, 2019.
- [3] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. *Advances in neural information processing systems*, 31, 2018.
- [4] H. Dai, B. Landry, L. Yang, M. Pavone, and R. Tedrake. Lyapunov-stable neural-network control. In *Robotics: Science and Systems*, 2021.
- [5] T.-V. Dang and N.-T. Bui. Obstacle avoidance strategy for mobile robot based on monocular camera. *Electronics*, 12(8), 2023.
- [6] J. S. et al. Proximal policy optimization algorithms. *arXiv 1707.06347*, 2017.
- [7] 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.
- [8] D. Honerkamp, T. Welschehold, and A. Valada. N^2m^2 : Learning navigation for arbitrary mobile manipulation motions in unseen and dynamic environments. *IROS 2022 Workshop on Mobile Manipulation and Embodied Intelligence*, 2022.
- [9] S. Huang and S. Ontañón. A closer look at invalid action masking in policy gradient algorithms. *The Intl. FLAIRS Conf. Proceedings*, 35, may 2022.
- [10] M. Kirtas et al. Deepbots: A webots-based deep reinforcement learning framework for robotics. In *Artificial Intelligence Applications and Innovations*, pages 64–75, Cham, 2020. Springer Intl. Publishing.
- [11] T. P. Lillicrap et al. Continuous control with deep reinforcement learning. *arXiv 1509.02971*, 2015.
- [12] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza. Learning high-speed flight in the wild. *Science Robotics*, 6(59):eabg5810, 2021.
- [13] O. Michel. WebotsTM: Professional Mobile Robot Simulation. *Int. Journal of Advanced Robotic Systems*, 1, 03 2004.
- [14] V. Mnih et al. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.
- [15] S. Narvekar et al. Curriculum learning for reinforcement learning domains: A framework and survey. *arXiv 2003.04960*, 2020.

- [16] N. Passalis et al. OpenDR: An open toolkit for enabling high performance, low footprint deep learning for robotics. In *Proc. IEEE/RSJ Intl. Conf. Intelligent Robots and Systems*, 2022.
- [17] A. S. Sadr et al. An efficient planning method for autonomous navigation of a wheeled-robot based on deep reinforcement learning. In *12th Intl. Conf. Computer and Knowledge Engineering*, pages 136–141, 2022.
- [18] F. Schmalstieg, D. Honerkamp, T. Welschhold, 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.
- [19] F. Schmalstieg, D. Honerkamp, T. Welschhold, and A. Valada. Learning hierarchical interactive multi-object search for mobile manipulation. *arXiv preprint arXiv:2111.12673*, 2023.
- [20] Y. Song, K. Shi, R. Penicka, and D. Scaramuzza. Learning perception-aware agile flight in cluttered environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1989–1995, 2023.
- [21] H. I. Ugurlu, X. H. Pham, and E. Kayacan. Sim-to-real deep reinforcement learning for safe end-to-end planning of aerial robots. *Robotics*, 11(5):109, 2022.
- [22] H. van Hasselt et al. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015.
- [23] K. Xu, Z. Hu, R. Doshi, A. Rovinsky, V. Kumar, A. Gupta, and S. Levine. Dexterous manipulation from images: Autonomous real-world rl via substep guidance. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5938–5945, 2023.
- [24] Y. Yin, Z. Chen, G. Liu, and J. Guo. A mapless local path planning approach using deep reinforcement learning framework. *Sensors*, 23(4), 2023.

A Lyapunov-inspired deep reinforcement learning for robot navigation in obstacle environments

Lyapunov-inspired deep reinforcement learning for robot navigation in obstacle environments

Halil Ibrahim Ugurlu and Erdal Kayacan

Abstract—After its success in games and simulated control tasks, deep reinforcement learning is studied extensively for robotics to learn neural network-based planners or controllers. However, in contrast to conventional control-theoretic methods, neural network controllers lack an understanding of safety or stability due to their black-box nature. This research paper proposes a deep reinforcement learning (DRL) strategy inspired by Lyapunov theory for addressing safe robot navigation problems with obstacle presence. The robot planning problem is formulated as a state-space control problem, incorporating obstacle locations as part of the state representation. A reward-shaping strategy is introduced, leveraging a Lyapunov function designed for the environment. Additionally, a constrained exploration method is proposed to guide the DRL training process. Experimental results demonstrate that the proposed method trains faster than a vanilla DRL policy and achieves better exploration, leading to the learning of superior policies in terms of completion rates compromising maintenance of speeds closer to the desired target speed. The findings highlight the potential of incorporating Lyapunov theory into DRL approaches for improving robot navigation.

I. INTRODUCTION

Despite the remarkable success of learning-based policies in various robot platforms and tasks [1, 2, 3], they have been subject to criticism due to their limited interpretability, particularly in terms of safety and stability. Ensuring stability in robotic systems is of utmost importance to mitigate undesirable behaviors and potential hazards. To tackle these concerns, researchers have explored various approaches at different levels. While advancements in more sophisticated algorithms have demonstrated improved safety experimentally across several domains [4], there has also been a concerted effort to integrate concepts from conventional control theory in an attempt to bridge this interpretability gap [5]. For instance, recent advancements include the synthesis of Lyapunov-stable neural network controllers for non-linear state feedback control [6], offering promising avenues for achieving stability in learning-based control systems. However, these methods are computationally expensive, and challenging to scale in higher state-space dimensions.

This research paper presents a novel approach for robotic navigation with obstacle presence using a deep reinforcement learning (DRL) strategy inspired by the principles of Lyapunov theory to enhance stability and safety. To achieve this, we formalize the robot planning problem as a state-space

H. I. Ugurlu is with the Artificial Intelligence in Robotics Laboratory (Air Lab), Department of Electrical and Computer Engineering, Aarhus University, 8000 Aarhus C, Denmark halil at ece.au.dk E. Kayacan is with the Automatic Control Group, Department of Electrical Engineering and Information Technology, Paderborn University, Paderborn, Germany (email: erdal.kayacan at uni-paderborn.de)

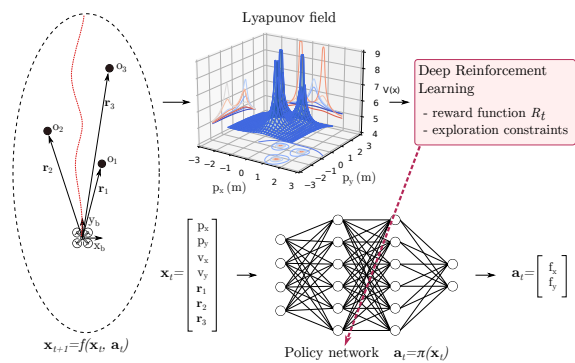


Fig. 1. The policy network is trained with deep reinforcement learning for obstacle avoidance leveraging Lyapunov theory.

control problem, integrating obstacle locations into the state representation. The proposed method is illustrated in Fig. 1. The objective of the control is to achieve constant velocity movement. A Lyapunov function is then designed to provide conditions for the safe travel of the robot. A reward-shaping strategy is introduced, leveraging the Lyapunov function designed for the environment, guiding the learning process. Furthermore, we introduce a constrained exploration strategy for DRL training, incorporating the Lyapunov condition to improve exploration efficiency and accelerate the training. By adopting the proposed strategy, the policy learns to satisfy the Lyapunov conditions, resulting in improved stability, convergence, and overall performance of the robotic navigation system in complex environments.

The proposed DRL strategy holds the promise of addressing the shortcomings of traditional learning-based policies, and its application in real-world robotic systems could significantly advance the capabilities and safety of autonomous navigation. By incorporating the principles of Lyapunov theory into the DRL framework, this research aims to contribute to a deeper understanding and practical applicability of DRL in robotics.

A. Contributions

The contributions of this work are as follows:

- A Lyapunov-inspired reward function, along with the Lyapunov function, for synthesizing a state-based robot navigation policy with DRL.
- A constrained exploration strategy to accelerate the training performance.

- An open-source Webots-based simulation environment for force-controlled obstacle avoidance task for training and evaluation of the proposed method ¹

The structure of the paper is as follows: Section II provides a comprehensive overview of related work in the fields of DRL and Lyapunov theory applications in learning-based robotics. Section III provides the background on DRL and Lyapunov stability. Section IV details the methodology and formulation of the proposed DRL strategy inspired by the Lyapunov theory. Section V presents the experimental setup, including the simulation environment and training procedures, and presents the results with a discussion of the performance of the proposed approach compared to alternative scenarios. Finally, Section VI concludes the paper and highlights potential future research directions.

II. RELATED WORK

Reinforcement learning (RL) has emerged as a prominent field within artificial intelligence, providing a powerful framework for autonomous agents to learn and adapt through interactions with their environments [7]. This paradigm encompasses a set of algorithms and techniques that enable agents to make sequential decisions, aiming to maximize cumulative rewards over time. RL has gained significant attention and achieved remarkable success in various domains, including robotics [8] and gaming [9]. Its ability to handle complex, dynamic environments and learn optimal strategies without explicit supervision has made it a compelling approach for addressing real-world problems. The field of DRL emerged to solve RL problems utilizing deep neural networks. By using neural networks, DRL algorithms can automatically learn complex and hierarchical representations, enabling them to effectively process raw sensory input. Several DRL algorithms have been proposed in the literature [9, 10, 11]. In this work, the Proximal policy optimization (PPO) [12] algorithm is utilized due to its success in state-based tasks.

RL agents must balance between exploring new actions and exploiting their existing knowledge to maximize rewards. Striking the right balance is crucial for efficient learning and achieving optimal policies. Inadequate exploration may lead to suboptimal solutions, while excessive exploration can hinder learning progress. Extensive research has focused on addressing exploration, resulting in the development of algorithms and techniques to guide agents in making informed decisions during the learning process such as ϵ -greedy exploration [7] in discrete state spaces and action noise or state-dependent noise [13] in continuous state spaces. On the other hand, the design of the environment also implicitly shapes the exploration space.

Safe RL for robotics is a burgeoning field that addresses the challenges of deploying learning-based algorithms in robotic systems with guaranteed safety and stability. The reader may refer to a recent review by Brunke et al. [14] for

¹The codes, trained models, and simulation environment can be found at github.com/open-airlab/lyapunov-rl

a detailed survey of safe learning in robotics. A particular line of literature aims to utilize the tools from conventional control theory, such as Lyapunov theory. Lyapunov theory is a fundamental concept in the field of stability analysis and control theory [15]. With the evolution of neural networks in the loop, the theory is addressed to provide similar explanations. A neural network is proposed to learn a Lyapunov function that adapts the largest safe region [16]. The idea is later improved to train and verify the Lyapunov network by counter-examples for piece-wise linear systems [17], which is followed by learning the whole control pipeline (system model, controller, and Lyapunov function) by feedforward neural networks [6]. The Lyapunov theory is also addressed another challenge of learned policies: to keep the system inside an in-distribution state and actions [18]. Although these methods have reached strong conclusions, they cannot easily be extended to higher dimensional tasks due to their computational complexity.

III. BACKGROUND

A. Deep reinforcement learning

The RL problem is typically formulated as a Markov Decision Process, $\mathcal{M} = (S, A, T, R, \gamma)$, consisting the set of states, S , the set of actions, A , the state transition function, $T(s|a)$, the reward function, $R(s, a)$, and the discount factor, γ . For a state, $\mathbf{x}_t \in S$, at the timestep, t , an action, $\mathbf{a}_t \in A$, is taken, resulting in a state transition according to $\mathbf{x}_{t+1} \sim T(\mathbf{x}_t, \mathbf{a}_t)$. Note that, the states can also be represented with, s , in the RL community; however, the common terminology in control engineering is preferred in this paper. Similarly, the state transition function is replaced with, $f(\mathbf{x}_t, \mathbf{a}_t)$, for convenience in the remainder of this paper. Each transition yields a reward, $r_t = R(\mathbf{x}_t, \mathbf{a}_t)$, signaling the quality of the taken action. The cumulative reward collected from a sequence of states and actions starting at timestep, t , is defined as,

$$G_t = \sum_{k=0}^{k=\infty} \gamma^k r_{t+k}, \quad (1)$$

where γ is the discount factor determining the weight of future rewards. The objective of the RL problem is to maximize this discounted cumulative reward by choosing a suitable policy, $\pi(\mathbf{x}_t)$.

B. Lyapunov stability

Let a discrete-time, deterministic dynamical control system is modeled as,

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{a}_t), \quad (2)$$

where $\mathbf{x} \in S \subset \mathbb{R}^n$ and $\mathbf{a} \in A \subset \mathbb{R}^m$ represent the system state and applied control input vectors, respectively, and subscripts, t and $t + 1$, represent the consecutive timesteps.

Definition 1 (Equilibrium state): A state, \mathbf{x}_g , is an equilibrium state if there exists and action, $\mathbf{a}_g \in A$, such that $\mathbf{x}_g = f(\mathbf{x}_g, \mathbf{a}_g)$.

Consider a state-feedback control policy, $\mathbf{a} = \pi(\mathbf{x})$, for the modeled system satisfying $\mathbf{a}_g = \pi(\mathbf{x}_g)$. The system model can be reformulated as a function of state,

$$\mathbf{x}_{t+1} = f_\pi(\mathbf{x}_t) = f(\mathbf{x}_t, \pi(\mathbf{x}_t)), \quad (3)$$

with the equilibrium state, $\mathbf{x}_g = f_\pi(\mathbf{x}_g)$.

Definition 2 (Lyapunov stability): Let a Lyapunov function, $V(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$ maps the system states to a scalar. The system, $f_\pi(\mathbf{x})$, is locally stable in a region, $\mathcal{X} \subset \mathbb{R}^n$, around the equilibrium state, \mathbf{x}_g , if there exists a Lyapunov function satisfying the following conditions:

$$V(\mathbf{x}_t) > 0, \quad \forall \mathbf{x}_t \in \mathcal{X}, \mathbf{x}_t \neq \mathbf{x}_g, \quad (4)$$

$$V(\mathbf{x}_{t+1}) \leq (1 - \psi)V(\mathbf{x}_t), \quad \forall \mathbf{x}_t \in \mathcal{X}, \mathbf{x}_t \neq \mathbf{x}_g, \quad (5)$$

$$V(\mathbf{x}_g) = 0, \quad (6)$$

where $\psi > 0$ is a positive scalar.

IV. LYAPUNOV-INSPIRED DRL FOR OBSTACLE AVOIDANCE

A. Problem formulation

The obstacle-avoidance problem is formalized as state-space control. A simplified quadrotor aerial robot model, force-controlled particle mass, is considered as the robot in constant altitude flight. The robot's state, $\mathbf{x}_{robot} \in \mathbb{R}^4$, is defined as,

$$\mathbf{x}_{robot} = [\mathbf{p}^T | \mathbf{v}^T]^T = [p_x, p_y, v_x, v_y]^T, \quad (7)$$

where \mathbf{p} and \mathbf{v} represent position and velocity vectors in the world frame, respectively, each consisting of two elements where subscripts denoting the axis. Note that, since the transformation between the body and world frame does not constitute rotation, the velocity is the same in both frames. Let N_o be the number of obstacles in the proximity of the robot. Each obstacle's position in the robot body frame is represented as,

$$\mathbf{r}_i = [r_{x,i}, r_{y,i}]^T, \quad (8)$$

where $i \in \{1, \dots, N_o\} \subset \mathbb{Z}^+$ is the obstacle number and x and y represents position in body coordinates. Then, the augmented state, $\mathbf{x} \in \mathbb{R}^{4+2N_o}$, is constructed as,

$$\mathbf{x} = [\mathbf{x}_{robot}^T | \mathbf{r}_1^T | \mathbf{r}_2^T | \dots | \mathbf{r}_{N_o}^T]^T. \quad (9)$$

The control command to the robot is defined as,

$$\mathbf{a} = [F_x, F_y]^T, \quad (10)$$

where $F_x, F_y \in [-1, 1]$ are the force applied to the robot in the corresponding axis. The terms explained in (7) - (10) are illustrated in Fig. 2

The aim is to travel in a target direction with a predefined speed while avoiding obstacles. The target direction is selected as the x-axis direction without loss of generality since the world and body frames can be rotated according to the desired direction in the horizontal plane. Then, the goal is defined as a subset of states,

$$\mathcal{X}_g = \{\mathbf{x} | p_y = 0, v_x = v_{des}, v_y = 0, r_{x,i} < 0\}, \quad (11)$$

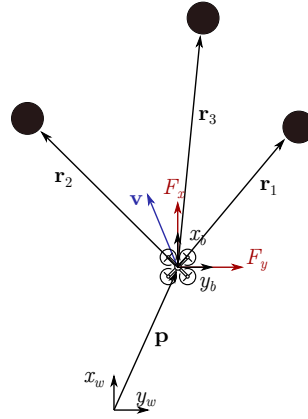


Fig. 2. Diagram showing body and world frames, acting forces on the robot, obstacle position vectors in body frame, and robot's position and velocity vectors.

where $v_{des} = 0.5\text{m/s}$ is the desired velocity of the robot.

Remark 1: Starting from any state in the goal set, $\mathbf{x} \in \mathcal{X}_g$, the constant action, $\mathbf{a}_g = [F_g, 0]$, keeps the system in the goal subset where F_g is a non-zero force to cancel friction force due to the constant velocity and maintain zero acceleration. In this case, p_y and v_y terms stay zero since $F_y = 0$, v_x is kept constant due to zero acceleration, and the condition on the obstacle position continues to hold due to $v_x > 0$, i.e., $r_{x,i}$ decreases.

Remark 2: Since the equilibrium condition for the defined obstacle-avoidance problem is a set instead of a single point, Lyapunov conditions in (4)-(6) are rewritten as,

$$V(\mathbf{x}_t) > 0, \quad \forall \mathbf{x}_t \in \mathcal{X}, \mathbf{x}_t \notin \mathcal{X}_g, \quad (12)$$

$$V(\mathbf{x}_{t+1}) \leq (1 - \psi)V(\mathbf{x}_t), \quad \forall \mathbf{x}_t \in \mathcal{X}, \mathbf{x}_t \notin \mathcal{X}_g, \quad (13)$$

$$V(\mathbf{x}_g) = 0, \quad \forall \mathbf{x}_g \in \mathcal{X}_g. \quad (14)$$

Hence, holding these conditions implies that the dynamical system will eventually reach the goal set.

B. DRL with Lyapunov-inspired reward function

This section comprises two main components. Firstly, we introduce a Lyapunov candidate function tailored for the control problem formalized earlier. Secondly, we elaborate on our deep reinforcement learning (DRL) strategy to train a control policy to satisfy all stability conditions utilizing the defined Lyapunov function. The Lyapunov function is constructed as a linear combination of two distinct functions, each catering to the objectives of efficient forward travel and obstacle avoidance, respectively.

The first candidate function, $V_{pose} : \mathbb{R}^4 \rightarrow \mathbb{R}$, depending only on the robot's pose is constructed as,

$$V_{pose}(\mathbf{x}_{robot}) = p_y^2 + \sqrt{(v_x - v_{des})^2 + v_y^2}. \quad (15)$$

The second candidate function, $V_{obs} : \mathbb{R}^4 \rightarrow \mathbb{R}$, inputting

the velocity and location of one obstacle, is defined as,

$$V_{obs}(\mathbf{v}, \mathbf{r}_i) = \left(\frac{|\mathbf{v} \cdot \mathbf{r}_i|}{\mathbf{r}_i \cdot \mathbf{r}_i} - \frac{|\mathbf{v}^\perp \cdot \mathbf{r}_i|}{\mathbf{r}_i \cdot \mathbf{r}_i} + \frac{|\mathbf{v}^\perp|}{|\mathbf{r}_i|} \right) \cos(\theta_i), \quad (16)$$

where \cdot defines the dot product, $\mathbf{v}^\perp = [v_y, -v_x]^T$ is perpendicularly rotated velocity vector, \mathbf{v} , and θ_i is the angle of the obstacle with respect to the body frame clipped to the obstacles in the forward area, defined as,

$$\theta_i = \begin{cases} \arctan 2(r_{y,i}, r_{x,i}), & r_{x,i} > 0, \\ 0, & r_{x,i} \leq 0. \end{cases}$$

The cosine multiplicand weights the function smoothly to zero while passing around the obstacle. The first term of the function assigns higher values when the robot's velocity is directed toward the obstacle, while the second term reduces the energy for velocities that revolve around the obstacle. Furthermore, the distance to the obstacle is inversely proportional to the function. Hence, following the negative gradient of this function enables the robot to move around the obstacles while getting closer to them.

Proposition 1: $V_{obs}(\mathbf{v}, \mathbf{r}_i)$ function is non-negative. The highest value obtained from the dot product in the subtracted term $\frac{|\mathbf{v}^\perp \cdot \mathbf{r}_i|}{\mathbf{r}_i \cdot \mathbf{r}_i}$ is calculated as,

$$\frac{|\mathbf{v}^\perp| |\mathbf{r}_i|}{|\mathbf{r}_i| |\mathbf{r}_i|} = \frac{|\mathbf{v}^\perp|}{|\mathbf{r}_i|},$$

when \mathbf{v}^\perp and \mathbf{r}_i are aligned. This amount is compensated by adding $\frac{|\mathbf{v}^\perp|}{|\mathbf{r}_i|}$. Hence, when \mathbf{v} and \mathbf{r}_i vectors are perpendicular, the function is equal to zero. $V_{pose}(\mathbf{x}_{robot})$ function is also non-negative since it adds two non-negative terms.

The Lyapunov function, then, is defined as a linear combination of (15) and (16),

$$V(\mathbf{x}) = k_p V_{pose}(\mathbf{x}_{robot}) + k_o \sum_{i=1}^{N_o} V_{obs}(\mathbf{v}, \mathbf{r}_i), \quad (17)$$

where k_p and k_o are positive real numbers for weighting each term.

Proposition 2 ($V(\mathbf{x}) = 0 \iff \mathbf{x} \in \mathcal{X}_g$): The function, $V(\mathbf{x})$, is zero only if the state is in the goal set.

Remark 3: $V(\mathbf{x})$ holds the condition (14) by Proposition 2. $V(\mathbf{x})$ holds the condition (12) since being non-negative by Proposition 1 and non-zero if not the state is in the goal set by Proposition 2.

Following the aforementioned remark, only the condition (13) should be satisfied by the trained control policy. Hence, a shaped reward function is defined as,

$$R_t = V(\mathbf{x}_t) - V(\mathbf{x}_{t+1}), \quad (18)$$

which gives a higher reward if the taken action decreases the Lyapunov candidate. The policy seeks to take actions with positive rewards and, hence, holds the Lyapunov condition since it is punished in the case of negative reward. Since this reward-shaping strategy indicates the necessary conditions, no terminal rewards are utilized. Hence, the discount factor, γ , is proposed to be low in the DRL objective defined in

(1). Therefore, the policy is prevented from learning to take negative reward actions to seek future positive rewards which violates the Lyapunov conditions. The terminal cases for an RL include a timeout in order to collect various samples and collisions. Furthermore, an exploration rule for the RL agent is also proposed. An episode is terminated when $R_t < -\epsilon$ restricts the exploration region with closer to stable actions.

V. EVALUATION AND RESULTS

A. Simulation environment and experimental design

Our methodology is implemented and tested using the Webots simulation software [19], which provides a realistic environment for training and evaluating our approach. To enable seamless interaction with RL algorithms, we have implemented the environment as an OpenAI gym [20] wrapper. Figure 3 illustrates the designed environment, featuring three circle-shaped obstacles with the radius of 0.2m. The robot is represented as a rigid body with Coulomb friction against the floor and is simulated using 10ms timesteps. The floor is constructed as 5m wide path.

The experiments are conducted with the number of concerned obstacles, $N_o = 3$. When an obstacle is passed it is replaced in the frontier of the robot. Obstacles are positioned randomly according to the following distribution,

$$\mathbf{r}_i \sim [\mathcal{U}(1, 3.5), \mathcal{U}(-1, 1)] + [x_{offset}, y_{offset}], \quad (19)$$

where $\mathcal{U}(\cdot, \cdot)$ is the uniform distribution within the provided limits. x_{offset} and y_{offset} are set to the robot's current position at the time of relocation of the obstacle.

B. Experimental evaluations

We conducted training on the policies using the PPO algorithm implementation from the `stable-baselines3` [21] Python package, on a simulated environment for a duration of two million timesteps. The policy updates were configured to occur every 2^{15} timesteps.

1) *Effect of reward discount factor and the exploration constraint:* In the first set of experiments, the effects of differing reward discount, γ , and constrained exploration, ϵ , parameters are explored. Typically, the episodic discount

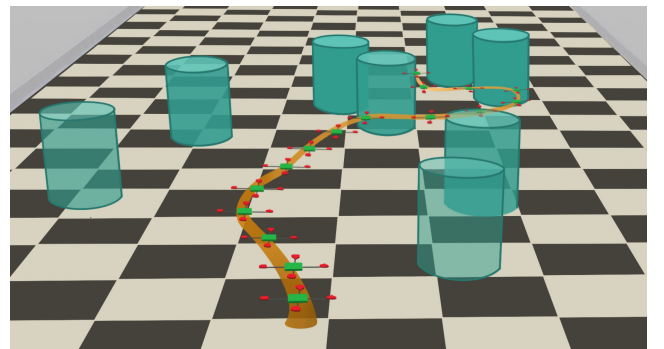


Fig. 3. The simulation environment with the trajectory of the quadrotor robot is visualized. The quadrotor deals with three obstacles at a time. Passed obstacles are replaced and shown transparently.

TABLE I

COMPARISON OF VANILLA REWARD DESIGN, APF-BASED REWARD DESIGN, AND DIFFERENT SETTINGS FOR THE PROPOSED LYAPUNOV-INSPIRED DRL. THE COMPLETION RATE, COLLECTED MEAN REWARD, AND AVERAGE VELOCITIES ARE REPORTED OVER 100 EPISODES.

	APF reward + PPO	Vanilla reward + PPO	Lyapunov-inspired PPO			
			$\gamma = 0.1$ $\epsilon = -0.1$	$\gamma = 0.99$ $\epsilon = -0.1$	$\gamma = 0.1$ $\epsilon = -0.006$	$\gamma = 0.1$ $\epsilon = -\infty$
Completion rate (%)	95	68	96	45	67	72
Mean reward ($\times 10^{-4}$)	4.1	4.2	4.9	4.5	-1.1	4.5
Average velocity	0.28	0.45	0.37	0.41	0.32	0.32

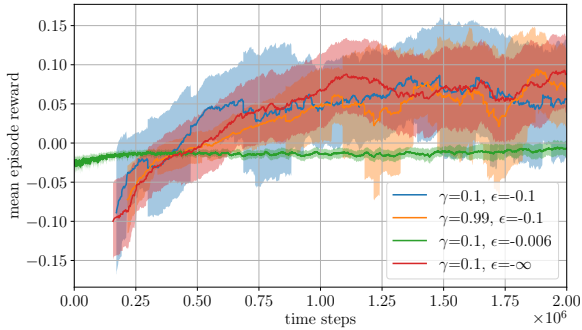


Fig. 4. Accumulated reward progress during training with different settings for the proposed Lyapunov-inspired strategy. The reward collected in every 100 episodes is averaged and one standard deviation region is shadowed for better visualization.

factor, γ , is chosen to be close to one to prioritize long-term reward seeking. However, in our proposed Lyapunov-based reward design, the function is uniformly distributed over the state space as there is no specific goal other than satisfying the Lyapunov conditions. Therefore, we conducted experiments with a value of $\gamma = 0.1$ and compared its performance against $\gamma = 0.99$.

The second set of experiments focuses on comparing the parameter ϵ used to terminate episodes when the reward falls below a certain threshold. This parameter assists the algorithm in constraining the search space by avoiding exploration in regions with high condition violations. However, setting ϵ too low can hinder overall exploration during training and lead to suboptimal performance. To address this, we compared the proposed value of $\epsilon = -0.1$ with two alternative cases: $\epsilon = -0.006$, which represents the typical lowest reward obtained by a trained model, and $\epsilon = -\infty$, indicating no termination based on low reward.

The collected reward during the training of the RL policy for the four settings described is depicted in Figure 4. The plot shows the average collected rewards every 100 episodes, with one standard deviation represented as a shaded area. Across all models, comparable learning speeds and reward levels are observed, except for the setting with constrained exploration ($\epsilon = -0.006$). In this case, although the trained model in the default setting is not subject to this constraint during our experiments, the algorithm fails to discover this policy due to limited exploration.

Table I presents the results of traversing 5m long trajectories for each trained policy, repeated 100 times. Consistent

with the training reward curves, the policy trained with over-constrained exploration ($\epsilon = -0.006$) was unable to achieve high rewards. In comparison, the proposed method exhibits a higher completion rate and can maintain a speed closer to 0.5m/s when compared to the unconstrained exploration ($\epsilon = -\infty$) scenario. Notably, the policy trained with a high discount factor ($\gamma = 0.99$) fails to complete the trajectory, despite being able to collect a comparable reward. It is important to emphasize that the average reward alone does not directly indicate success in our proposed formulation, as the policy may risk receiving negative rewards temporarily to ensure higher rewards in the future.

2) *Baseline - vanilla reward*: We have constructed a reward function inspired by DRL baselines [22, 1] for obstacle avoidance with depth images defined as,

$$R_{vanilla} = 1 - 0.5\sqrt{(v_x - v_{des})^2 + v_y^2} - 0.1|p_y|, \quad (20)$$

which informs the agent to keep the desired speed and centered position in non-terminal states. Furthermore, the agent gets -10 and -20 negative rewards for timeout and collisions respectively, and $+5$ and $+20$ positive rewards when an obstacle is passed or the episode length, 10m, is completed. In this case, the PPO algorithm is trained with discount factor, $\gamma = 0.99$, for 2.5 million timesteps to converge. Hence, the presented method provides a shorter training time for the same problem. Moreover, a similar test with 100 individual runs is conducted and reported in Table I. The reported reward is calculated with the Lyapunov-inspired reward for comparison. Although vanilla reward results in more accurate tracking for the desired velocity, it encounters collisions more often.

3) *Baseline - artificial potential field (APF) based reward design*: An APF-based [23] method is constructed as the second baseline. The conventional method generates an attractor vector for the goal and a repulsive vector for obstacles. The attractive vector is defined as,

$$\mathbf{pf}_{att} = [2, -p_y]^T, \quad (21)$$

indicating the desired velocity direction and correction in the y-axis. The repulsive vector is defined as,

$$\mathbf{pf}_{rep} = \sum_{i=1}^{N_o} \frac{-\mathbf{r}_i}{(|\mathbf{r}_i| - 0.2)^2}, \quad (22)$$

indicating the reverse direction from the obstacle inversely proportional to the distance to the obstacle. The summation of the attractive and the repulsive vectors is considered the

direction of the movement. A reference velocity vector is created with a magnitude of v_{des} as,

$$\mathbf{v}_{apf} = v_{des} \frac{\mathbf{p}\mathbf{f}_{att} + \mathbf{p}\mathbf{f}_{rep}}{|\mathbf{p}\mathbf{f}_{att} + \mathbf{p}\mathbf{f}_{rep}|} \quad (23)$$

The parameters of the vectors are tuned with a proportional velocity controller. Then a shaped reward is defined as the distance between the reference velocity and the current velocity as,

$$R_{apf} = |\mathbf{v} - \mathbf{v}_{apf}|. \quad (24)$$

The policy is trained with the PPO algorithm with discount factor, $\gamma = 0.1$, for 2 million timesteps similar to the proposed method. In this case, the training time is sufficient since the reward is well-shaped compared to the vanilla reward. The results are presented in Table I, similarly. Although the APF-based reward design gets a high completion rate similar to the presented method, it maintains a slower speed on average.

VI. CONCLUSION

This paper presents a novel DRL training strategy for addressing robot navigation problems by leveraging the principles of the Lyapunov theory. We formulate the robot planning problem in an obstacle environment as a state-space control problem, where obstacle locations are treated as part of the state representation. To guide the learning process, we introduce a reward-shaping strategy based on a Lyapunov function designed specifically for the formulated environment. Additionally, we propose a constrained exploration scenario for DRL training that incorporates the Lyapunov condition. Experimental results demonstrate that our proposed method achieves improved exploration, leading to the learning of a superior policy.

In the future, there are several directions for further exploration and improvement based on the findings of this research. Firstly, it would be valuable to extend the proposed DRL training strategy to more complex and dynamic environments, where the robot needs to adapt to changing obstacles or varying task requirements. Additionally, investigating the scalability of the approach to large-scale robotic systems would be crucial. Lastly, it would be interesting to explore the transferability and generalization capabilities of the trained policies to real-world environments or tasks. Addressing these aspects would contribute to a deeper understanding and practical applicability of the proposed DRL approach for robot navigation problems.

ACKNOWLEDGMENT

This work is 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.

REFERENCES

- [1] H. I. Ugurlu, X. H. Pham, and E. Kayacan, "Sim-to-real deep reinforcement learning for safe end-to-end planning of aerial robots," *Robotics*, vol. 11, no. 5, p. 109, 2022.
- [2] Y. Song, K. Shi, R. Penicka, and D. Scaramuzza, "Learning perception-aware agile flight in cluttered environments," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1989–1995.
- [3] K. Xu, Z. Hu, R. Doshi, A. Rovinsky, V. Kumar, A. Gupta, and S. Levine, "Dexterous manipulation from images: Autonomous real-world rl via substep guidance," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5938–5945.
- [4] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021.
- [5] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A lyapunov-based approach to safe reinforcement learning," *Advances in neural information processing systems*, vol. 31, 2018.
- [6] H. Dai, B. Landry, L. Yang, M. Pavone, and R. Tedrake, "Lyapunov-stable neural-network control," in *Robotics: Science and Systems*, 2021.
- [7] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [8] H. I. Ugurlu, S. Kalkan, and A. Saranlı, "Reinforcement learning versus conventional control for controlling a planar bi-rotor platform with tail appendage," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 4, pp. 1–17, 2021.
- [9] 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, pp. 529–533, 2015.
- [10] 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.
- [11] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.
- [12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
- [13] A. Raffin, J. Kober, and F. Stulp, "Smooth exploration for robotic reinforcement learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 1634–1644.
- [14] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.
- [15] H. K. Khalil, *Nonlinear control*. Pearson New York, 2015, vol. 406.
- [16] S. M. Richards, F. Berkenkamp, and A. Krause, "The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *Conference on Robot Learning*. PMLR, 2018, pp. 466–476.
- [17] H. Dai, B. Landry, M. Pavone, and R. Tedrake, "Counter-example guided synthesis of neural network lyapunov functions for piecewise linear systems," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 1274–1281.
- [18] K. Kang, P. Gradu, J. J. Choi, M. Janner, C. Tomlin, and S. Levine, "Lyapunov density models: Constraining distribution shift in learning-based control," in *International Conference on Machine Learning*. PMLR, 2022, pp. 10 708–10 733.
- [19] O. Michel, "Cyberbotics ltd. webots™: professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, p. 5, 2004.
- [20] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.
- [21] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, "Stable baselines3," 2019.
- [22] E. Camci, D. Campolo, and E. Kayacan, "Deep reinforcement learning for motion planning of quadrotors using raw depth images," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–7.
- [23] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 1985, pp. 500–505.

B Curiosity-Driven Reinforcement Learning based Low-Level Flight Control

Curiosity-Driven Reinforcement Learning based Low-Level Flight Control

Amir Ramezani Dooraki and Alexandros Iosifidis

Abstract—Curiosity is one of the main motives in many of the natural creatures with measurable levels of intelligence for exploration and, as a result, more efficient learning. It makes it possible for humans and many animals to explore efficiently by searching for being in states that make them surprised with the goal of learning more about what they do not know. As a result, while being curious, they learn better. In the machine learning literature, curiosity is mostly combined with reinforcement learning-based algorithms as an intrinsic reward. This work proposes an algorithm based on the drive of curiosity for autonomous learning to control by generating proper motor speeds from odometry data. The quad-copter controlled by our proposed algorithm can pass through obstacles while controlling the Yaw direction of the quad-copter toward the desired location. To achieve that, we also propose a new curiosity approach based on prediction error. We ran tests using on-policy, off-policy, on-policy plus curiosity, and the proposed algorithm and visualized the effect of curiosity in evolving exploration patterns. Results show the capability of the proposed algorithm to learn optimal policy and maximize reward where other algorithms fail to do so.

I. INTRODUCTION

Humans and intelligent creatures can learn in different ways; among them is learning by experience. Further, they use a spectrum of motivations: some are triggered internally (intrinsic motivations), and some are triggered externally and by the environment (extrinsic motivations). An intelligent creature learns to act in the direction of responding to its motivations. Observing this paradigm in nature, machine learning and control communities created the framework of the Markov Decision Process (MDP) and Reinforcement Learning (RL) algorithms to replicate this optimization process in robots and machines. Further, several computational models of intrinsic motivations, such as curiosity, have been implemented in the past decades.

At the same time, considering the advances in computer hardware, different intelligent algorithms for autonomous control of ground, aerial, underwater, surface, and legged robots have been created during the past decade. Specifically, these advancements were significant for multi-copter drones where the weight of the whole robot is significant in terms of its ability to fly and maneuver capabilities in three axes. Nowadays, it is possible to see the application of Unmanned Aerial Vehicles (UAVs) such as quad-copters in several areas. Autonomous inspection, search and rescue missions, and navigation in unknown environments are examples of high-level control where an algorithm takes high-level decisions and passes it to a low-level controller for execution using

robots' actuators. Autonomous learning of aggressive maneuvers, drone racing, and fault-tolerant control are examples of low-level flight controllers where the algorithm directly controls the actuators.

Combining the framework of learning-based algorithms, such as reinforcement learning and deep reinforcement learning, with UAVs pushed their autonomous control to new frontiers, allowing them to autonomously learn to control both in high-level and low-level state spaces.

This paper proposes a new reinforcement learning-based low-level flight controller that learns by parameterized intrinsic (a computational model of curiosity) and extrinsic (external immediate and auxiliary rewards) motivations to directly control the quad-copter's motor speeds and flies toward the desired position while avoiding obstacles. Our contributions are summarized as follows:

- We propose a new approach for learning low-level flight policy using parameterized curiosity module.
- In our proposed approach, we consider both passing through obstacles and controlling the Yaw direction towards the desired location.
- To achieve the mentioned contributions, we propose a new approach for calculating the curiosity reward based on the prediction error.

In the rest of this paper, first, we discuss the related literature and the difference between our work and other related works. Next, we describe the proposed methodology including the reinforcement learning approach, the curiosity module, the simulation environment, and the visualization of the curiosity effect. Then, we describe the experimental evaluation and provide results along with a discussion about important matters related to our work. Finally, we provide concluding remarks.

II. LITERATURE REVIEW

In the literature on unmanned aerial vehicles' control using reinforcement learning, a wide range of works exists that can be divided into two main groups, namely high-level and low-level control. By low-level control, we refer to the direct control of motors by providing their actual angular velocity or by providing thrust, roll, pitch, and yaw. Low-level control works include [1] where RL is used to learn for direct control of UAV motors speeds from odometry, [2] where a combination of RL and PD is used to train the controller, [3] where RL used to learn more general policies for low-level quad-copter control, and [4] where RL is used for control and tracking of a trajectory. By high-level control, we refer to a trajectory of waypoints or attitudes generated by the controller. High-level control works include [5] where

A. Ramezani Dooraki and A. Iosifidis are with the Department of Electrical and Computer Engineering, Aarhus University, 8000 Aarhus C, Denmark {amir, ai} at ece.au.dk

an end-to-end approach using deep RL is used to learn to control three axes of quad-copter when RGB-D image is provided as input.

Curiosity as an intrinsic motive is observed widely in the literature. In some early works, such as [6] and [7], the authors defined a framework of intrinsic motivation and curiosity as one such motivation. In recent years, and considering the new computational capabilities offered by advancements in hardware systems, more realistic computational models of curiosity have been researched and developed in the literature. As a result, different types of curiosity methods, such as Information Theoretic based [8], Prediction based (e.g., surprisal [9]), and Count based [10], have been proposed. For example, a module called intrinsic curiosity module (ICM) is used in [9] to predict the future states and actions that need to be taken to reach those states, and considered the prediction error between the actual future states and predicted states as the curiosity reward. Further, there are works such as [10] which considered curiosity as a measure to count how many times a state is visited during the agent’s lifetime. Curiosity has been used to achieve specific qualities in robot actions. For example, [11] used curiosity to achieve gentle touch while grasping objects. In other literature, it used to increase the performance of well-known objectives in robotics. For example, in [12] curiosity is used for motion planning of humanoid robots, in [13] for control of swarm of robots, and in [14] for robot navigation.

Curiosity in the low-level control of the quad-copter is an area that is much less researched in the literature. In some works, curiosity is defined as a measure of difference rather than surprise or novelty. For example, [15] defines curiosity as the difference between states observed by a policy at two different times. In this work, curiosity is defined as a measure of novelty and a parameterized function that gradually learns the previously visited states, loses its interest in them, and constantly searches for novel states.

III. METHODOLOGY

A. Main Algorithm

The algorithm proposed in this paper is comprised of several parts which are illustrated in Figure 1. We introduce an RL-based approach for training a curiosity-driven policy network. Curiosity is incorporated to direct the exploration of the RL method, especially in complex scenarios. Our algorithm is designed to solve the problems formulated in a Markov Decision Process (MDP) [16] framework. The goal of the algorithm is to optimize the policy and value networks in a way that they could maximize the long-term rewards received from the environment and generated by the curiosity module. Further, an environment is designed and implemented to test the capabilities of the proposed algorithm and compare it with other approaches. In the following, different parts of the algorithm are described in detail.

B. Reinforcement Learning

The standard formulation of RL-based algorithms is used in this paper. Subsets of $s_t \in S$, $a_t \in A$, and $r_t \in R$ are defined for states, actions, and rewards. The initial starting state D is defined as a set of possible initial states for the agent. Further, the following standard definitions are used throughout the paper:

$$\begin{aligned} Q_\pi(s_t, a_t) &= \mathbb{E}_{s_{t+1}, a_{t+1}, \dots} \left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l+1}) \right], \\ V_\pi(s_t) &= \mathbb{E}_{a_t, s_{t+1}, \dots} \left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l+1}) \right], \\ A_\pi(s, a) &= Q_\pi(s, a) - V_\pi(s). \end{aligned}$$

where $\gamma \in (0, 1)$ is a discount factor, and $r(s_t)$ is the reward the agent receives at time t .

C. Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) [17] is used in this paper as the RL algorithm for optimizing the weights of the policy and value networks. PPO is an algorithm based on and surpassing Trusted Region Policy Optimization (TRPO) [18]. This paper briefly discusses the TRPO and PPO optimization objectives before discussing the multiple value heads used in our proposed method.

TRPO updates the policy and value networks parameters by solving the following constrained optimization problem:

$$\begin{aligned} \max_{\theta} \mathbb{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A_t \right], \\ \text{subject to: } \mathbb{E}_t \left[KL(\pi_{\theta_{old}}(\cdot | s) || \pi_\theta(\cdot | s)) \right] \leq \delta. \end{aligned} \quad (1)$$

The constraint can be incorporated in the form of a penalty weighted with a coefficient β :

$$\max_{\theta} \mathbb{E}_t \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} \hat{A}_t - \beta KL[\pi_{\theta_{old}}(\cdot | s), \pi_\theta(\cdot | s)] \right]. \quad (2)$$

The conservative policy iteration (CPI) [19] corresponds to the following surrogate objective:

$$L^{CPI}(\theta) = \mathbb{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] = \mathbb{E}_t \left[r_t(\theta) \hat{A}_t \right], \quad (3)$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is a probability ratio comparing policy with parameters θ with the old policy with parameters θ_{old} , thus, $r(\theta_{old}) = 1$.

PPO maximizes the Equation 3 and penalizes changes that move $r_t(\theta)$ away from 1 using the following equation, instead of using constraint:

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

where ϵ is a hyper-parameter, that is set equal to 0.2 in this paper.

The complete loss function defined in PPO is:

$$L_t^{CLIP+VF+S}(\theta) = \mathbb{E}_t \left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) - c_2 S[\pi_\theta](s_t) \right] \quad (5)$$

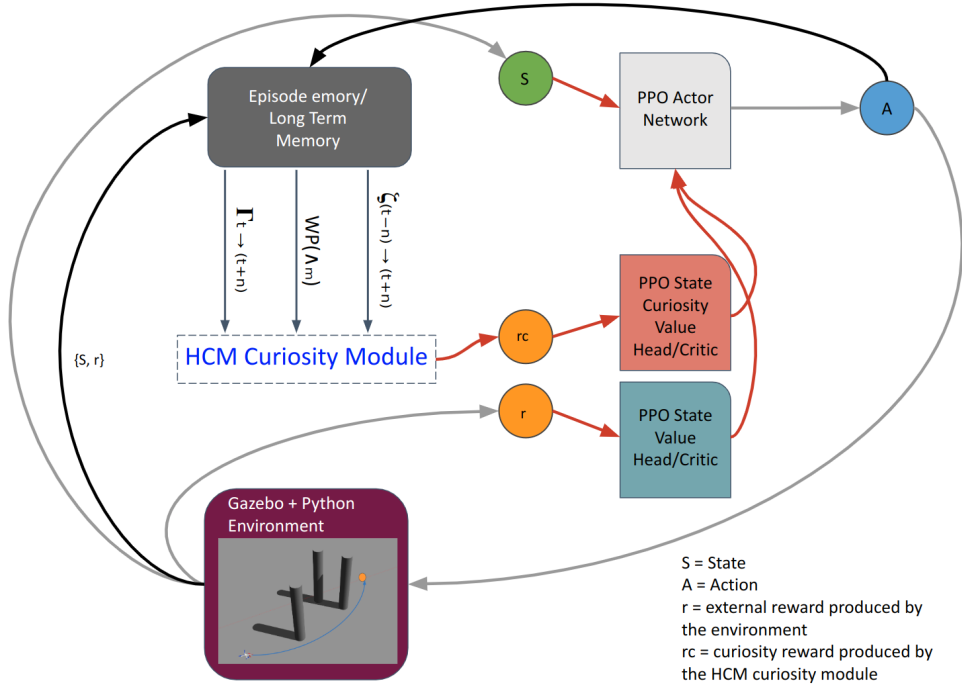


Fig. 1. The flow of data in our algorithm including the policy network, value networks (value heads), and curiosity module comprised of curiosity heads.

where c_1 and c_2 are hyper-parameters, S is an entropy bonus similar to entropy of policy mentioned in [20]. L_t^{VF} is a squared-error loss, i.e., $L_t^{VF} = (V_\theta(s_t) - V_t^{targ})^2$, where $V_t^{targ} = \hat{A}_t + V_{\theta_{old}}(s_t)$. To calculate \hat{A}_t , a truncated version of generalized advantage estimation [21] is used:

$$\begin{aligned} \hat{A}_t &= \delta_t + (\gamma\lambda)\delta_{t+1} + \dots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \\ \delta_t &= r_t + \gamma V_\theta(s_{t+1}) - V_\theta(s_t), \end{aligned} \quad (6)$$

where $\lambda \in (0, 1)$ is an exponential discount factor.

D. Multiple Value-Function Heads for PPO

As mentioned above, our method incorporates a curiosity module which is described in Section III-H. We used it in our algorithm to measure the interestingness of states and direct the exploration of our algorithm.

The subtle and challenging part about a curiosity method based on neural networks is that its output changes after every n episodes of training. In other words, the state value (that is, how good it is for the agent to be in a specific state) for the RL algorithm changes and is not constant such as in a standard approach where the states' values are approximated by rewards received from a constant function. The reward in such cases is the output of the reward function and is always constant for a specific pair of 'state and action'. In the case of a curious algorithm, a pair of 'state and action' could produce a high prediction error and be interesting at time t (when the state visited for the first time by the agent), and after a couple of training epochs, the neural network output would change, resulting in a new curiosity value (which considering the training, would be less than the previous curiosity value).

Thus, in a setting with external and curiosity rewards, the RL algorithm must learn a value function that changes over time (that is, the sum of extrinsic and intrinsic values). To have a more stable learning, we use the idea of separating the parameterized value functions. We define two value heads for the PPO, the first value head called 'State Extrinsic Value Head' learns the extrinsic value of the state (State Extrinsic Value) generated based on the external rewards, which is constant and comes from the environment, and the second parameterized value function called 'State Intrinsic Value Head' learns the intrinsic value of the state (State Intrinsic (or Curiosity) Value), which is dynamic (that is, changes over time) and is generated based on the curiosity reward. Using this approach makes it possible to 1) stabilize the learning of the value function by using separate heads, and 2) use different learning rates for each one of the heads, thus being able to control the rate of learning the value of external reward and curiosity separately. Figure 1 shows the flowchart of our algorithm.

As a result, there is one extra loss function for the curiosity value head:

$$\begin{aligned} L_t^C(\theta) &= V_{C_\eta}(s_t) - V_{C_t}^{targ}, \\ V_{C_t}^{targ} &= \hat{A}_{C_t} + V_{C_{\theta_{old}}}(s_t). \end{aligned} \quad (7)$$

Further, the δ_t estimation mentioned in Equation 6 would change as the following:

$$\begin{aligned} \delta_t &= (r_{ext} + r_{int}) + \gamma \left((V(s_{t+1}) - V(s_t)) \right. \\ &\quad \left. + (V_C(t+1) - V_C(t)) \right). \end{aligned} \quad (8)$$

r_{ext} and r_{int} are rewards generated by the environment and curiosity module and are described in Section III-O.

E. Training a Quad-copter

We designed and implemented an environment (explained thoroughly in Section III-N) to evaluate the performance of the proposed method and to compare it with that of other methods. In the case of a quad-copter control using reinforcement learning, the agent should generate 100 actions per second (100 Hz) and collect the state, action, and reward in each time-step. We execute the training at the end of each episode of 16,394 steps, mainly because it is very stable, but the algorithm can be trained in every 4,096 steps or 8,192 steps, where the training would be less stable.

F. Policy and Value Networks

The policy network is a neural network with two hidden layers, each formed by 256 neurons. The input to the network (the agent state or perception of the world) is formed by the odometry data (including both linear and angular accelerations), the previous motor speeds (that can be considered as the proprioception of quad-copters' motors state), the distances of the obstacles to the agent, and the distance of the agent to the goal. Distances are calculated based on the X and Y coordinates. For each obstacle the distance comprises of the distance in X coordinate, the distance in Y coordinate, and Euclidean distance based on the previous two measures, while the distance to the goal is comprised of only the Euclidean distance based on X and Y. The odometry part of the state is introduced as input to the network, while the rest of the state is concatenated to the output of the first layer, which is subsequently introduced to the second layer of the network. The output of the policy network is a 4-dimensional vector that is used to generate the action. An identical structure is used for the state value network and the curiosity value network, except for the output, which is the state value of a particular observation. All networks are optimized using the Proximal Policy Optimization (PPO) algorithm.

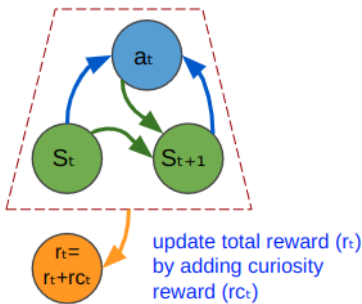


Fig. 2. The dynamics between state, action, and reward in the ICM [9]. The method is discussed briefly at the beginning of Section III-G.

G. Intrinsic Curiosity Module

We used the idea of curiosity based on the prediction error similar to the method in [9]. In the original Intrinsic

Curiosity Module (ICM) method, two models are defined for calculating curiosity. The first model, called inverse dynamics, is used to learn to predict the action that the agent took to move from S_t to S_{t+1} , i.e.,:

$$\hat{a} = g(s_t, s_{t+1}; \theta_I), \quad (9)$$

and a loss function is used for reducing the error between the predicted action and the actual action:

$$\min_{\theta_I} L_I(\hat{a}_t, a_t). \quad (10)$$

Further, there is a forward dynamics model that is responsible for the prediction of the state feature $\phi(s_{t+1})$ based on the state $\phi(s_t)$ and action a_t :

$$\hat{\phi}(s_{t+1}) = f(\phi(s_t), a_t; \theta_F) \quad (11)$$

with its corresponding loss function being:

$$L_F(\phi(s_t), \hat{\phi}(s_{t+1})) = \frac{1}{2} \left\| \hat{\phi}(s_{t+1}) - \phi(s_{t+1}) \right\|^2. \quad (12)$$

The intrinsic (curiosity) reward in each time-step r_t^i is then defined to be the error of prediction between the actual feature state $\phi(s_{t+1})$ and the predicted feature state $\hat{\phi}(s_{t+1})$, i.e.,:

$$r_t^i = \frac{\eta}{2} \left\| \hat{\phi}(s_{t+1}) - \phi(s_{t+1}) \right\|. \quad (13)$$

The overall loss function for the curiosity module is defined to be:

$$\min_{\theta_I, \theta_F} \left[(1 - \beta)L_I + \beta L_F \right]. \quad (14)$$

Figure 2 shows a schematic of the dynamics of the described approach.

Overall, while the above-described method is a capable approach useful in some problems, it cannot help our algorithm learn to maximize the reward and, as shown in Section IV, it fails to effectively train the agent. Therefore, we propose a new curiosity approach, explained in the next section.

H. High-level Curiosity Module

The approach this paper proposes for calculating curiosity is based on the prediction error with the following modifications:

- A new state space for the curiosity module that uses the segment of states instead of a normal state.
- A pre-processing function that transforms the segment of actions before using it by the curiosity module.
- In addition to predicting the curiosity using the dynamics in the state and action spaces, our method uses the dynamics between the state, action, and external rewards for predicting the curiosity.
- Instead of using a single network, our algorithm incorporates an ensemble of curiosity networks, as shown in Figure 5.
- Finally, instead of calculating a curiosity reward for a single step, our method calculates and updates a trajectory of the steps using a decay factor.

We call it High-level Curiosity Module (HCM) because it learns based on the trajectory of the agent’s interaction with the environment. As a result, it is a high-level approach compared to learning from a single interaction. Further, as Curiosity is an intrinsic value by nature, we do not use the intrinsic name in the title of the algorithm.

I. Curiosity in States-Actions-States Space

Considering a low-level control problem where the agent receives sensor data 100 times per second, the changes between the states would be minor, generating a proper curiosity reward would be challenging. One way to address this issue is to modify the state, action, and reward space for the curiosity module. In order to make a curious agent that can converge to an optimal solution, we create a segment of states called ζ where it can select a set of states in backward pass or forward pass, considering the current time $T = t$. We create a segment of the trajectory of the agent with length n steps starting back in time and ending at the current time-step, i.e., $\zeta_{(t-n)\rightarrow t} = \{s_{t-n}, s_{t-(n-1)}, \dots, s_{t-0}\}$. We also create a second segment of the trajectory of the agent starting at the current time-step and ending n steps into the future, i.e., $\zeta_{t\rightarrow(t+n)} = \{s_t, s_{t+1}, \dots, s_{t+n}\}$. Intuitively, it is necessary to use a segment of actions for the transition between $\zeta_{(t-n)\rightarrow t}$ to $\zeta_{t\rightarrow(t+n)}$, which we call it Λ . While all the actions for moving from $\zeta_{(t-n)\rightarrow t}$ to $\zeta_{t\rightarrow(t+n)}$ are effective, the actions that are more near to the connection of the two segments (or trajectories) are more important. Thus, considering only half of the actions from each segment would be enough. So, $\Lambda_m = \{a_{t-m}, \dots, a_{t+m}\}$, i.e., it comprises of the set of actions starting from time-step $T = t - m$ to $T = t + m$, where $m = n/2$. However, training a neural network to predict Λ_m based on $\zeta_{(t-n)\rightarrow t}$ and $\zeta_{t\rightarrow(t+n)}$ (that is, $\Lambda_m = g_{SS}(\zeta_{(t-n)\rightarrow t}, \zeta_{t\rightarrow(t+n)}; \theta_{ISS})$ where θ_{ISS} is the set of the g_{SS} model parameters) is not trivial and does not produce good results, mainly because of the size of Λ_m . A better solution is to first convert the Λ_m to a low-dimensional vector that characterizes the transition. To do this, we define a function called F_{WP} .

One general approach would be to define F_{WP} as a convolutional neural network or variational auto-encoder that extracts the features or the latent space of the Actions segment and consider it as the low-dimensional vector that characterizes the transition. However, considering that we have access to positions and attitudes in our problem, an easier approach is to directly calculate the waypoints that indicate the transition. A waypoint could be considered as a ‘position’ or a combination of ‘position and attitude’ transition. Here, we consider it as a ‘position’ transition because our goal is to reduce the dimensionality of the Actions segment. Thus, instead of using the action space to characterize the transition, we consider the change in position space as the transition. Moreover, to capture the transition adequately well, more than one waypoint is needed. We use three waypoints in order to have a measure from the beginning, middle, and end of the transition. As a result, the F_{WP} input is Λ_m , a subset of $\{\zeta_{(t-m)\rightarrow t}, \zeta_{t\rightarrow(t+m)}\}$

comprised of only position data where $m = n$. m could be smaller or larger than n in general approach, such as $m = n/2$. The output of F_{WP} is $\{wp_1, wp_2, wp_3\}$.

Our parameterized inverse dynamics function called g_{SS} predicts the output of F_{WP} and is defined as follows:

$$\hat{F}_{WP}(\Lambda_m) = g_{SS}(\zeta_{(t-n)\rightarrow t}, \zeta_{t\rightarrow(t+n)}; \theta_{ISS}), \quad (15)$$

where θ_{ISS} is the set of the g_{SS} model parameters, and the loss function becomes:

$$\min_{\theta_{ISS}} L_{ISS}(\hat{F}_{WP}(\Lambda_m), F_{WP}(\Lambda_m)). \quad (16)$$

Further, the forward dynamics are defined as follows:

$$\hat{\phi}(\zeta_{t\rightarrow(t+n)}) = f_{SS}\left(\phi(\zeta_{(t-n)\rightarrow t}), F_{WP}(\Lambda_m); \theta_{FSS}\right) \quad (17)$$

where ϕ is the function to extract the features, and θ_{FSS} is a set of the f_{SS} model parameters. We then minimize the following loss function:

$$L_{FSS}(\phi(\zeta_{t\rightarrow(t+n)}), \hat{\phi}(\zeta_{t\rightarrow(t+n)})) = \frac{1}{2} \left\| \hat{\phi}(\zeta_{t\rightarrow(t+n)}) - \phi(\zeta_{t\rightarrow(t+n)}) \right\|^2. \quad (18)$$

The curiosity reward generated by this head is calculated in the following way:

$$r_{CSS} = \left[(1 - \beta)L_{ISS} + \beta L_{FSS} \right]. \quad (19)$$

Finally, the total loss function is:

$$\min_{\theta_{ISS}, \theta_{FSS}} \left[r_{CSS} \right]. \quad (20)$$

J. Curiosity in States-Actions-Rewards Space

Considering that an immediate reward exists in our problem setting, generating the curiosity reward would be more complex because the policy constantly receives the reward from the environment, making the problem of generating a proper curiosity reward more challenging. In order to make a curious agent that can converge to the optimal solution, we define a new pair of forward and inverse dynamics models where instead of predicting the next state, they would work and predict the segment of external reward. In a nutshell, we use the $\zeta_{(t-n)\rightarrow t}$ and F_{WP} as they described in the previous section. Further, we define a segment of external rewards and show it by $\Gamma_{t\rightarrow(t+n)}$. Also, we define a new set of functions for inverse and forward dynamics:

$$\hat{F}_{WP}(\Lambda_m) = g_{SR}(\zeta_{(t-n)\rightarrow t}, \Gamma_{t\rightarrow(t+n)}; \theta_{ISR}), \quad (21)$$

where θ_{ISR} is set of the g_{SR} model parameters, and the loss function

$$\min_{\theta_{ISR}} L_{ISR}(\hat{F}_{WP}(\Lambda_m), F_{WP}(\Lambda_m)). \quad (22)$$

Further, the forward dynamics are defined as follows:

$$\hat{\phi}(\Gamma_{t\rightarrow(t+n)}) = f_{SR}(\phi(\zeta_{(t-n)\rightarrow t}), F_{WP}(\Lambda_m); \theta_{FSR}) \quad (23)$$

where ϕ is the function to extract the features, and $\theta_{F_{SR}}$ is a set of the f_{SR} model parameters. We then minimize the following loss function:

$$L_{F_{SS}}(\phi(\Gamma_{t \rightarrow (t+n)}), \hat{\phi}(\Gamma_{t \rightarrow (t+n)})) = \frac{1}{2} \left\| \hat{\phi}(\Gamma_{t \rightarrow (t+n)}) - \phi(\Gamma_{t \rightarrow (t+n)}) \right\|^2. \quad (24)$$

The curiosity produced by this head is:

$$r_{C_{SR}} = \left[(1 - \beta)L_{I_{SR}} + \beta L_{F_{SR}} \right], \quad (25)$$

and the total loss function is:

$$\min_{\theta_{I_{SR}}, \theta_{F_{SR}}} \left[r_{C_{SR}} \right]. \quad (26)$$

K. Ensemble of Curiosity Sub-modules

In the previous sections, two curiosity sub-modules were defined, namely 1) curiosity error based on prediction error in $(\zeta, WP(\Lambda_m), \zeta)$ transition space (called curiosity module part a), and 2) curiosity error based on the error of prediction in $(\zeta, WP(\Lambda_m), \Gamma)$ transition space (called curiosity module part b). To have a less biased curiosity module, we define an ensemble of curiosity similar to [22] in terms of using multiple networks to measure the prediction error. In our setting, our final curiosity module comprises of five curiosity sub-modules type ‘a’ and five curiosity sub-module type ‘b’, as illustrated in Figure 5. The output of the curiosity module then is the average of all networks’ outputs, and the final formula for calculating the curiosity reward consisting of n ensemble of networks (for each separate sub-module) is:

$$r_{curiosity} = \alpha_{curiosity} * \left(\frac{1}{2n} \sum (r_{C_{SS_n}} + r_{C_{SR_n}}) \right), \quad (27)$$

where $\alpha_{curiosity}$ is a coefficient for curiosity reward.

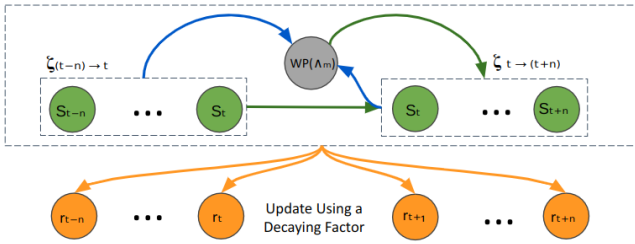


Fig. 3. Part one of our proposed curiosity method where the prediction error is based on the state segments and action segments.

L. Updating a Trajectory of State Curiosity Values

Considering that the curiosity reward generated by our method is generated based on $(\zeta, WP(\Lambda), \Gamma)$, a segment of states, actions, and rewards, we convert the generated curiosity reward to a trajectory of curiosity rewards and use it to update a trajectory of State Curiosity Values as follows:

$$r_{c_{t \pm x}} = r_{c_{t \pm x}} + \kappa^x (r_{curiosity}), \quad (28)$$

where $x = 0, 1, \dots, n$ and $\kappa \in (0, 1)$ is a decay factor.

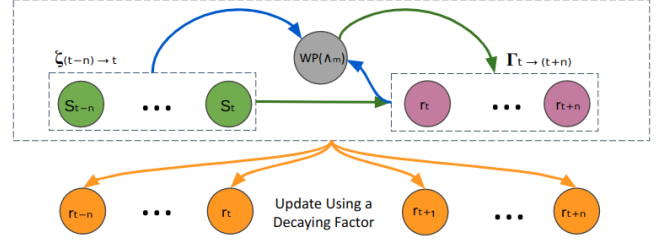


Fig. 4. Part two of our proposed curiosity method where the prediction error is based on the state segments, action segments, and reward segments.

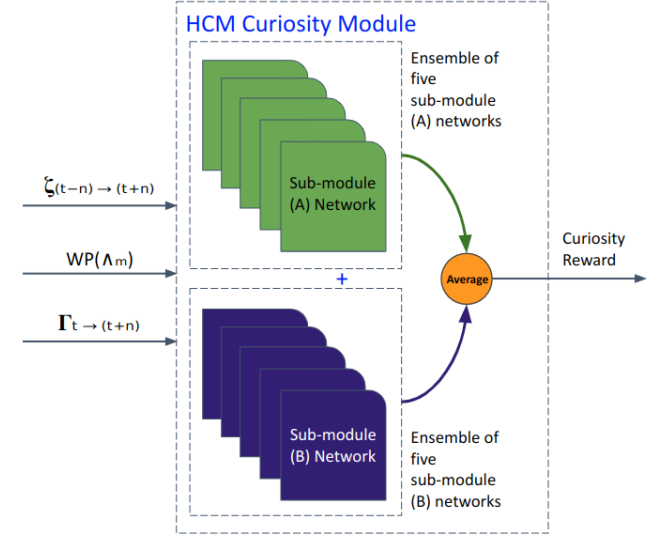


Fig. 5. Overall architecture of our curiosity module where n Module A and n Module B heads are used to create the curiosity module.

M. Visualizing the Effect of Curiosity

One way to visualize the effect of curiosity is to trace and visualize the trajectories traversed by the robot/agent, which is the method we used and described here. We first consider the XY plane of the agent position and trace its movement in that plane. In order to trace its movement, we divide the XY plane into X columns and Y rows and define a visitation value for each cell. Further, each time the agent passes through a row and column, we add the visitation value of that cell by +1. Finally, we normalize the matrix that collects this information and visualize it as an image. A sample image that shows the agent’s movement in the environment is shown in Figure 7.

N. Environment

The environment is an essential part of an RL-based approach considering it generates the new state s_t and external reward r_t by executing the action a_t . The environment used in this work is based on the Gazebo simulator [23] and the RotorS package [24], which is used to simulate a model of Ascending Technology Hummingbird. The environment is comprised of a quad-copter and three obstacles. To make the environment realistic and reduce the gap between the

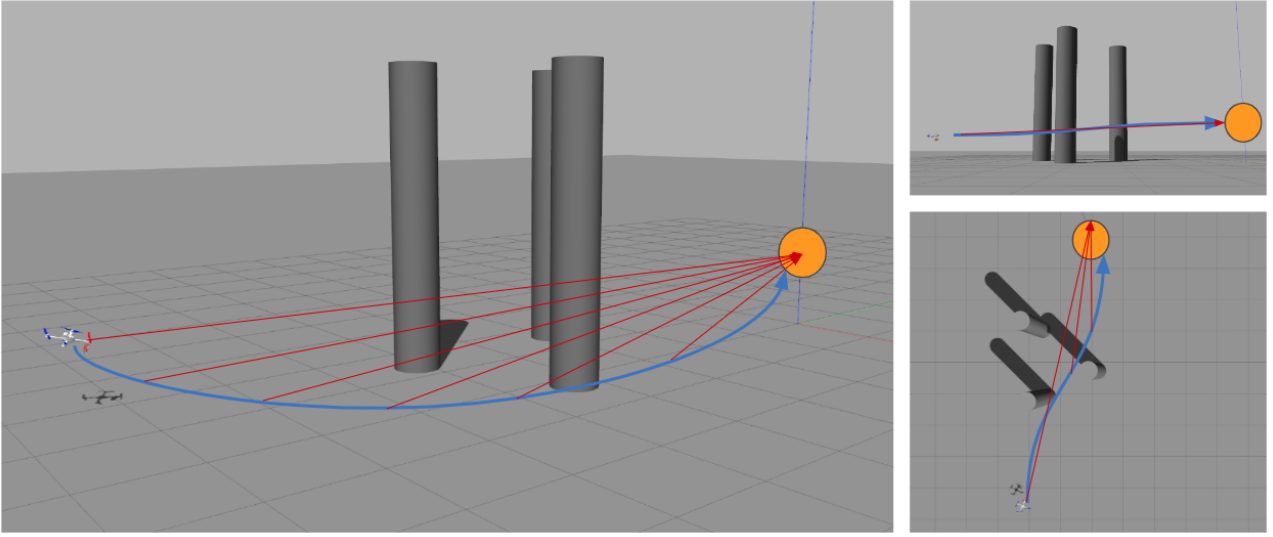


Fig. 6. Visualization of our environment (3D view (left), side view (top right), and top view (bottom right)), which is comprised of a quad-copter and three obstacles initialized in a random position between the drone and the desired position. The goal is to 1) pass the obstacles and reach a goal destination known by the algorithm by a position and attitude pair (the blue line draws a sample desired path), and 2) control the Yaw towards the desired position (in this case, the coordinate frame origin), the red lines visualize the desired Yaw directions.

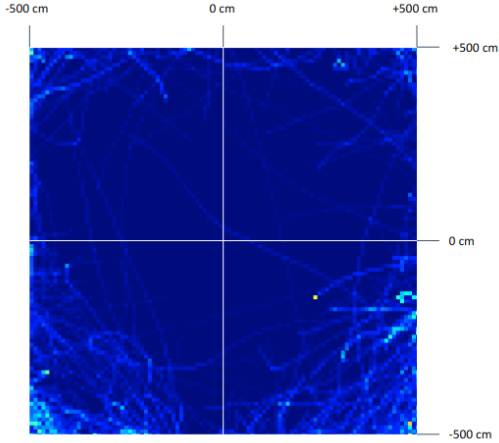


Fig. 7. The trajectories traversed by the agent in the environment from a top-view point of view (i.e., in the XY plane) after 50 episodes.

simulation and real world our environment adds noise to the motors and simulate the damping effect. The primary goal for the quad-copter agent is to learn to fly and control itself from any random initial position and attitude. The secondary goal is explained using the following items:

- Pass the obstacles and fly towards the desired position and attitude.
- Control the yaw direction toward the desired position in terms of X and Y.

Besides that, a Python script is used to communicate with the Gazebo simulator to retrieve the new state, generate the extrinsic rewards after passing the action generated by the policy network, detect the terminal states and reset the

quad-copter position, and gather all the statistical information related to the quad-copter, the extra goals, and errors. The communication between the Python script and the Gazebo simulator is handled by using Robot Operating System (ROS) [25] to enhance the mobility of the code.

O. Reward

This work defines three types of rewards for training our RL-based policy:

- An immediate extrinsic reward is generated by the environment in each time step; this reward is a measure of how near the agent is to the desired position and attitude and is defined using the following formula:

$$r_{flight} = \alpha_p \left(\left(1 - \frac{|x_d - x_c|}{x_{max}}\right) + \left(1 - \frac{|y_d - y_c|}{y_{max}}\right) + \left(1 - \frac{|z_d - z_c|}{z_{max}}\right) \right) - \alpha_a \left(|\theta_d - \theta_c| + |\phi_d - \phi_c| + |\psi_d - \psi_c| \right), \quad (29)$$

where (x, y, z) denotes the position coordinates and (θ, ϕ, ψ) are the attitude of the agent. The index d indicates the desired position and attitude, which are $x_d = 0m$, $y_d = 0m$, $z_d = 1.5m$, $\theta_d = 0$, $\phi_d = 0$, and ψ_d is equal to the Yaw that points towards the origin of the world coordinate system. The index c indicates the current position and attitude. α_p and α_a are coefficients for error in position and error in attitude, respectively. Finally, we use a shaping reward technique to motivate agent activity in the desired areas by defining a threshold for x, y, and z axes. If the agent distance in a specific axis gets more than a predefined threshold, the position reward for that axis would be

−1.0. For example, if $|x_d - x_c| > \frac{x_{max}}{2}$ then we consider −1.0 instead.

- Two auxiliary immediate extrinsic rewards that are generated by the environment. The first one is related to the control of the yaw toward the goal direction, as shown in the following:

$$r_{yaw} = |\psi_d - \psi_c|, \quad (30)$$

where ψ is the Yaw of the attitude of the agent, the index d indicates the desired Yaw value, and the index c the current Yaw value.

The second one is related to the velocity of the agent as shown in the following:

$$r_{vel} = \alpha_\nu * \|\nu\| + \alpha_\omega * \|\omega\|, \quad (31)$$

where ν and ω are the linear and the angular velocity of the quad-copter, respectively, and α_ν and α_ω are coefficients.

- An intrinsic immediate reward that is generated by the curiosity module. This reward represents the surprise and motivates the agent’s exploration and is fully explained in Section III-H.

Finally, the final reward is calculated in two parts, considering the two Value Heads defined for the PPO (i.e., State Value Head and State Curiosity Value Head), the internal reward explained in Section III-H, and the external reward is formulated as shown in the following:

$$r_{ext} = \begin{cases} (r_{flight} * \alpha_{flight}) + (r_{yaw} * \alpha_{yaw}) + r_{vel}, \\ -10, \text{ if quad-copter hits obstacle or crashes.} \end{cases} \quad (32)$$

IV. EXPERIMENTS

In order to illustrate the performance of the proposed algorithm we compared it with other powerful algorithms which serve as the baselines:

- PPO is used as the baseline on-policy algorithm for testing and performance comparison in our tests.
- SAC is used as an off-policy algorithm with memory replay to compare its performance with the other on-policy algorithms mentioned in this section.
- PPO+ICM is the PPO algorithm combined with ICM [9] module (for the curiosity reward).
- PPO+HCM is the PPO algorithm combined with our proposed curiosity approach (i.e., HCM).

The PPO algorithm we used here as the baseline is a highly tuned PPO algorithm against our fly environment with exploration noise generated by a Gaussian distribution with a mean of 0 and standard deviation of 1.0. The default parameters used for the SAC algorithm (the stable baseline version). We tested the performance of the competing algorithms by running our scenario multiple times. Each algorithm was run six times, and the min, max and average results are calculated. Our code can be found in the corresponding GitHub repository¹.

¹https://github.com/a-ramezani/CDRL-L2FC_u_HCM

A. Reward Maximization and Quad-copter Low-level Control

Reward maximization is the main goal in reinforcement learning-based algorithms. Figure 8 illustrates the performance of the competing algorithms. Looking at the left side of the figure, one can observe that only PPO+HCM (i.e., our proposed curiosity-based algorithm) is able to maximize the reward over time. In other words, only PPO+HCM can learn to perform according to the goals mentioned in Section III-N in the environment. The right side of the same figure displays some information about Failed Fly. A ‘Failed Fly’ is when the quad-copter is initiated randomly in the environment, the algorithm cannot learn to control it and, as a result, the quad-copter crashes. Again, only PPO+HCM can control the quad-copter and reduce the number of ‘Failed Fly’ over time, which is another sign that the algorithm is successful in the environment.

Considering the problem our algorithm tries to solve is controlling a quad-copter, the averaged position and attitude errors are collected and displayed in Figure 9. The information shown in the figure can be divided into two categories 1) Attitude Control Information, and 2) Position Control Information. Reducing errors for Attitude Control implies that the algorithm can control the quad-copter and not crash. However, it does not give any information about how far or near the quad-copter is from the desired position. That information can be retrieved from the Position Control-related diagrams (i.e., the left side of the Figure 9). Overall, both Attitude and Position control information illustrates the capability of our proposed curiosity-based method (i.e., HCM) while showing the lack of ability of other algorithms in terms of control and flying toward the desired position and attitude.

So far, our result showed this paper’s algorithm capability in controlling the quad-copter agent and flying toward the desired position and attitude (i.e., the first part of the objectives of the environment described in Section III-N). However, to illustrate the algorithm’s capabilities, some information regarding the obstacles is necessary (i.e., the second part of the environment objectives), which is mentioned in the following.

B. Goal and Obstacles Distances

As described in Section III-N, the algorithm should control a quad-copter agent and fly it toward a desired location while avoiding three obstacles. Figure 10 is comprised of two diagrams. The left diagram shows the distance between the quad-copter and the position of the goal, and the right image shows the distance between the quad-copter and obstacles in the environment. The obstacle distance is the mean of the three Euclidean distances between the quad-copter and each one of the obstacles in the environment. Both diagrams in Figure 10 show the capability of PPO+HCI in terms of decreasing the distance between the quad-copter and the location of the goal while slightly increasing and then maintaining the distance between the quad-copter and obstacles. These two diagrams illustrate that the quad-copter

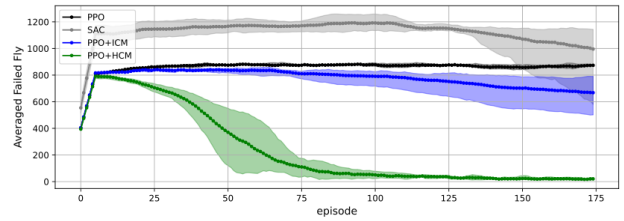
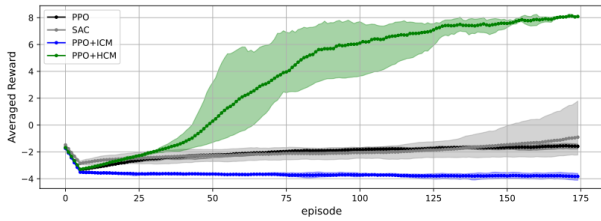


Fig. 8. Capability of each algorithm in maximizing the reward through time (left). Capability of each algorithm to reduce the Failed Fly (crashes) measure (right).

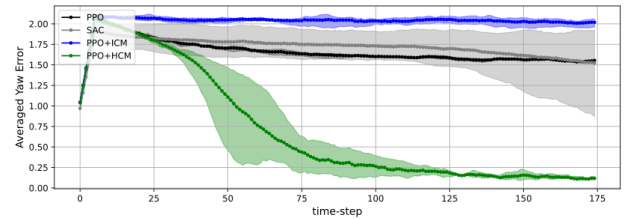
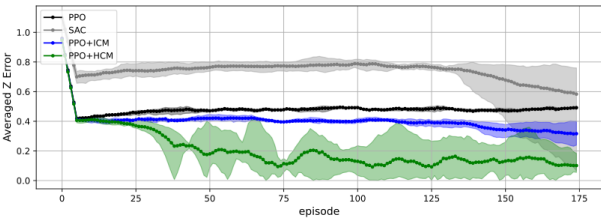
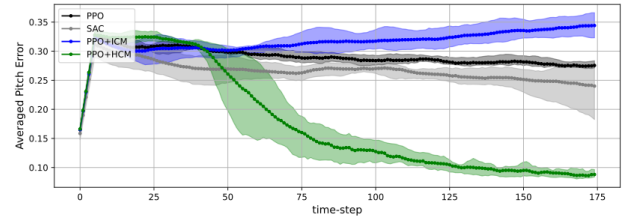
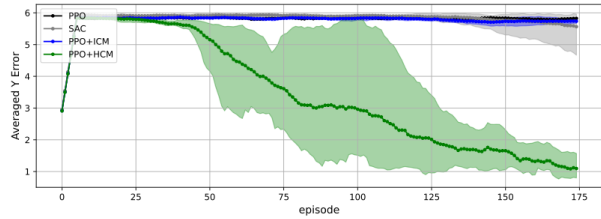
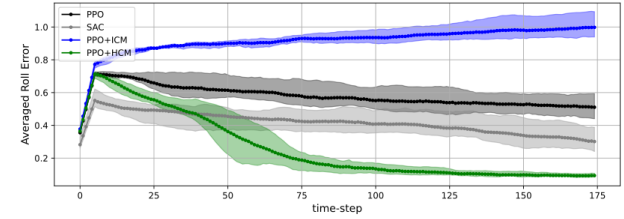
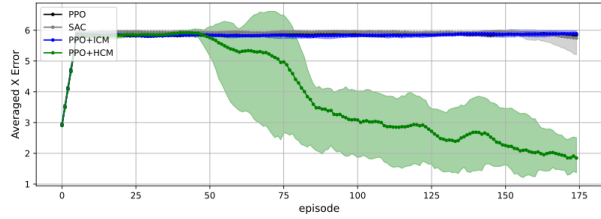


Fig. 9. Position and attitude errors: position errors for X, Y, and Z (left), and attitude error in Roll, Pitch, and Yaw (right).

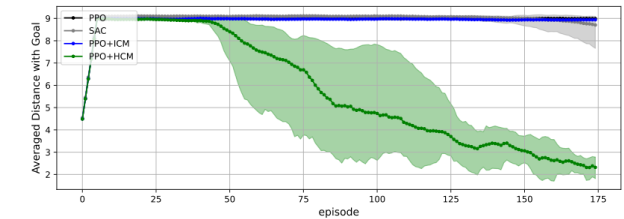
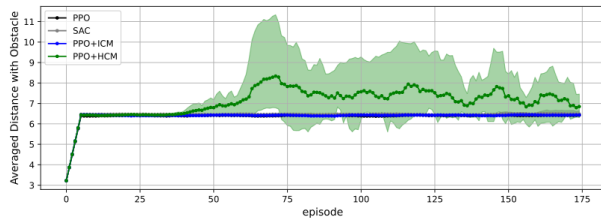


Fig. 10. Average distance between the quad-copter and the obstacles (left), and average distance between the quad-copter and the goal position (right).

controlled by PPO+HCP reaches the goal while avoiding the obstacles.

C. Curiosity Visualization

Considering the description in Section III-M regarding the visualization of the curiosity effect, Figure 11 and 12 show the effect of curiosity in evolving exploration patterns when the number of episodes increases. The figures are comprised of two rows; the first row shows the PPO exploration pattern, and the second row shows the PPO+HCM exploration pattern. For PPO, as can be seen in the figures, the pattern does

not evolve and is almost static where it does not move toward the center of the box (i.e., toward the desired position); this observation is also supported by the Position Control Information section of Figure 9, as the algorithm does not reduce the position error by time. For the PPO+HCM, on the other hand, as the illustrated in the second row of the figures, the pattern of exploration is changing. By increment of the number of episodes, the algorithm explores more trajectories that ends up towards the center of the box (i.e., desired position), and also, more concentration can be seen in that

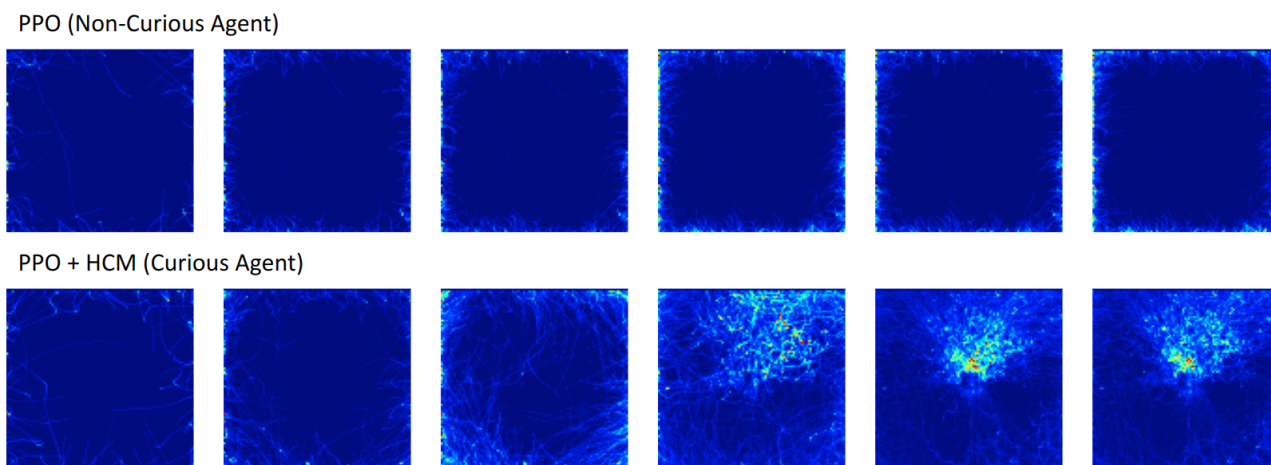


Fig. 11. Visualization of the effect of curiosity in the agent’s trajectories (in XY plane, i.e., top-view). Starting from left to right of the figure, by increasing the number of episodes, the curious agent would have more activities toward the center of the image or toward the desired position, but the non-curious agent avoids moving toward the center mainly because it avoids hitting the obstacles. This figure focuses only on the areas important for exploration.

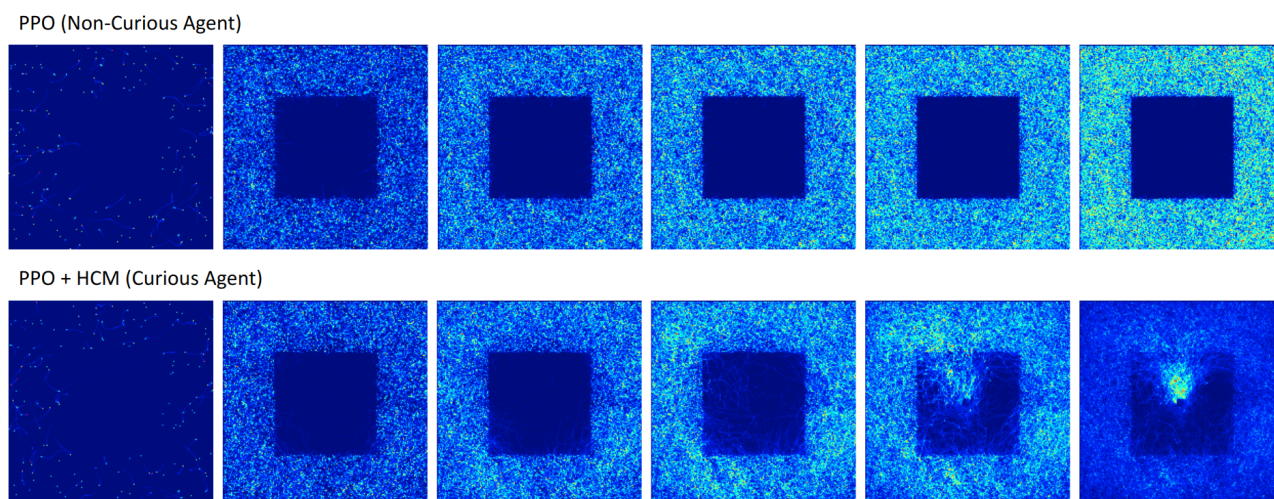


Fig. 12. Visualization of the effect of curiosity in the agent’s trajectories (in XY plane, i.e., top-view). Starting from left to right of the figure, by increasing the number of episodes, the curious agent would have more activities toward the center of the image or toward the desired position, but the non-curious agent avoids moving toward the center mainly because it avoids hitting the obstacles. This figure visualizes on all the areas traversed by the agent.

area.

D. Trajectories of Optimal Policy

Figure 13 shows two sample trajectories generated by an optimal policy trained using the curiosity-based algorithm proposed in this paper (i.e., PPO+HCM) when the quad-copter is initiated in random initial positions and attitudes and three obstacles are initiated in three random positions. The trajectories are visualized using the right-hand coordinate system where the red arrow shows the Yaw orientation of the quad-copter. As can be seen, the algorithm can control the quad-copter and pass the obstacles while controlling the vehicle’s Yaw toward the desired location.

V. DISCUSSION

In this section, we discuss some matters that need further explanation.

A. Algorithm Selection Strategy

We used the PPO as the baseline algorithm for the performance test because it is the algorithm that is mostly used for learning to fly such as in [1] [2] [3] and is a powerful on-policy algorithm. Moreover, we also selected SAC to show the result of a powerful off-policy RL algorithm that incorporates a memory replay. Our goal in this paper was to show that integrating curiosity with reinforcement learning-based flight controller makes solving complex low-level flight control problems possible. Thus, we selected PPO+ICM for testing the performance. Finally, we showed

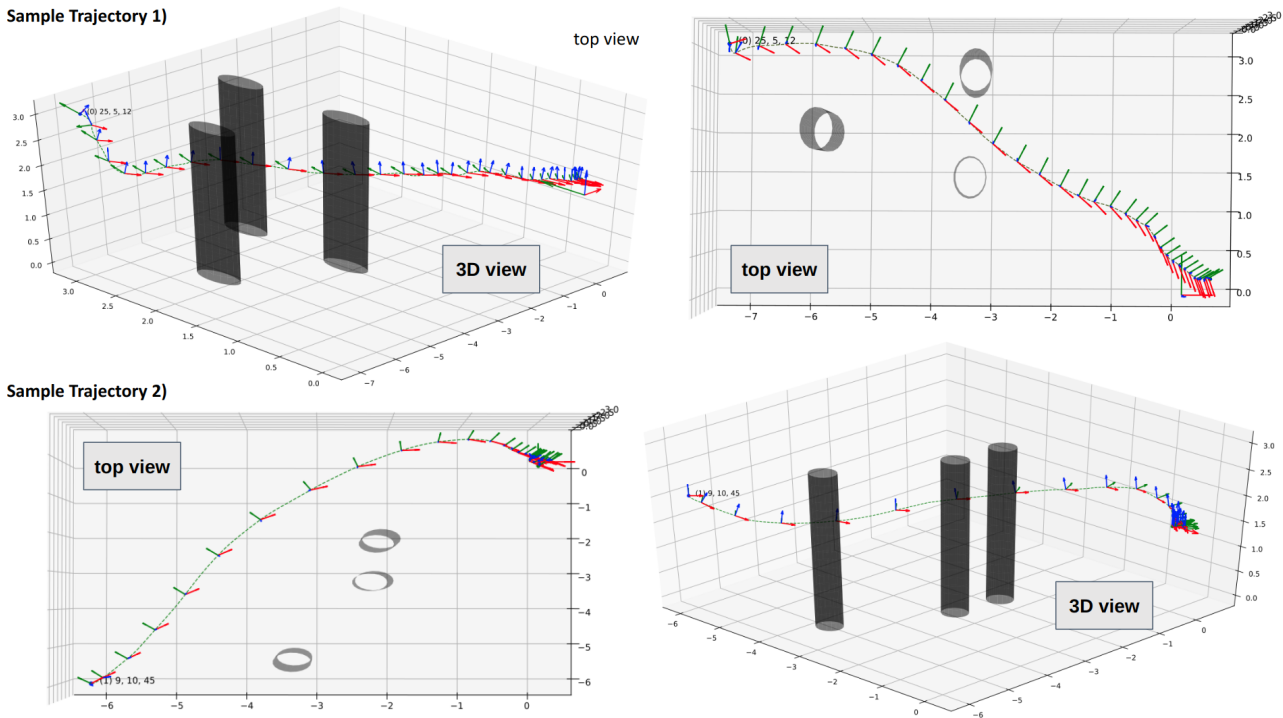


Fig. 13. Visualization of two sample trajectories of our optimal policy (the top view and 3D view of each trajectory are shown per row). The quad-copter is initialized in random position and attitude, and obstacles are initiated in random positions.

the capability of the proposed algorithm (i.e., PPO+HCM) compared with the competing ones and the importance of our contribution at the algorithm level.

B. The Gap between the Simulation and Real World

One area for improvement with the simulation-based learning algorithm is the gap between the simulation and the real world. One way to address this problem is by reducing the difference between the simulation and the real-world environments. However, this paper focuses on the learning part and the effect of adding curiosity to the learning policy for low-level flight control. For this reason, we did not enter the area of real-world tests in this paper, especially considering that it is already proven in [2] [3] [4] that a reinforcement learning-based policy can be used in a real-world quad-copter for low-level control.

C. Exploration, Exploitation, and Curiosity

It is a well-known capability of curiosity to increase and orient the exploration towards surprise, or in better words, to explore the environment meaningfully. While reinforcement learning algorithms, by default, use a trade-off between exploitation and random exploration (exploration mechanisms such as epsilon greedy in Q-learning or noise injected to the output action in methods such as DDPG, SAC, and PPO), they do purely random explorations. However, curiosity, as implemented in our work, generates an intrinsic reward that increases in the states unknown to the agent (by measuring the agent prediction error on those states) and decreases in the states that the agent visited frequently. Thus, curiosity

can be seen as a parameterized neural network architecture that rewards the agent more in surprising states, motivating the agent to explore those states more. As a result, curiosity aims to make the exploitation part of the RL more efficient while still using the default exploration mechanism.

D. Computational Time

Considering adding an extra head to PPO networks and having ten sub-modules in the proposed curiosity algorithm, the learning part is much heavier than regular RL-based low-level flight control learning. However, that is only for the learning time, which usually happens on a powerful machine. For the execution time or deployment on an actual quad-copter, our approach is similar to the mentioned approaches because only one network (i.e., Policy Network) is needed.

VI. CONCLUSION

In this work, we proposed a new approach for autonomous learning of low-level control policy. To achieve that, we proposed a new approach for implementing a computational model of curiosity motive using prediction error. To measure the capability of our algorithm, we designed and implemented a complex environment in Gazebo for learning quad-copter low-level flight control policy. In the designed environment, the algorithm should learn to directly control the quad-copter by generating the proper motor speed from odometry data. Further, the algorithm should learn to fly through obstacles while controlling the Yaw direction of the quad-copter toward the desired location. We ran tests

to measure and compare the proposed algorithm to other baseline algorithms.

As shown in Section IV of this paper, the proposed approach (i.e., PPO+HCM) can learn a flight policy when other algorithms fail to do so. By incorporating the proposed curiosity module, the algorithm can evolve the exploration pattern and fly to areas where other algorithms avoid flying. As a result, it can learn to control the quad-copter, control the Yaw direction of the quad-copter toward the desired location, and avoid hitting obstacles (as a low-level controller). In future works, we plan to measure the effect of incorporating machine imagination in a low-level framework.

ACKNOWLEDGMENT

This work is 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.

REFERENCES

- [1] A. Ramezani Dooraki and D.-J. Lee, “An innovative bio-inspired flight controller for quad-rotor drones: Quad-rotor drone learning to fly using reinforcement learning,” *Robotics and Autonomous Systems*, vol. 135, p. 103671, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S092188902030511X>
- [2] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, “Control of a quadrotor with reinforcement learning,” *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, oct 2017. [Online]. Available: <https://doi.org/10.1109/2Flra.2017.2720851>
- [3] A. Molchanov, T. Chen, W. Hönig, J. A. Preiss, N. Ayanian, and G. S. Sukhatme, “Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors,” *CoRR*, vol. abs/1903.04628, 2019. [Online]. Available: <http://arxiv.org/abs/1903.04628>
- [4] C.-H. Pi, K.-C. Hu, S. Cheng, and I.-C. Wu, “Low-level autonomous control and tracking of quadrotor using reinforcement learning,” *Control Engineering Practice*, vol. 95, p. 104222, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0967066119301923>
- [5] A. Ramezani Dooraki and D.-J. Lee, “A multi-objective reinforcement learning based controller for autonomous navigation in challenging environments,” *Machines*, vol. 10, no. 7, 2022. [Online]. Available: <https://www.mdpi.com/2075-1702/10/7/500>
- [6] P.-Y. Oudeyer and F. Kaplan, “What is intrinsic motivation? a typology of computational approaches,” *Frontiers in Neurobotics*, vol. 1, 2007. [Online]. Available: <https://www.frontiersin.org/articles/10.3389/neuro.12.006.2007>
- [7] J. Gottlieb, P.-Y. Oudeyer, M. Lopes, and A. Baranes, “Information-seeking, curiosity, and attention: computational and neural mechanisms,” *Trends in Cognitive Sciences*, vol. 17, no. 11, pp. 585–593, 2013. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1364661313002052>
- [8] R. Houthoofd, X. Chen, Y. Duan, J. Schulman, F. D. Turck, and P. Abbeel, “Curiosity-driven exploration in deep reinforcement learning via bayesian neural networks,” *CoRR*, vol. abs/1605.09674, 2016. [Online]. Available: <http://arxiv.org/abs/1605.09674>
- [9] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, “Curiosity-driven exploration by self-supervised prediction,” *CoRR*, vol. abs/1705.05363, 2017. [Online]. Available: <http://arxiv.org/abs/1705.05363>
- [10] G. Ostrovski, M. G. Bellemare, A. van den Oord, and R. Munos, “Count-based exploration with neural density models,” in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML’17. JMLR.org, 2017, p. 2721–2730.
- [11] S. H. Huang, M. Zambelli, J. Kay, M. F. Martins, Y. Tassa, P. M. Pilarski, and R. Hadsell, “Learning gentle object manipulation with curiosity-driven deep reinforcement learning,” *CoRR*, vol. abs/1903.08542, 2019. [Online]. Available: <http://arxiv.org/abs/1903.08542>
- [12] M. Frank, J. Leitner, M. Stollenga, A. Förster, and J. Schmidhuber, “Curiosity driven reinforcement learning for motion planning on humanoids,” *Frontiers in Neurobotics*, vol. 7, 2014. [Online]. Available: <https://www.frontiersin.org/articles/10.3389/fnbot.2013.00025>
- [13] T. K. Kaiser and H. Hamann, “Innate motivation for robot swarms by minimizing surprise: From simple simulations to real-world experiments,” *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3582–3601, 2022.
- [14] K. Cai, W. Chen, C. Wang, H. Zhang, and M. Q.-H. Meng, “Curiosity-based robot navigation under uncertainty in crowded environments,” *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 800–807, 2023.
- [15] Q. Sun, J. Fang, W. X. Zheng, and Y. Tang, “Aggressive quadrotor flight using curiosity-driven reinforcement learning,” 2022.
- [16] “Markov decision process,” https://en.wikipedia.org/wiki/Markov_decision_process, accessed: 2023-03-16.
- [17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: <http://arxiv.org/abs/1707.06347>
- [18] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, “Trust region policy optimization,” *CoRR*, vol. abs/1502.05477, 2015. [Online]. Available: <http://arxiv.org/abs/1502.05477>
- [19] S. Kakade and J. Langford, “Approximately optimal approximate reinforcement learning,” in *Proceedings of the Nineteenth International Conference on Machine Learning*, ser. ICML ’02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 267–274. [Online]. Available: <http://dl.acm.org/citation.cfm?id=645531.656005>
- [20] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1928–1937. [Online]. Available: <http://proceedings.mlr.press/v48/mnih16.html>
- [21] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, “High-dimensional continuous control using generalized advantage estimation,” 2015.
- [22] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and scalable predictive uncertainty estimation using deep ensembles,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS’17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6405–6416.
- [23] “Gazebo simulator,” <http://gazebo.org/>, accessed: 2023-03-16.
- [24] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Cham: Springer International Publishing, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625.
- [25] “Robot operating system,” <https://www.ros.org/>, accessed: 2023-03-16.

C

2023 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 17–20, 2023, ROME, ITALY

DEEP REINFORCEMENT LEARNING WITH ACTION MASKING FOR DIFFERENTIAL-DRIVE ROBOT NAVIGATION USING LOW-COST SENSORS

Konstantinos Tsampazis, Manos Kirtas, Pavlos Tosidis, Nikolaos Passalis, Anastasios Tefas

Computational Intelligence and Deep Learning (CIDL) Group, AIIA Lab.,
Department of Informatics,
Aristotle University of Thessaloniki, Thessaloniki, Greece
E-mails: {tsampaka, eakirtas, ptosidis, passalis, tefas}@csd.auth.gr

ABSTRACT

Driving a wheeled differential-drive robot to a target can be a complicated matter when trying to also avoid obstacles. Usually, such robots employ a variety of sensors, such as LiDAR, depth cameras, and others, that can be quite expensive. To this end, in this paper, we focus on a simple differential-drive wheeled robot that uses only inexpensive ultrasonic distance sensors and touch sensors. We propose a method for training a Reinforcement Learning (RL) agent to perform robot navigation to a target while avoiding obstacles. In order to increase the efficiency of the proposed approach we design appropriate action masks that can significantly increase the learning speed and effectiveness of the learned policy. As we experimentally demonstrated, the proposed agent can robustly navigate to a given target even in unknown procedurally generated environments, or even when denying part of its sensor input. Finally, we show a practical use-case using object detection to dynamically search for, and move to objects within unknown environments. The code used for conducted experiments is available online on Github.

Index Terms— Robot Navigation, Low-Cost Robot Sensors, Deep Reinforcement Learning, Action Masking

1. INTRODUCTION

Autonomous navigation of mobile robots has been a popular research topic in the field of robotics for decades. The ability to navigate in complex and dynamic environments is essential for many applications, such as search and rescue, logistics, and home care. One of the most common and versatile types of mobile robots is the differential-drive wheeled robot. These robots are easy to build and control, and their differential-drive system allows them to turn on the spot and move in any direction. At the same time, Deep Reinforcement Learning (DRL) [1] has also emerged as a promising technique for training autonomous agents to perform complex tasks, leading to several robotics applications [2], including navigation, manipulation, and control.

Recent research mainly targets robots that use multiple sensors and relatively expensive configurations with LiDAR, depth cameras and others. One such example is in [3], where the authors used a Jetson Nano for obstacle avoidance using a monocular camera. In another work [4], the authors used the Turtlebot 3 Waffle Pi, and trained a Double DQN [5] agent to navigate to a target. In a more recent work using the same robot [6], a mapless local path planning approach was presented, that used variants of Deep Q-Network [7] to increase success rates, proposing the n-Step Dueling Double DQN with Reward-Based ϵ -Greedy (RND3QN) algorithm. In [8], the authors successfully used Deep Deterministic Policy Gradient [9] to train an agent to drive a differential-drive robot to a target, improving on this paper's authors' earlier work in [10]. They used curriculum learning [11] to gradually train the agent on a small set of increasingly difficult maps. Another more generic example is the established Robot Operating System (ROS)¹ navigation stack, which while it can work with inexpensive ultrasonic distance sensors, it is more well-suited to work with LiDAR and depth sensors. As a result, many DRL stacks are designed exclusively for high-end robotic hardware, which limits their potential impact in a wide range of applications, from education to low-cost mass production robots. At the same time, training DRL agents with low-fidelity and noisy sensors can worsen sample efficiency. This necessitates the use of more sample-efficient paradigms in such applications.

In this paper, we propose a method for training an agent to drive a low-cost differential-drive wheeled robot navigating to a target while avoiding obstacles, using the well-established Proximal Policy Optimization (PPO) [12] RL algorithm. To improve training efficiency we employ invalid action masking [13], also known as Maskable PPO, after appropriately designing two masks that can lead to increased performance. This work a) introduces a more systematic and effective approach to perform action masking, as well as b) paves the way for introducing a state-of-the-art DRL approach using

¹<https://www.ros.org/>

low-cost sensors in the robotic navigation domain. The agent used, apart from its sensor values, takes only the relative angle and distance to its target and not its own or the target’s absolute position, and thus can act as a local path planner and navigate dynamically in unknown environments. We developed a randomized procedural map generation method within the Webots robotics simulator [14], to be able to train and realistically evaluate the agent in complex environments with obstacles of various challenging shapes, using realistic noisy sensors. The code used for conducted experiments is available online².

The rest of the paper is structured as follows. The proposed method is provided in Section 2, while the experimental evaluation is provided in Section 3. Finally, Section 4 concludes the paper.

2. PROPOSED METHOD

2.1. Background and Setup

In this work, we employ a custom robot, as shown in Fig. 1a. This robot is created in Webots and consists of two motors connected to wheels providing differential drive, a forward-placed bumper that is split between two touch sensors, one left and one right, and 13 forward-facing ultrasonic distance sensors that are placed in equal-spaced angles between $[-\pi, \pi]$. The ultrasonic distance sensors have a range of $1m$ and return valid values when their ray angle of incidence to obstacles is close to vertical. Moreover, the values returned are noisy, simulating real ultrasonic distance sensors.

We used a discrete action space with a set of five actions that cumulatively control the motor speeds of the robot. The first one increases both motor speeds by a fixed amount up to a limit, the second decreases them, the third and fourth increase one motor speed but decrease the other and the fifth action does not cause any changes to the motor speeds. We refer to these actions as “forward”, “backward”, “left”, “right” and “no action”.

The observation space of the agent is primarily described by the following vectors:

$$\mathbf{a}_t = [d_t, a_t, m_{l,t}, m_{r,t}, ts_{l,t}, ts_{r,t}], \quad (1)$$

where d_t is the current Euclidean distance to the target, a_t is the current angle to the target in regards to the facing angle of the robot, $m_{l,t}$ and $m_{r,t}$ are the current left and right motor speeds, and finally $ts_{l,t}$ and $ts_{r,t}$ are the left and right touch sensor values, for timestep t and

$$\mathbf{b}_t = [ac_{0,t}, \dots, ac_{4,t}, ds_{0,t}, \dots, ds_{12,t}], \quad (2)$$

where $ac_{i,t}$ for $i \in [0, 4]$ is the one-hot vector representing the previous action, and $ds_{j,t}$ for $j \in [0, 12]$ represents the latest distance sensors values.

²https://github.com/aidudezzz/deepworlds/tree/dev/examples/find_and_avoid_v2

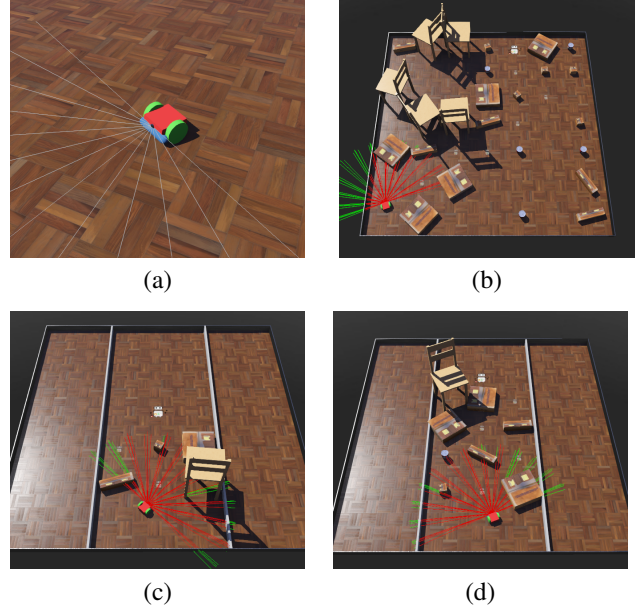


Fig. 1. (a) the robot used, (b) random map with all 25 obstacles, (c) corridor map with 2 rows of obstacles, (d) corridor map with 4 rows of obstacles.

To construct the full observation \mathbf{o}_t at each timestep t , we concatenate \mathbf{a}_t and \mathbf{b}_t with \mathbf{a}_{t-w} and \mathbf{b}_{t-w} , where \mathbf{a}_{t-w} and \mathbf{b}_{t-w} are the vectors containing the values of timestep $t-w$ approximately one second before timestep t . Therefore, the full observation is defined as:

$$\mathbf{o}_t = [d_t, a_t, \dots, ds_{12,t}, d_{t-w}, a_{t-w}, \dots, ds_{12,t-w}], \quad (3)$$

where $w = \lceil \frac{1000}{s} \rceil$ and s is a single timestep time defined in milliseconds. This way, even though the agent uses a simple feed forward neural network, it takes a time window as observation giving it insight into how the observation values are changing over time, which experimentally provided the best results. All values are normalized appropriately in the $[-1, 1]$ range.

The reward R is defined as $R = w_{dr}dr + w_{ar}ar + w_{dsr}dsr + w_{rtr}rtr + w_{cr}cr$, where dr is the distance to target reward, ar is the angle to target reward, dsr is the distance sensors reward, rtr is the reach target reward and cr is the collision reward, who all lie within or are normalized in the range $[-1, 1]$. Then every sub-reward is multiplied by their corresponding weights (experimentally) set as $w_{dr} = 1.0$, $w_{ar} = 1.0$, $w_{dsr} = 10.0$, $w_{rtr} = 1000.0$, $w_{cr} = 100.0$. The distance to target reward is comprised of two components itself, one continuous that penalizes the agent the farther away it is from the target, and one discrete that rewards the agent every time it achieves a new minimum distance to the target. The agent is rewarded when facing the target or when it turns towards the target, and penalized when it turns away from the target. The angle reward is zeroed out when there are obstacles detected nearby, to enable the agent to turn away from

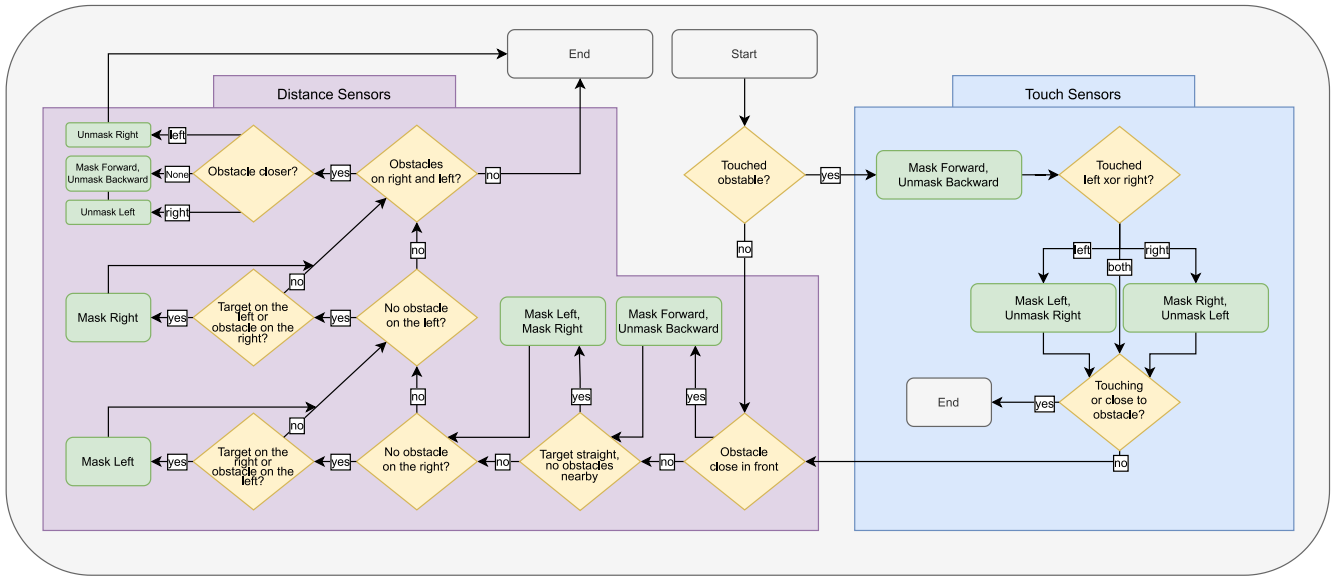


Fig. 2. The proposed mask flowchart.

the target to avoid them. The distance sensors reward penalizes the agent when the sensors are reading below a threshold value and rewards it otherwise. When the agent reaches the target it is rewarded, but the reward is scaled based on the episode time elapsed, i.e. when the agent instantaneously reaches the target the reward is 1.0 and when it reaches the target just before the episode ends it is rewarded with 0.5. Finally, the agent is penalized when it collides with obstacles as detected by its touch sensors. An episode is terminated when the agent either reaches the target or it collides with obstacles for 4, 096 steps.

2.2. Proposed Action Masking

Invalid action masking [13] is a technique used with PPO to handle tasks involving invalid or forbidden actions. It prevents the agent from selecting invalid actions during training and evaluation by setting their probability to zero. This leads to more efficient policies, especially in tasks with common invalid actions or severe consequences. It can be applied to both large and small action spaces. The technique has been most notably, successfully used in Dota 2 [1].

In this work we propose two different masking approaches, a “simple” baseline action mask and a more sophisticated “advanced” mask. Typically, the masks provide a vector of truth values, one for each action, that are either true, enabling or unmasking the action, or are false, disabling or masking the action. These vectors are calculated for every simulation step.

The *simple mask* developed initially, masks the forward action when close to obstacles or masks the backward action when no obstacles are detected. This prevents unnecessary

collisions and unwarranted backwards driving. The mask also prevents turning into obstacles by masking left or right actions when respective sensors detect low values.

The *advanced mask* flowchart is provided in Fig. 2. This method runs in every simulation step starting with all actions unmasked and returns the final mask for the next step. As can be seen in the flowchart, the mask method is split into two components, one using the distance sensors and one using the touch sensors. As long as nothing is detected with the touch sensors, the robot’s actions are masked via the relative current target angle and distance sensors. When the distance sensors component fails and a collision is detected, the touch sensors component takes over until the robot is clear of obstacles. Thus, the robust touch sensors act as a fallback for the ultrasonic distance sensors which are quite unreliable, as the incidence angle of their rays to the obstacles must be close to vertical to return a valid value. However, obstacles that can be detected via distance sensors can be effectively avoided.

3. EXPERIMENTAL EVALUATION

3.1. Training

For all the conducted experiments we used a simple feed forward neural network that has three hidden layers with [1024, 512, 256] neurons for the actor and [2048, 1024, 512] for the critic. To this end, we utilized the RL implementation provided by stable-baselines³. All other network architecture values and training parameters are left at their default values apart from γ set to 0.999, entropy coefficient set to

³<https://github.com/DLR-RM/stable-baselines3>

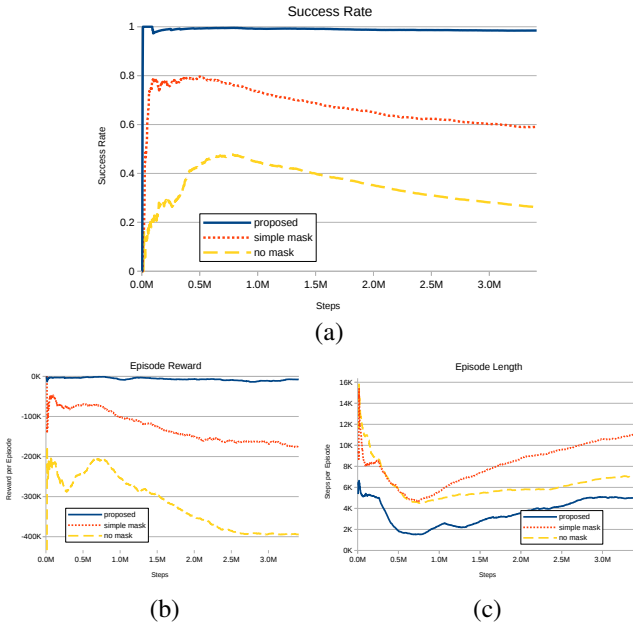


Fig. 3. The training results for the three masks used over training steps: (a) the average success percentage, (b) the average episode reward, (c) the average episode length. Each masking method was trained for 6 different seeds and the average results are presented.

0.001 and base learning rate set to $3e-4$ and decaying with a linear schedule.

We leveraged the open-source robotics simulator Webots [14] to create a randomizable environment for training and evaluating the agents. The stable-baselines3 agent implementation was integrated with the simulator through the deepbots [10] framework, which enables the creation of custom OpenAI gym-compliant⁴ RL environments in Webots. The training and evaluation maps consist of walled arenas with various shapes of obstacles, known as “random maps”. The number of obstacles can be customized, and the robot and target are placed randomly in free spaces with varying distances between them. An example map can be seen in Fig. 1b. Moreover, the map can be randomly generated as a corridor with adjustable number of rows of obstacles between the robot and the target, referred to as “corridor map”, as can be seen in Fig. 1c, d.

The agents undergo a total of 3,407,872 training steps, with each new map serving as an episode lasting 16,384 steps. We employed curriculum learning [11] during training, gradually increasing the difficulty. Initially, for 262,144 steps, we used the “random map” configuration with 10 random obstacles. The target was placed at a Manhattan distance of 10 from the robot. This setup provided open maps with few obstacles and wide paths, making it the easiest starting point for the agents. Subsequently, we introduced a series

of “corridor maps” of increasing difficulty, with the agents trained for 524,288 steps on each difficulty level. They start out with 1 row of 2 obstacles and end up with 5 rows of 10 obstacles between the target and the robot. Fig. 1d illustrates the challenging nature of corridor maps, which feature dead ends and intricate paths to navigate. Moreover, even in its easiest configuration the “corridor map” puts obstacles between the robot and the target, forcing the agents to learn to traverse around the obstacles. Finally, the agents are trained for another 1,048,576 steps on a “random map” with all 25 available obstacles and the target Manhattan distance set to a random value between 10 and 12. This configuration provides the agents with a more realistic environment with many obstacles spread around.

The training results can be seen in Fig. 3. The average success percentage over the timesteps seen in Fig. 3a is the ratio of episodes that the robot reached the target over the total number of episodes. The agent without any mask only ever achieves $\approx 47.0\%$ success percentage before decaying to under $\approx 30\%$ at the end of its training with its average reward collapsing. The episodes terminate early due to collisions, thus achieving a lower average episode length than the simple-mask agent. The agent that uses the simple mask has much better performance, as it begins with a success percentage of $\approx 80\%$ and drops to just under 60% before the training ends. The proposed method with the advanced mask quickly achieves a success percentage of close to 100% in the initial easy maps and decays over time to 98.5%, as the curriculum gets harder. As expected, all methods have sharp drops in their initial average episode length as the agents get better at reaching the target efficiently, and increase over time as the maps get more and more complicated.

3.2. Evaluation

	Baseline (no mask)	Proposed (simple mask)	Proposed (advanced mask)
Success (%)	43.7 ± 20.9	64.1 ± 23.1	98.8 ± 0.2
Reward ($\times 10^2$)	-14.6 ± 6.3	-2.1 ± 3.0	8.8 ± 0.1
Ep. Len. ($\times 10^3$)	8.6 ± 10.8	10.8 ± 8.6	2.5 ± 1.9

Table 1. The evaluation results for the different masks used, averaged across 6 runs with different seeds and their corresponding standard deviation.

To evaluate the trained agents in the various mask configurations, we used a set of 600 previously unseen maps of 6 difficulty setups with 100 maps each, using the same seed to get exactly the same robot/target and obstacles randomization for all agents to provide a fair evaluation and comparison. Starting out, the first 100 maps are simple “corridor maps” without any obstacles, that show whether the agent can efficiently move straight to the target that is placed in various

⁴<https://github.com/openai/gym>

distances along the corridor. Earlier in development, unsuccessful agents had trouble reaching the target even in empty corridors and would overshoot or hug the walls. The next 4 sets of 100 maps each are “corridor maps” of increasing difficulty in terms of how many rows of obstacles are placed between the robot and the target, similar to the training setup described previously. The last set of 100 maps uses the “random map” configuration that includes all 25 available obstacles. Note that the seed used for evaluation is different from the training seeds, thus the maps used in evaluation are not previously seen by the agents.

The reward function weights were set as $w_{dr} = 0.0$, $w_{ar} = 0.0$, $w_{dsr} = 0.0$, $w_{rtr} = 1000.0$, $w_{cr} = 1.0$ to provide a better constrained reward score, where the agent gets rewarded only by reaching the target or punished for colliding. The average evaluation results can be seen in Table 1, along with their standard deviation across the 6 seeds. Similar to the training results, the no mask setup ends episodes prematurely due to collisions without reaching the target, which is reflected in the poor success rate and more decisively in the low average episode reward, and as a consequence gets a lower average episode length than the simple mask. The proposed method converges to a near optimal policy that manages to solve $\approx 98.8\%$ of the evaluation maps with a high average reward of ≈ 880 and low average episode length of ≈ 2500 steps, outperforming the other two methods by far. Finally, the proposed advanced masking method shows much better consistency represented by the very low standard deviation values across the seeds.

3.3. Component ablation and sensor denial

	No DS	50% DS	No TS	Only mask	Full
Success (%)	93.8	95.5	84.5	69.3	98.6
Reward ($\times 10^2$)	4.9	7.4	8.0	-6.8	8.7
Length ($\times 10^3$)	2.2	3.4	8.0	15.2	2.3

Table 2. The results of the various component ablation and sensor denial experiments. “DS” refers to distance sensors and “TS” refers to touch sensors.

We evaluated one of the trained agents with the advanced mask in the evaluation maps using the same seed and procedure, but in three different sensor configurations: disabling all distance sensors, randomly disabling 50% of the distance sensors in each episode, and removing touch sensors entirely setting their values to zero. Removing the distance sensors effectively disables the part of the proposed mask depends on them, and similarly, removing the touch sensors disables the touch sensor part of the mask. Disabling sensors also means that the agent has to work with an incomplete observation. Moreover, we evaluated a random policy that practically only uses the advanced mask to navigate, picking ran-

dom unmasked actions with equal probability. All the results can be seen in Table 2, where the four ablation configurations are compared to the full method. Denying half or all of the distance sensors decreases performance slightly in terms of success percentage. Without any distance sensors the agent moves blindly colliding with obstacles a lot, thus decreasing its average reward, but also less cautiously making it reach the target faster. The agent without touch sensors, is hampered in its ability to reach the target, resulting in a decrease of approximately $\approx 14\%$ in success percentage and prolonged navigation times due to the limitations of realistic ultrasonic distance sensors in detecting obstacles effectively. The high reward is a byproduct of the touch sensors not detecting any collisions. Using the mask alone, that includes the human expert knowledge, coupled with a random policy yields the worse results in all of the metrics, demonstrating that the learned behaviour of the agent via the reward function is a crucial component. These results show that the method is resilient to distance sensor denial, and has an adequate success percentage even without the touch sensors. On a real robot the distance sensors used in simulation can be emulated by a single distance sensor rotating on a servo motor in predetermined positions, which will affect the proposed method’s performance marginally due to its resiliency to the distance sensor denial shown.

3.4. Example Use-Case

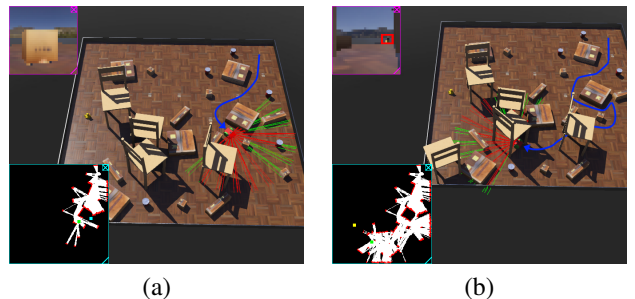


Fig. 4. Screenshots of the robot (green box on the map) looking for the rubber duck in a random map: (a) robot is moving towards a random exploration target (light blue box), (b) robot has acquired the target (yellow box) and moving towards it. The path the robot has followed is roughly sketched in blue. On top-left the perspective of the camera can be seen, as well as the bounding box of the detected object on screenshot (b). The current map can be seen on the bottom-left.

We used the trained agent with the advanced mask on a more practical use-case, combined with object detection provided by the simulator and a simple custom mapping functionality, where the agent searches the map for a rubber duck as can be seen in Fig. 4. The mapping functionality relies only on the distance sensors, starting out with a black map image

that, as the robot moves around slowly gets filled with white pixels for empty space and red pixels for obstacles detected by the distance sensors, as can be seen on the lower-left parts of the screenshots on Fig.4. The mapping functionality provides exploration targets to the agent that lie in black patches, i.e. unexplored parts of the current map. As the agent explores the environment with the help of these targets, it can detect the rubber duck via the camera's object detection module. When this happens the agent is provided with a target that corresponds to the rubber duck's position as approximated by the bounding box on the camera image. Consequently, using the trained agent with the proposed method as a local path planner and a rudimentary exploration and mapping functionality, the robot can find targets within the environment and navigate successfully and efficiently to them, using only simple distance and touch sensors as well as a camera for target object acquisition.

4. CONCLUSIONS

In this paper, we demonstrated that by crafting an appropriate masking function we can effectively combine RL with expert human knowledge to provide an efficient policy for differential-drive robot navigation without using expensive robot configurations and sensors. We trained the three methods on multiple seeds and evaluated the trained agents on the same random maps to provide a fair comparison, while also conducting sensor ablation and denial experiments. The experimental results suggest that the proposed masking method yields significantly better results than the baseline and converges to a near-perfect policy. The proposed method was also able to successfully navigate to the target in nearly all the evaluation maps, even though the random map generation method used produces very complicated maps with a variety of obstacles both in terms of size and shape, that the realistic noisy ultrasonic distance sensors have a great difficulty detecting. The nature of the proposed method makes it suitable to be used as a local path planning algorithm of low cost once trained, which is shown via an example use-case that uses a rudimentary custom mapping functionality and object detection via a camera. In the few failure cases, the agent gets stuck in narrow passages between obstacles it cannot detect or gets trapped in long dead-end paths, that are expected shortcomings of the nature of the method and sensor setup. Future work includes incorporating RGB camera input, as well as adding end-to-end mapping capabilities to the agent to address these challenges.

Acknowledgments 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.

5. REFERENCES

- [1] Christopher Berner et al., "Dota 2 with large scale deep reinforcement learning," *CoRR*, vol. abs/1912.06680, 2019.
- [2] Nikolaos Passalis et al., "OpenDR: An open toolkit for enabling high performance, low footprint deep learning for robotics," in *Proc. IEEE/RSJ Intl. Conf. Intelligent Robots and Systems*, 2022.
- [3] Thai-Viet Dang and Ngoc-Tam Bui, "Obstacle avoidance strategy for mobile robot based on monocular camera," *Electronics*, vol. 12, no. 8, 2023.
- [4] Hamza Aydemir, Mehmet Gök, and Mehmet Tekerek, "Reinforcement learning based local path planning for mobile robot," in *Interdisciplinary Conf. Mechanics, Computers and Electrics*, 11 2021.
- [5] Hado van Hasselt et al., "Deep reinforcement learning with double q-learning," *CoRR*, vol. abs/1509.06461, 2015.
- [6] Yan Yin, Zhiyu Chen, Gang Liu, and Jianwei Guo, "A mapless local path planning approach using deep reinforcement learning framework," *Sensors*, vol. 23, no. 4, 2023.
- [7] Volodymyr Mnih et al., "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.
- [8] Ali Salimi Sadr et al., "An efficient planning method for autonomous navigation of a wheeled-robot based on deep reinforcement learning," in *12th Intl. Conf. Computer and Knowledge Engineering*, 2022, pp. 136–141.
- [9] Timothy P. Lillicrap et al., "Continuous control with deep reinforcement learning," *arXiv 1509.02971*, 2015.
- [10] M. Kirtas et al., "Deepbots: A webots-based deep reinforcement learning framework for robotics," in *Artificial Intelligence Applications and Innovations*, Cham, 2020, pp. 64–75, Springer Intl. Publishing.
- [11] Sanmit Narvekar et al., "Curriculum learning for reinforcement learning domains: A framework and survey," *arXiv 2003.04960*, 2020.
- [12] John Schulman et al., "Proximal policy optimization algorithms," *arXiv 1707.06347*, 2017.
- [13] Shengyi Huang and Santiago Ontañón, "A closer look at invalid action masking in policy gradient algorithms," *The Intl. FLAIRS Conf. Proceedings*, vol. 35, may 2022.
- [14] Olivier Michel, "Webotstm: Professional mobile robot simulation," *Int. Journal of Advanced Robotic Systems*, vol. 1, 03 2004.

D

Data efficient Deep Reinforcement Learning for Robust Inertial-based UAV Localization

Dimitrios Tsiakmakis^a, Nikolaos Passalis^a, Anastasios Tefas^a

^a*Computational Intelligence and Deep Learning (CIDL) Group, Artificial Intelligence and Information Analysis (AIIA) Lab., Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, 541 24, Greece*

Abstract

Precise localization is a critical task for many UAV-based applications. Inertial Measurement Units (IMUs), which measure acceleration and angular velocity, are commonly used for UAV localization due to their low cost and small size. However, IMU-based localization is prone to accumulating errors over time, which can significantly impact the accuracy of the localization. To address this issue, we propose a data efficient Deep Reinforcement Learning (DRL) method that enables learning how to correct localization errors from IMUs. Our approach utilizes a novel data augmentation method, along with an appropriate “hint” loss that can provide additional supervision during the training process. As a result, the proposed method requires a very small number of real-world examples and can be implemented using widely available low cost RGB sensors, ensuring that it can be readily applied in a wide range of different applications. We demonstrate the effectiveness of the proposed method in both simulation and real-world UAV experiments. In comparison to traditional supervised and DRL approaches, the proposed approach allows for achieving more precise localization with fewer real-world examples, making it a practical tool for adapting DL-based localization models for UAV applications.

Keywords: Deep Reinforcement Learning, Inertial-based Localization, Data Augmentation

1. Introduction

Unmanned Aerial Vehicles (UAVs), also known as drones, are increasingly being used for a wide range of applications, including precision agricul-

ture (Radoglou-Grammatikis et al., 2020), search and rescue operations (Alotaibi et al., 2019), and indoor surveillance (Boonyathanmig et al., 2021). The ability to accurately determine the location of a UAV is critical for the success of these applications, as well as the majority of other UAV-based applications, since localization of UAVs is crucial for both mission control and safety reasons, i.e., avoiding flying over prohibited regions. Various techniques to UAV localization have been developed, with each depending on various sensors and delivering a different level of accuracy.

One of the most common methods for localizing a UAV is through the use of satellite-based radio-navigation systems, such as the Global Positioning System (GPS) (Sukkarieh et al., 1999; Han and Wang, 2012). These systems are relatively inexpensive, but they can be prone to inaccuracies, which can hinder the performance of UAV applications that require precise localization. Despite these limitations, GPS and similar technologies are widely used due to their convenience and ease of implementation. According to the official GPS documentation, GPS-enabled devices are typically precise to within 4.9 meters (16 feet), which is insufficient for a wide range of tasks. In addition, there are a variety of locations without GPS coverage (Vetrella and Fasano, 2016), and these methods cannot be deployed indoors due to the same cause. One way to minimize the inaccuracies of satellite-based radio-navigation systems is through the use of real-time kinematic (RTK) positioning (Henkel et al., 2016), which involves the use of additional base stations to provide more precise location information. While RTK positioning can improve the accuracy of UAV localization, it can also increase the cost and decrease the adaptability of UAVs due to the need for additional base stations. Another option for UAV localization is the use of light detection and ranging (LIDAR) systems (Jaboyedoff et al., 2012; Xu and Zhang, 2021), which can provide high-precision localization when used in combination with simultaneous localization and mapping (SLAM) techniques (Liu et al., 2022a; Giubilato et al., 2022). However, LIDAR systems often require expensive sensors and have higher computational and energy requirements compared to other localization methods (Gyagenda et al., 2022). Other techniques have been proposed by taking advantage of radiation fields (Newaz et al., 2016) or computationally intensive monocular vision (Xiao et al., 2019).

On the other hand, Inertial Measurement Units (IMUs) consist of a combination of accelerometers, gyroscopes, and magnetometers (Ahmad et al., 2013), and they can provide low-cost solutions for UAV localization that do not require external hardware or connectivity, such as satellites or base

stations. An Inertial Navigation System (INS) uses IMU data for dead reckoning to achieve localization (Barshan and Durrant-Whyte, 1995). There have also been investigations into using visual simultaneous localization and mapping (SLAM) with mono (Jin et al., 2022; He et al., 2022) and stereo cameras (Koestler et al., 2022) for UAV localization, but these methods tend to have higher computational costs. INS are often implemented using Micro Electro-Mechanical Systems (MEMS), which is a type of sensor technology that has become popular due to their compact size, low cost, and ability to be easily integrated into a wide range of robots, including UAV applications. These sensors provide real-time measurements and are capable of measuring linear acceleration and rotational velocity with a small margin of error. However, over time, the errors in these measurements can accumulate and result in significant position drift, which can compromise the accuracy of the sensor in mission-critical applications where it is being used as the sole localization sensor.

These restrictions have stimulated research into methods for improving UAV inertial-based navigation (Brossard et al., 2019; Cortés et al., 2018; Herath et al., 2020). Recent methods often use deep learning (DL)-based models that considerably enhance the localization process. Nevertheless, despite these enhancements, these methods have a significant drawback. These approaches use supervised learning, either regression-based or classification-based. As a result, they require a large number of annotated samples that need to be collected in order to train the corresponding models, which significantly increases the cost of data acquisition. In some cases, this might not even be possible since it might require increasing the payload of the drone, e.g., by adding additional sensors. Also, these approaches typically depend on the hardware used to collect the data, while when they are used on different hardware their performance can deteriorate. This means that new data needs to be collected and the models need to be retrained. Moreover, even when employing equivalent hardware, manufacturing tolerances may result in sensors with varying noise characteristics, making the use of supervised learning approaches challenging.

On the other hand, Deep Reinforcement Learning (DRL) could potentially allow for overcoming these limitations by enabling the IMU correction model to directly adapt to the underlying hardware and operate on more easily obtainable reward signals instead of accurate ground truth annotations. This signal can be sparse, i.e., provided only at the end of a training episode, and can be noisy, e.g., low cost RGB cameras can be used to generate a

reward based on fixed landmarks, which significantly reduces the annotation cost compared to supervised learning and improves the flexibility of DRL approaches. However, an important challenge is that DRL algorithms typically require a large number of episodes or iterations to learn effectively, which can be impractical for UAVs due to their limited onboard computing power and energy constraints. While it is possible to bypass the first limitation by using visual cues to generate a reward signal, the low data efficiency of DRL techniques remains a significant barrier to their practical use in UAV navigation.

In this work we propose a pipeline that can enable the use of DRL on UAVs for inertial-based navigation. The proposed method addresses the aforementioned challenges, including the need for a feedback signal to evaluate the learned policy and the high number of episodes typically required for learning. To achieve this, we introduce a two-stage pipeline. In the first step we propose training a generic DL backbone network that learns the dynamics of IMUs without focusing on a single sensor. This enables the more efficient application of transfer learning using DRL. Indeed, in the second step, DRL is applied by directly using a UAV to adapt the aforementioned backbone to correct the errors introduced by the onboard IMU. In order to ensure that the proposed method can be applied in practice, we propose two application-specific DRL improvements towards significantly improving the data efficiency of the learning process. First, we employ a data augmentation approach that generates numerous simulated episode trajectories from a single actual episode. Second, we propose an additional loss function that provides extra feedback when fine-tuning the learned policy depending on the sign of the observed reward signal. Finally, for obtaining the reward signal we employ a simple, yet efficient visual landmark-based method for obtaining a reward signal using low-resolution RGB cameras. In this way, the proposed method enables adapting the IMU model to each drone separately, accounting for possible drone-specific dynamics, with a very small cost compared to supervised approaches, e.g., (Herath et al., 2020). Extensive experiments using both simulations and actual hardware were performed demonstrating the improvements obtained using the proposed method.

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 an 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. Therefore, using Euler’s method we obtain:

$$p(t + h) = p(t) + \Delta t v(t), \quad (1)$$

where $p(t)$ denotes the position at time step t , Δt the time step duration, and $v(t)$ the velocity estimated by the IMU, typically as $v(t) = v(t - 1) + \frac{dv}{dt} \Delta t$. Thus, we estimate the next instant position, taking into account an initial position at every constant time step. Note, we assume that the velocity between two measurements remains constant throughout the flight. This simple approach enables UAV localization through IMU sensors that can provide speed estimates. However, the noise that 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 localization accuracy. Let $G_{\mathbf{W}} : \mathbb{R}^m \rightarrow \mathbb{R}^n$ 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 displacement measurements, including the current one, provided by the IMU. Then, the model $\mathbf{y} = G_{\mathbf{W}}(\cdot)$ can be trained to provide corrected displacement estimates for the current state, denoted by $\mathbf{y} \in \mathbb{R}^n$. Furthermore, in this paper we opted to deal only with the errors in the 2D plane ($n = 2$), since the vertical component is typically provided by barometric altimeters which are much more accurate in measuring displacements in typical conditions (Tang et al., 2005). 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 displacement estimates. After collecting adequate training samples of IMU velocity estimations and the associated ground-truth displacements then the model $G_{\mathbf{W}}(\cdot)$ can be trivially trained, e.g., by minimizing the mean square error between the predictions and ground truth data. During the supervised training step the model is trained to correct IMU errors using pairs of noisy IMU displacement observations and ground

truth displacement annotations. These are generated in simulation by using predefined noise models and the parameters of the simulation to get the absolute position of the drone. The setup is very similar to the learning from demonstrations paradigm (Puranic et al., 2021). After estimating the displacement error, we can acquire a more reliable estimation of the UAV’s position. Therefore, the final position at time step t is estimated as:

$$p(t+h) = p(t) + \Delta t v(t) + G_{\mathbf{w}}(\mathbf{v}_t), \quad (2)$$

where \mathbf{v}_t denotes the velocity vector at time step t .

2.2. Data efficient DRL-based training

Even though 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 as shown in Fig. 1: a) train a generic DL-based backbone model in a simulator and b) fine-tune this model on a real UAV using DRL. In this way, the employed DRL fine-tuning approach allows for dealing with potential inconsistencies between the simulation environment and the real drone, as well as for adapting the employed model to the exact dynamics of each drone. 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 an additional loss function that can increase data efficiency. As we demonstrate in Section 3, this can drastically reduce the number of training data that needs to be collected, making it feasible to apply the proposed method in real world scenarios.

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 state and a sphere indicating its estimated position/state. The DRL agent controls the latter by providing continuous corrections in the two axes of the 2d plane. To make the developed agent agnostic to the reference point of the used coordinate system, we employ displacements instead of actual positions. Therefore, the state space of the agent $\mathcal{S} \in [-d_{max}, d_{max}]^2$ consists of all possible displacements in 2D

the drone can have in one time step, where d_{max} denotes the maximum displacement that can occur in one time step. Then, an observation \mathbf{o}_t at each time step consists of the displacements estimated by the drone’s IMU. For a state $\mathbf{s}_t \in \mathcal{S}$ the corresponding observation \mathbf{o}_t is generated through a non-deterministic function $h(\cdot)$ that models the IMU dynamics, i.e., $\mathbf{o}_t = h(\mathbf{s}_t)$. Then, the action space is defined to be identical with the state space, i.e., $\mathcal{A} = \mathcal{S}$, since the goal of the DRL agent is to actually estimate its state based on the current observation. In other words, the goal of the DRL formulation employed in the proposed method is to model the function $h(\cdot)$ by enabling the agent to estimate its state. Then, the agent’s reward is provided by measuring how much error we have experienced in state estimations at the end of one training episode. Also, note that the Markov property is satisfied, since indeed the evolution of the process only depends on the present state and the past behavior does not affect in any other way the future.

This setup also enables an easy way to acquire the feedback signal for training the agent both in simulation and in the real world. 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 intuitive, 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:

$$r_k = \frac{1}{1 + |p_k|}, \quad (3)$$

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 a 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) (Schulman et al., 2017) 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 adapt 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 $\phi \in [0, 360)$. This process is shown in Fig. 1 and significantly improves the generalization abilities of the agent by providing angle invariance without the need for collecting additional data.

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 loss that provides additional supervision during the training process. The aim of the proposed hint loss is to provide a *direct* way to alter the direction of weight updates based on the structure (overshooting or undershooting) of observed errors in order to accelerate the learning process and increase data efficiency. The proposed hint loss for each axis is defined as:

$$\mathcal{L}_{hint,k} = -\alpha_{hint} \cdot \delta_k \cdot g_{RL}(x) \quad (4)$$

where α_{hint} is the weight of the hint loss, δ_k 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 proposed hint loss for both axes to the PPO loss.

3. Experimental Evaluation

3.1. Experimental Setup

We performed tests employing both a simulated environment, i.e., for supervised learning and validation of the proposed DRL methodology, as well as a real UAV. For the simulated experiments, we employed Webots (Michel, 2004). For supervised learning, we collected 500 episodes with velocities and ground truth positions. We also experimentally found that the IMU measurements of the drone (DJI Mavic Mini 2) used in the conducted experiments is

biased depending on the vehicle’s velocity and it is always underestimated. We use the following noise model for speed measurements v' when generating IMU observations:

$$v' = \frac{v}{(1 + 1/(1 + c * |v|))}, \quad (5)$$

where $c \sim \mathcal{N}(\mu, \sigma)$, μ and σ are IMU-depended factors, $\mathcal{N}(\mu, \sigma)$ denotes a normal distribution with mean μ and standard deviation σ and v is the ground truth velocity. Therefore, the trained agent tries to uncover the structure of errors induced by this noise model (either directly - when trained in a supervised fashion - or indirectly - when trained using DRL). For training the supervised learning model, we used $c = 5$ and $\sigma = 0.1$. The DRL experiments focused on evaluating the ability of the proposed method to promptly adapt to changes that can occur due to configuration changes, e.g., changing the characteristics of the drone or even deploying the trained agent into other platforms. Therefore, after pre-training the backbone network, we changed the value of c to 2 (σ is still set to 0.1) to simulate the drift that can occur due to the aforementioned changes.

In this work, we opted for a multi-layer perceptron (MLP)-based architecture since recent studies on similar tasks showed small differences between MLPs and more powerful estimators (Liu et al., 2022b). The MLP used has two hidden layers, each containing 12 neurons, and used the hyperbolic tangent (tanh) activation function. The output of the MLP was fed into two branches, each of which was responsible for predicting a continuous value (one corresponding to the velocity error of each axis). In every branch, there are two extra trainable parameters that are used for shifting and scaling the output of the network. We found that when we re-training the network with new data, using these parameters to adjust the output 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 axis. The mean squared error loss function was used for the supervised model training. The Adam optimizer with a learning rate of 3×10^{-2} was used. The default hyper parameters (i.e., α and β) were used (Kingma and Ba, 2014). 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 experiments we used an actor-critic architecture. We directly used the above network as the DRL actor. For the critic model we

Table 1: DRL fine-tuning evaluation on Webots (straight trajectories) using a distribution shift scenario ($c \sim \mathcal{N}(2, 0.1)$)

metric	10 secs		
	baseline	supervised	proposed
MSFPE	12.297	2.975	0.142
MFPE	3.438	1.692	0.324
MPE	1.714	0.931	0.242
ATE	2.008	1.073	0.290
100 secs			
MSFPE	1225.922	312.100	18.832
MFPE	34.437	17.444	3.816
MPE	18.376	141.05	2.053
ATE	21.014	10.696	7.613

used a similar MLP with two hidden layers of 12 neurons. The models were trained for 100 epochs with a learning rate of 5×10^{-5} and 5×10^{-3} in actor and critic models accordingly. The PPO algorithm was used for fitting the DRL agent, while the clipping factor was set to 0.15. For these experiments, an episode is defined as a predetermined sequence of M micro-steps during flight. In each episode the drone follows a random trajectory that consists of M steps. For instance, a random episode may involve 20 micro-steps forward followed by 40 micro-steps left. For all the conducted experiments the episode duration was set to 10 seconds ($M = 300$), while the IMU was pooled at a frequency of 25 Hz. Finally, we have conducted various experiments to evaluate the impact of the weight of the hint loss a_{hint} . These experiments indicated that for values around 2.5 we obtain the best performance consistently ($a_{hint} = 2.6$ was used for all the conducted experiments).

3.2. Evaluation in Simulation Environment

First, we evaluated the proposed method in a simulation environment using Webots. The results are reported in Table 1 (straight trajectories) and in Table 2 (circular trajectories), where we report the mean squared final position error (MSFPE), mean final position error (MFPE), mean positional error (MPE), and absolute trajectory error (ATE) between the ground truth displacement and the estimated by the DRL model. We introduce the first two metrics and calculate the error between the *last* actual and estimated vehicle’s position. These results (Table 1) indicate that the proposed method

Table 2: DRL fine-tuning evaluation on Webots (circular trajectories) using a distribution shift scenario ($c \sim \mathcal{N}(2, 0.1)$)

metric	10-20 secs		
	baseline	supervised	proposed
MSFPE	20.22	4.99	0.47
MFPE	4.45	2.22	0.66
MPE	2.63	1.39	0.49
ATE	3.01	1.57	0.55
metric	40-50 secs		
	baseline	supervised	proposed
MSFPE	26.52	6.27	0.65
MFPE	4.99	2.48	0.77
MPE	3.19	1.62	0.58
ATE	3.52	1.78	0.64

Table 3: DRL fine-tuning evaluation on Webots (circular trajectories) using a distribution shift scenario ($c \sim \mathcal{N}(2, 0.5)$)

metric	10-20 secs		
	baseline	supervised	proposed
MSFPE	21.35	5.58	0.61
MFPE	4.57	2.35	0.75
MPE	2.70	1.46	0.54
ATE	3.09	1.65	0.61
metric	40-50 secs		
	baseline	supervised	proposed
MSFPE	28.23	7.13	0.82
MFPE	5.15	2.64	0.87
MPE	3.28	1.71	0.64
ATE	3.62	1.89	0.71

can improve DRL agents’ performance when training under a constrained number of episodes (i.e., 15 only episodes). Note that as the duration of an episode increases, the error still accumulates. Note that for both types of trajectories, i.e., both straight and circular ones, the proposed method leads to significant improvements in all the evaluated metrics, even though the error is slightly higher on the more complex trajectories. Furthermore, we also investigated the sensitivity of the evaluated methods by alerting the standard deviation of noise (σ) in the IMU model. The experimental results are reported in Table 3. Again, the proposed method still reduces the error significantly, while only slightly reducing the accuracy compared to the best results reported in Table 2.

A qualitative comparison between Euler’s method with no adjustments, 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. 2. Again in this evaluation we include both straight trajectories (first three evaluations), as well as circular ones (remaining nine evaluations). We demonstrate that employing DRL for the purpose of fine-tuning the model that was learned in simulation leads to significant improvements. Indeed, the proposed method manages to significantly reduce all the error metrics compared to the baseline.

Next, we performed additional experiments to evaluate the impact of using the proposed data efficient DRL strategy compared to directly using vanilla DRL. The results for different numbers of episodes are shown in Table 4. It is worth noting that the proposed method can lead to significant improvements since in just three episodes we can achieve the same performance as the vanilla PPO in fifteen episodes. Furthermore, we conducted an ablation study to evaluate the impact of each component of the proposed method in order to better justify the main design choices of the proposed method. The experimental evaluation of this ablation study is reported in Fig. 3. First, note that the proposed augmentation method leads to improvements, but the improvements are evident after at least 30 episodes, which can be prohibitive for real applications. Then, in order to further accelerate the learning process, we employ the proposed hint loss which accelerates the learning process from the beginning. Finally, combining both approaches can lead to improvements during the early stages of the learning process, which are the most crucial for real applications. However, as the training progress the benefits obtained using the data augmentation approach are smaller.

Table 4: Comparing the agent’s performance relative to training episodes with and without using the proposed data efficient DRL method.

PPO						
episodes	0	3	6	9	12	15
MSFPE	3.03	2.18	1.71	1.39	1.48	1.60
MFPE	1.71	1.45	1.30	1.16	1.20	1.24
MPE	0.92	0.79	0.72	0.66	0.68	0.71
ATE	1.07	0.91	0.84	0.77	0.79	0.82
PPO + PROPOSED						
MSFPE	3.03	1.67	1.12	0.60	0.37	0.14
MFPE	1.71	1.26	1.04	0.73	0.57	0.32
MPE	0.92	0.69	0.56	0.44	0.29	0.24
ATE	1.07	0.80	0.65	0.51	0.34	0.29

Zero in episodes indicates directly using the supervised model without DRL fine-tuning for the specific UAV.

Table 5: DRL fine-tuning evaluation using a DJI Mavic Mini 2 UAV.

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

The percentage of estimated distance covered to the true distance (MPE, %) is reported for different flying speeds.

Table 6: Experimental evaluation using a DJI Mavic Mini 2 using specific target distances for various speeds

Goal	Velocity	baseline			proposed		
		TF	GT	AETD%	TF	GT	AETD%
12.7	0.31	49.77	16.85	75.38	42.84	14.5	87.58
12.2	0.31	49.04	16.09	75.83	42.37	13.9	87.77
25.2	0.61	45.64	28.61	88.07	43.07	27.0	93.33
14.0	1.71	9.30	15.05	93.03	8.96	14.5	96.55

3.3. Evaluation using a real UAV

We also conducted experiments using data collected from a real UAV. We used two independent setups, where a DJI Mavic Mini 2 was employed. First, we validated the trained agent proposed method using several trajectories performed at different velocities. The results are reported in Table 5. The obtained results confirm that for a wide range of different speeds the proposed method indeed leads to better performance in relation to the baseline.

We also conducted a second evaluation, in which we measure the total positional error of several UAV flights. In this experiment we used the following setup: the user requested that the UAV move forward by x meters at a constant velocity. Thus, using any localization method, the UAV will be able to travel at a predetermined speed toward the user-specified location. When the employed localization method estimates that the UAV has traveled at least x meters, then it immediately stops its movement. This experiment allows us to evaluate both the accuracy of the localization, as well as to examine the impact of total flight time in the accumulated errors. The results are reported in Table 6, where TF denotes the total time-flight in seconds, GT is the ground-truth displacement in meters and AETD denotes the validation metric which is defined as the proportion of the actual distance over the predicted distance whether the vehicle has moved in straight ahead (only forward/backward). The proposed RL method was compared to the baseline Euler’s method in this experiment.

Four experimental flights were conducted to this end. For the first experiment, we set the drone’s goal to move 12.7 meters forward at a speed of 0.31 meters per second. Using the baseline approach the drone traveled 16.85 meters while using the proposed method it stopped significantly closer to the target reaching 14.5 meters. For the second experiment, we used a slightly smaller distance, and similar results were observed. For the subsequent experiments, we increased the speed and varied the target distance to be traveled. In all cases, we observed that the obtained error got smaller as the speed increased. The obtained error got even smaller as the total flight duration got smaller, as expected.

4. Conclusions

In this paper we introduced a method for improving the accuracy of data-driven navigation for UAVs using IMUs. The proposed method involves training a base model using supervised learning, followed by fine-tuning with

an appropriately modified data efficient DRL technique in order to accommodate the needs of the specific application. We demonstrated that the proposed method can effectively improve inertial-based navigation, particularly in situations where the IMUs on UAVs may have diverse characteristics and require customized fine-tuning with only a small number of actual flight episodes. We demonstrated the effectiveness of the proposed method on different trajectory types, including both straight and circular trajectories. However, the performance of the proposed method on different applications might be dependent on the actual characteristics of the involved trajectories.

References

- Ahmad, N., Ghazilla, R.A.R., Khairi, N.M., Kasi, V., 2013. Reviews on various inertial measurement unit (imu) sensor applications. *International Journal of Signal Processing Systems* 1, 256–262.
- Alotaibi, E.T., Alqefari, S.S., Koubaa, A., 2019. Lsar: Multi-UAV collaboration for search and rescue missions. *IEEE Access* 7, 55817–55832.
- Barshan, B., Durrant-Whyte, H.F., 1995. Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation* 11, 328–342.
- Boonyathanmig, N., Gongmanee, S., Kayunyeam, P., Wutticho, P., Prongnuch, S., 2021. Design and implementation of mini-UAV for indoor surveillance, in: *Proceedings of the 9th International Electrical Engineering Congress*, pp. 305–308.
- Brossard, M., Barrau, A., Bonnabel, S., 2019. Rins-w: Robust inertial navigation system on wheels, in: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2068–2075.
- Cortés, S., Solin, A., Kannala, J., 2018. 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*, pp. 1–6.
- Giubilato, R., Stürzl, W., Wedler, A., Triebel, R., 2022. Challenges of slam in extremely unstructured environments: The dlr planetary stereo, solid-state lidar, inertial dataset. *IEEE Robotics and Automation Letters* 7, 8721–8728.

- Gyagenda, N., Hatilima, J.V., Roth, H., Zhmud, V., 2022. A review of gnss-independent uav navigation techniques. *Robotics and Autonomous Systems* , 104069.
- Han, S., Wang, J., 2012. Integrated GPS/ins navigation system with dual-rate kalman filter. *GPS Solutions* 16, 389–404.
- He, Y., Yuan, M., Zhao, L., Dong, S., 2022. Application of compressed sensing in mono slam system with fusion of imu, in: *Advances in Guidance, Navigation and Control*, pp. 5121–5130.
- Henkel, P., Mittmann, U., Iafrancesco, M., 2016. Real-time kinematic positioning with GPS and glonass, in: *Proceedings of the 24th European Signal Processing Conference*, pp. 1063–1067.
- Herath, S., Yan, H., Furukawa, Y., 2020. Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, & new methods , 3146–3152.
- Jaboyedoff, M., Oppikofer, T., Abellán, A., Derron, M.H., Loye, A., Metzger, R., Pedrazzini, A., 2012. Use of lidar in landslide investigations: a review. *Natural Hazards* 61, 5–28.
- Jin, Y., Yu, L., Chen, Z., Fei, S., 2022. A mono slam method based on depth estimation by densenet-cnn. *IEEE Sensors Journal* 22, 2447–2455. doi:10.1109/JSEN.2021.3134014.
- Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 .
- Koestler, L., Yang, N., Zeller, N., Cremers, D., 2022. Tandem: Tracking and dense mapping in real-time using deep multi-view stereo, in: *Conference on Robot Learning*, pp. 34–45.
- Liu, X., Nardari, G.V., Ojeda, F.C., Tao, Y., Zhou, A., Donnelly, T., Qu, C., Chen, S.W., Romero, R.A., Taylor, C.J., et al., 2022a. Large-scale autonomous flight with real-time semantic slam under dense forest canopy. *IEEE Robotics and Automation Letters* 7, 5512–5519.
- Liu, Y., Luo, Q., Zhou, Y., 2022b. Deep learning-enabled fusion to bridge gps outages for ins/gps integrated navigation. *IEEE Sensors Journal* 22, 8974–8985.

- Michel, O., 2004. Cyberbotics ltd. webots™: professional mobile robot simulation. *International Journal of Advanced Robotic Systems* 1, 5.
- Newaz, A.A.R., Jeong, S., Lee, H., Ryu, H., Chong, N.Y., 2016. Uav-based multiple source localization and contour mapping of radiation fields. *Robotics and Autonomous Systems* 85, 12–25.
- Puranic, A.G., Deshmukh, J.V., Nikolaidis, S., 2021. Learning from demonstrations using signal temporal logic in stochastic and continuous domains. *IEEE Robotics and Automation Letters* 6, 6250–6257.
- Radoglou-Grammatikis, P., Sarigiannidis, P., Lagkas, T., Moscholios, I., 2020. A compilation of UAV applications for precision agriculture. *Computer Networks* 172, 107148.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Sukkarieh, S., Nebot, E.M., Durrant-Whyte, H.F., 1999. A high integrity imu/GPS navigation loop for autonomous land vehicle applications. *IEEE Transactions on Robotics and Automation* 15, 572–578.
- Tang, W., Howell, G., Tsai, Y.H., 2005. Barometric altimeter short-term accuracy analysis. *IEEE Aerospace and Electronic Systems Magazine* 20, 24–26.
- Vetrella, A.R., Fasano, G., 2016. Cooperative uav navigation under nominal gps coverage and in gps-challenging environments, in: *Proc. IEEE International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow*, pp. 1–5.
- Xiao, L., Wang, J., Qiu, X., Rong, Z., Zou, X., 2019. Dynamic-slam: Semantic monocular visual localization and mapping based on deep learning in dynamic environment. *Robotics and Autonomous Systems* 117, 1–16. URL: <https://www.sciencedirect.com/science/article/pii/S0921889018308029>, doi:<https://doi.org/10.1016/j.robot.2019.03.012>.
- Xu, W., Zhang, F., 2021. Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter. *IEEE Robotics and Automation Letters* 6, 3317–3324.

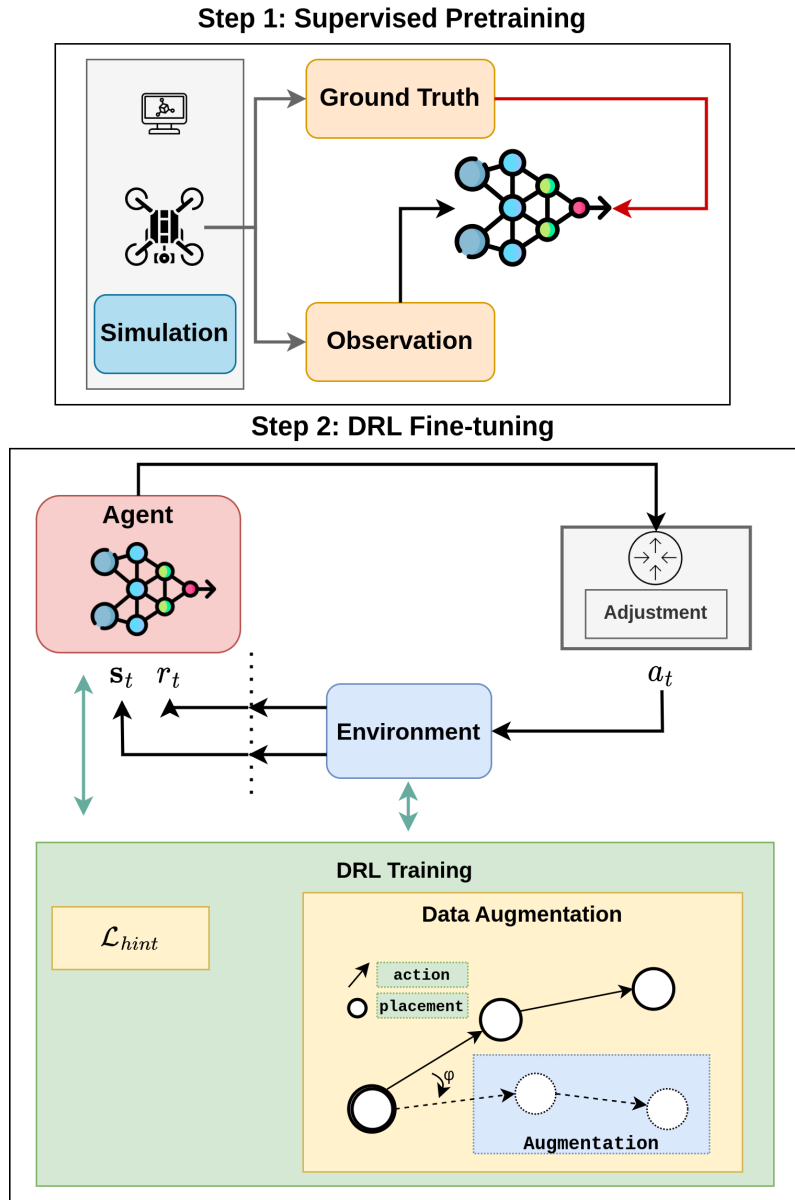


Figure 1: The proposed method employs a two step pipeline. First, we train a generic DL-based backbone model in a simulator in a supervised fashion and b) fine-tune this model on a real UAV using DRL. To increase data efficiency and reduce the number of episodes required to fine-tune the model we employ both a hint loss and a data augmentation method to generate synthetic episodes by rotating existing episodes by a random angle.

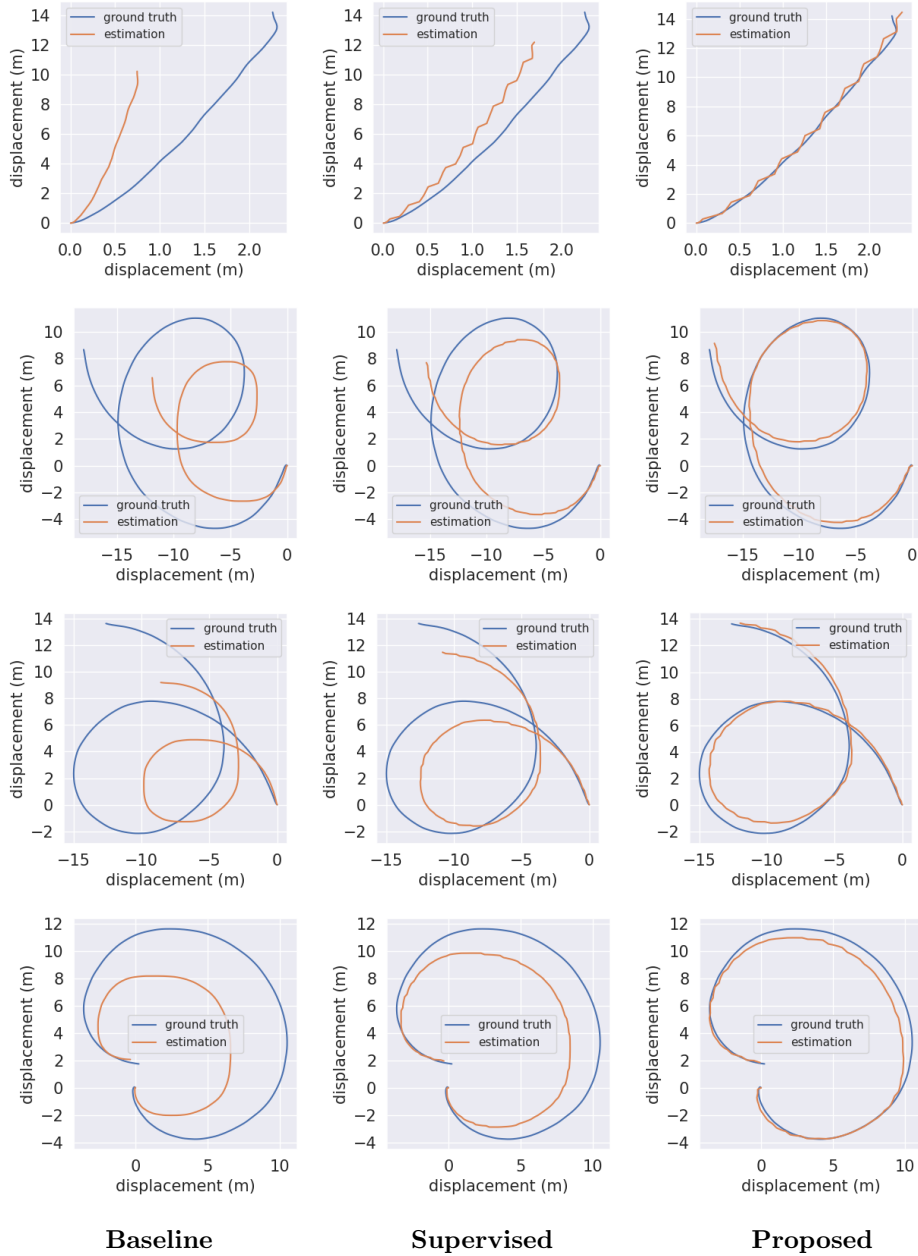


Figure 2: Comparison between baseline (Euler’s method) (left of each triplet), supervised training (middle of each triplet) and RL-based fine-tuning (right of each triplet). Each axis corresponds to the displacement of a UAV in the 2d space when flying on a pre-determined course, including four different trajectory types used for the evaluations.

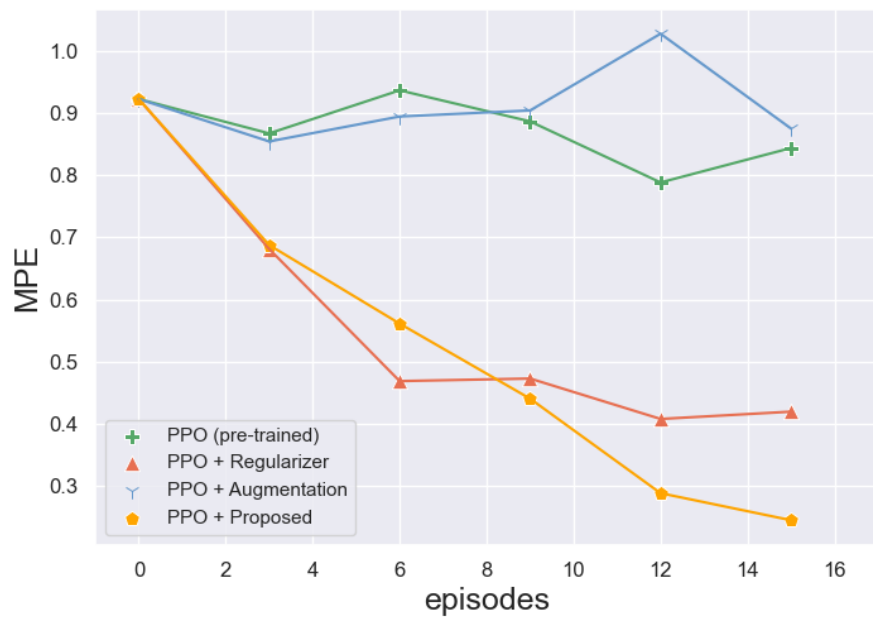


Figure 3: Ablation study evaluating the impact of each of the proposed components of the proposed method.

E

Learning Hierarchical Interactive Multi-Object Search for Mobile Manipulation

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

Abstract—Existing object-search approaches enable robots to search through free pathways, however, robots operating in unstructured human-centered environments frequently also have to manipulate the environment to their needs. In this work, we introduce a novel interactive multi-object search task in which a robot has to open doors to navigate rooms and search inside cabinets and drawers to find target objects. These new challenges require combining manipulation and navigation skills in unexplored environments. We present HIMOS, a hierarchical reinforcement learning approach that learns to compose exploration, navigation, and manipulation skills. To achieve this, we design an abstract high-level action space around a semantic map memory and leverage the explored environment as instance navigation points. We perform extensive experiments in simulation and the real-world that demonstrate that HIMOS effectively transfers to new environments in a zero-shot manner. It shows robustness to unseen subpolicies, failures in their execution, and different robot kinematics. These capabilities open the door to a wide range of downstream tasks across embodied AI and real-world use cases.

I. INTRODUCTION

Autonomous navigation and exploration in unstructured indoor environments require a large variety of skills and capabilities. Pathways may be blocked and objects of interest may be stored away. Thus far, existing multi-object search tasks and methods have focused on environments that can be freely navigated with openly visible target objects [1], [2], [3]. We introduce a novel *Interactive Multi-Object Search* task in which target objects may be located inside articulated objects such as drawers and closed doors have to be opened to explore the environment. As a result, only navigation is insufficient to accomplish the task and the robotic agent has to physically interact with the environment to manipulate it to its needs.

Multi-object search tasks pose long-horizon problems with non-trivial optimal policies. Methods such as frontier exploration [4] offer guarantees to explore the entire environment if given enough time. However, they often do not take the context of the environment into account and result in long far from optimal paths while moving from one frontier point to the next. On the other end of the spectrum, learning-based methods can take unstructured observations into account and have been shown to learn good exploration strategies [1], [3], but they struggle with the long-horizon nature of the task. Moreover, since the robot also has to interact with the environment, both the action space and task horizon increase even further, and existing exploration methods are insufficient to accomplish the task.

*These authors contributed equally.

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. Toyota Motor Europe supported this project with an HSR robot for experiments.

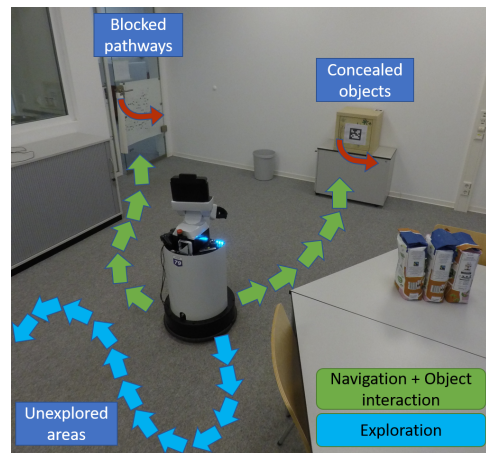


Fig. 1. We introduce the *Interactive Multi-Object Search* task in which an agent has to autonomously search and manipulate the environment to find a set of target objects. To succeed, the agent has to free pathways by opening doors and searching inside articulated objects such as cabinets and drawers.

In this work, we propose Hierarchical Interactive Multi-Object Search (HIMOS), a hierarchical reinforcement learning approach to learn both exploration and manipulation skills and to reason at a high level about the required steps. We combine learned motions for local exploration in continuous action spaces [3] and frontier exploration for long-horizon exploration [4] together with mobile manipulation skills for object interactions [5]. We use semantic maps as the central memory component, which have shown to be an expressive and sample-efficient representation for these tasks [3] and design a high-level action space that exploits the acquired knowledge about the environment. By leveraging explored object instance locations as navigation waypoints, our approach efficiently learns these complex tasks from little data and consistently achieves success rates above 90% even as the number of target objects increases. By equipping all the low-level skills with mobility, we remove the “hand-off” problem in which subpolicies have to terminate in the initial set of the following skill [6], [7]. Lastly, we transfer the trained agent to the real world and demonstrate that it successfully accomplishes these tasks in a real office environment. In particular, we replace the subpolicies from simulation with unseen real-world variations and find that the policy is able to generalize to these unseen subpolicies and is robust to failures in their execution, making it highly modular and flexible for transfer. Finally, we present ablation studies to evaluate the impact of the main design decisions.

To summarize, the following are the main contributions:

- 1) We propose an interactive multi-object search task that requires physical interactions with articulated objects, opening doors, and searching in cabinets and drawers.
- 2) We present a hierarchical reinforcement learning approach that combines exploration and manipulation skills based on semantic knowledge and instance navigation points to efficiently solve these long-horizon tasks.
- 3) We demonstrate the capabilities of this approach in both simulated and real-world experiments and show that it achieves zero-shot transfer to the real world, unseen environments, unseen subpolicies, and is robust to unseen failures.
- 4) We make the code for both the task and models publicly available at <http://himos.cs.uni-freiburg.de>.

II. RELATED WORK

Object search tasks: Exploration and the ability to find items of interest is a key requirement for a wide range of downstream tasks. Previous work has proposed methods to maximize coverage of explored space [8] and to find specific objects of interest based on vision [9], auditory signals [10], [11] or target object categories [12], [13]. In ordered multi-object search tasks, the agent has to find k items in a fixed order in game environments [14] or realistic 3D apartments [1]. In unordered multi-object search, the agent simply has to find the target objects as fast as possible, irrespective of the order [2], [3]. We focus on this unordered task. As we aim to demonstrate our system on a real robot, we follow Schmalstieg et al. [3] and use the full continuous action space. This is in contrast to most previous work which focuses on a simplified granular discrete action space. Existing search tasks assume that the desired goals can be freely reached by the agent. The interactive navigation task [15] relaxes this assumption by placing objects that the robot has to push away to reach the target. In contrast, we introduce an interactive search task for mobile robots equipped with a manipulator, that requires interaction with articulated objects to clear the path or reveal concealed objects. This requires integrating navigation and manipulation. Lastly, in contrast to most previous work, we demonstrate that our approach successfully transfers to the real world.

Exploration requires both understanding and memorizing the seen environment and decision-making to explore the remaining space. Previous work has introduced both implicit and explicit memory mechanisms. Implicit memory agents either learn a direct end-to-end mapping from RGB-D images to actions or store embeddings of previous observations and retrieve them with an attention mechanism [2]. Other methods build explicit maps of the environment by projecting the RGB-D inputs into a global map. Commonly, this map is also annotated with semantic labels [14], [3], [1]. Further, combining short- and long-term exploration by learning an auxiliary prediction of the direction to the next closest object has proven to result in a strong performance in continuous action spaces [3]. We use this approach for low-level exploration. Frontier exploration [4] samples points on the frontier of the explored space and then navigate to these points. Instead of sampling, Ramakrishnan et al. [16] predict a potential function

towards a target object. SGoLAM [17] combines mapping and a goal detection module. If no goal object is detected, it explores with frontier exploration and navigates directly to the goal otherwise. This results in strong results without any learning component. We include a similar exploration strategy as option in our hierarchical approach.

Articulated object manipulation such as opening doors and drawers requires control of both the base and arm of the robot [18]. Existing approaches often separate both aspects and execute sequential navigation and manipulation. In our evaluation in simulation, we follow this approach and use BiRRT [19] to generate a motion plan for the robot arm. Recent work trains an agent to control the base of the robot via reinforcement learning to follow given end-effector motions [20], [5]. We use this method in our real world evaluation as it has been shown to generalize across different robots, tasks, and environments.

Hierarchical methods introduce layers of abstraction by decomposing the decision-making into higher and lower-level policies. This shortens the time horizon of the Markov decision process (MDP) for the higher-level policies and enables the agent to combine different modules or skills at lower levels. At the same time, joint training of low- and high-level policies is often unstable and hard to optimize [21]. We focus on combining pretrained subpolicies. Joint finetuning of these policies offers a path to further performance improvements in the future. Pretraining is a common approach to increase the stability of the policy in hierarchical reinforcement learning [22]. While naive skill-chaining of arbitrary skills often results in "hand-off" failures in which the subsequent skill cannot start from the current state [7], we resolve this issue by adding mobility to all the low-level skills in the real world execution, without the need for region-rewards [6]. A common navigation abstraction is to learn to set waypoints [23], [24], however, these often end up as very near points to the agent. In contrast, we propose instance navigation that provides a prior on important locations and action granularity. ASC [25] learns to combine navigation and pick skills for given receptacle locations. In contrast, we learn to search in unexplored environments with objects hidden in articulated objects. Alternatively, behavior trees are often used to decompose tasks hierarchically into a tree structure. These trees can either be fully constructed manually or be used in conjunction with a planner [26]. In contrast to these approaches, our proposed HIMOS learns high-level decision-making with a two-layer hierarchy.

III. INTERACTIVE MULTI-OBJECT SEARCH

We propose an interactive object search task in which a robotic agent with a mobile base and a manipulator arm is randomly spawned in an unexplored indoor environment. The agent receives a goal vector that indicates k objects out of c categories that it has to find. The episode is considered successful if the agent finds all k objects, where an object is considered found when the agent has seen the object and navigated within a distance of 1.3m of it. The episode is terminated early if the agent exceeds 1,000 timesteps. To succeed, the agent has to explore the space while opening

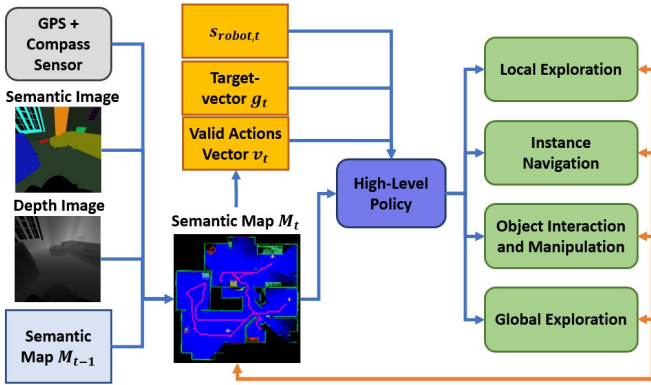


Fig. 2. Schematic overview of HIMOS. A semantic map M_t serves as a central memory component and is used and updated across low- as well as high-level modules. This map is extended to a partial panoptic map with instance IDs of relevant objects. Given the remaining target objects g_t to find, the robot state $s_{robot,t}$, and the derived valid actions v_t , the high-level policy acts in an abstract action space. Low-level actions comprise local and global exploration, navigation to previously mapped object instances, and a mobile manipulation policy.

doors that block the way and opening the cabinets that contain the target objects. This results in very long-horizon tasks with complex shortest paths, as in contrast to previous multi-object search tasks [2], [14], [1], [3], the agent needs to manipulate its environment to achieve its goals.

The agent is acting in 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. The objective 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. In each step, the agent receives a visual observation o from an RGB-D and semantic camera, together with its current pose in the environment and a binary vector g indicating the objects it has to find. The true state s of the environment is unknown and can only be inferred from its observations.

IV. HIERARCHICAL INTERACTIVE MULTI-OBJECT SEARCH

The challenges introduced by this task require the robot to master an increasing number of different behaviors and skills. To stem this increase in complexity, we propose a hierarchical reinforcement learning approach and design efficient abstractions over states and actions. The model consists of three main components: a mapping module, a set of subpolicies for exploration, navigation, and interaction, and a high-level policy module to select the next low-level action to execute. An overview of the approach is depicted in Figure 2.

Assumptions: The focus of this work is on learning long-horizon decision-making, exploration, and search. We abstract from real-world perception and assume to have access to the following: (i) a semantic camera that produces accurate semantic object labels, (ii) accurate depth and localization, (iii) an object detection module to perceive object poses in the environment and to detect whether an object interaction was successful.

In training and simulation, the object detection module uses ground truth poses from the simulator while in real-world experiments, we rely on AR markers placed on the respective objects. Interaction failure is detected based on the change in pose of these markers after the interaction.

A. Mapping Module

A semantic map of the environment serves as a central memory component across all policies and modules. To construct this map, we build upon our previously introduced mapping module [3]. In the simulation, the robot receives the semantic labels from the simulator’s semantic camera, then uses the 128×128 pixels depth and semantic camera observations to project the points into a local top-down map. In the real world, we simulate access to semantic labels as follows: we pre-build a map of the environment using Hector SLAM [27] and annotate it with labels for target objects, cabinets, and doors. All the other occupied space is mapped to the wall category. At test time, we then use the robot’s depth camera to build the same local map as in the simulation and overlay it with this pre-annotated semantic map. Localization in the real world is done on the same pre-recorded map. With the local map and its current localization, the agent then updates an internal global map which is further annotated with the agent’s trace and encoded into an RGB image.

In each step, the agent then extracts an egocentric map from the global map 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. I.e., they cover $14.8 \text{ m} \times 14.8 \text{ m}$ and $2.77 \text{ m} \times 2.77 \text{ m}$, respectively. After the agent has segmented and approached a target object, it updates the object’s annotation to a fixed color coding to mark the corresponding object as “found”. We extend this map to a partial panoptic segmentation [28] map by labeling task-specific objects such as doors and cabinets with an instance-specific color, which is randomly assigned whenever a new instance is detected, i.e. an instance’s color changes from episode to episode, but remains consistent within an episode.

B. Subpolicy Behaviors

In this section, we describe the different subpolicy behaviors that are available to the high-level policy.

Local Exploration: Object search requires smart movement for local navigation and efficient exploration around corners and corridors. For this, we use our previously introduced exploration policy [3]. The policy receives the semantic map, robot state, and target objects and predicts the direction to the next closest target object. It then communicates this prediction to a reinforcement learning agent which produces target velocities for the base of the robot. The local exploration policy is pretrained on the multi-object-search task [3] in the same train/test split. However, we change the robot to Fetch, adapt the collision penalty from -0.1 to -0.15 , and include open doors in the scenes. After the exploration policy is trained, it is kept frozen during high-level policy training. We adjust the

panoptic labels provided by the mapping module on the fly to match the simpler, instance-unaware, semantic map that this subpolicy was trained with. In particular, doors and cabinets are colored as obstacles for the subpolicy. This shields the subpolicy from information that is not required to solve the downstream task. During training, when selected by the high-level policy, the exploration policy is executed for four time steps, giving the high-level policy control to quickly react to new knowledge of the environment. During the evaluation, we found it beneficial to execute it for a longer period of 20 time steps.

Global Exploration: While the local exploration policy has been shown to produce efficient search behavior, it can struggle to navigate to faraway areas. This, however, is a strength of frontier exploration [4], which samples points at the frontier to (often far away) unexplored areas. While frontier exploration on its own can lead to long inefficient paths, the high-level policy can learn to select the appropriate exploration strategy for the current context. In each iteration, the sampled frontier point is drawn onto the map, allowing the high-level policy to observe where it would navigate to before deciding which subpolicy to execute. Frontier points that lie outside the range of the agent’s egocentric map are projected onto a circle around the agent and marked in a different color, indicating that it is potentially a long-distance navigation. If selected by the high-level policy, the agent uses its navigation policy described below to navigate to the frontier point.

Instance Navigation: Learning navigation at the right level of abstraction can be challenging. Approaches such as setting waypoints are often difficult to optimize or decay to only selecting nearby points, removing the benefits of the abstraction. Instead, we leverage the acquired knowledge about the environment by using object instances as navigation points: the high-level policy can directly navigate to the discovered object instances by selecting their instance ID (for simplicity restricted to target objects, doors, and cabinets). We implement this action space as a one-hot encoding that maps to instance colors on its map (this assumes a maximum number of instances). We furthermore find it beneficial for learning speed [29] to only allow the agent to navigate to doors or cabinets that have not been successfully opened yet (cf. invalid action masking below). While less fine-grained than arbitrary waypoints, this results in an efficient set of navigable points across the map that, as we demonstrate in Section V, is well optimizable and results in a strong final policy.

The respective navigation goals are set to a pose slightly in front, or for the goal objects directly to the detected pose of the corresponding object. This navigation goal is then fed to an A^* -planner which produces a trajectory at a resolution of 0.5 m. For training speed, we do not execute the full path in simulation, but set the robot’s base pose to the generated waypoints and only collect observations from these points. In the real world, we use the ROS navigation stack to move the robot to the goal. The policy may fail in some situations, for example due to collision with obstacles or narrow doorways. In this case, the agent returns to the last feasible waypoint, and given the updated semantic map M_t , the high-level policy has to make a new decision.

Object Interaction and Manipulation: If the high-level policy chooses to navigate to a closed door or cabinet, this automatically triggers an interaction action that is executed once the navigation has been successfully completed. For fast simulation, we train the agent with magic actions that either open the object successfully or fail with a probability of 15% and leave the object untouched, in which case the agent has to decide whether to try to open it again. The training with failure cases enables it to learn a re-trial behavior to recover from failed attempts.

At test time, the agent has to physically execute the interactions. In the simulation, we replace the interaction subpolicy with a BiRRT motion planner [19] and inverse kinematics to execute a push-pull motion. The success of these motions depends on the pose of the robot and the object. Implementation details can be found in the supplementary material. In the real world, we replace these subpolicies with the N^2M^2 mobile manipulation policy [5]. Given the pose of the object handle (based on an AR marker) and the object label (door, drawer, or cabinet), it generates end-effector motions learned from demonstrations to open the object together with base commands that ensure that these motions remain kinematically feasible. The model is pretrained without additional retraining or finetuning. The agent returns a success indicator to the high-level policy and in case of failure, the agent again has to decide whether to repeat the interaction.

C. High-level Decision Making

Efficient high-level decision making requires the right level of abstractions of states and actions. We hypothesize that object- and instance-level decision making is such an efficient level of abstraction for embodied search tasks. We design a high-level policy around this idea. In particular: (i) We propose an instance navigation subpolicy that leverages the agent’s accumulated knowledge about the environment. It provides a prior on important places and on the granularity of navigation points, making it data-efficient and well-optimizable. (ii) As objects are discrete instances, the resulting full action space remains discrete, avoiding the complexities of mixed action spaces. At the same time, all the subpolicies still act directly in continuous action spaces, allowing for direct transfer to real robotic systems. (iii) We abstract from reasoning about exact robot placements in the real world by shifting the responsibility of mobility into the subpolicies. This ensures that the subpolicies can start from a large set of initial positions, resolving the “hand-off” problem from naive skill-chaining [7] and strongly simplifies the learning process for the high-level policy. Furthermore, it enables us to change out the subpolicies to unseen subpolicies in the real world. (iv) We incorporate subpolicy failures into the training process, enabling the high-level policy to learn a re-trial behavior if execution fails.

The high-level policy acts in a Semi-Markov Decision Process (SMDP) in which the actions model temporarily extended behaviors and act at irregular intervals [30]. The high-level action space consists of (i) invoking the local exploration policy [3], (ii) invoking global frontier exploration [4], (iii) instance navigation with subsequent object interaction (if available), where instance IDs are selected through a one-hot

vector mapping to fixed colors. The current task instantiation assumes a maximum of ten instances per episode, resulting in an overall 12-dimensional discrete action space.

Adaptive Discounting: The high-level policy acts at irregular intervals, as the duration of the subpolicies varies largely. We correct for this time bias and accurately reflect the long-term consequences of actions with adaptive discounting [31].

Invalid Action Masking: The availability of high-level actions varies with the state of the environment, e.g. navigating to an object instance depends on the instance being mapped, opening a cabinet is only possible if it is mapped and closed. We infer a valid actions vector v_t from the agent’s observations and incorporate it into the training process by masking out invalid actions as well as including it in the observation space of the agent. As a result, the agent can learn more effectively, speeding up the training process [29]. We implement the masking by replacing the logits of the invalid actions with a large negative number.

Architecture: The high-level policy observes the coarse semantic map which it encodes with a ResNet-18 [32]. Then it concatenates the features with the target object vector g_t , the binary valid actions vector v_t (see invalid action masking) and a robot state vector $s_{robot,t}$ consisting of linear and angular base velocities, sum of collisions over the last ten steps, a current collision flag, and a normalized history over the last 16 high-level actions taken. The high-level policy is trained with Proximal Policy Optimization (PPO) [33], based on an open-source implementation [34]. We report hyperparameters and architecture details in the supplementary material.

Reward Functions: The high-level policy is trained with the accumulated rewards of the invoked subpolicies. The subpolicies collect the following rewards: (i) A sparse positive reward of +10 for finding a target object, (ii) A sparse positive reward of +3 for opening a door, (iii) A penalty of -0.1 per collision for navigation policies, (iv) A negative traveled distance reward to encourage the high-level policy to find efficient compositions of the subpolicies. As the navigation policy does not get physically executed during training, we set it to -0.05 for each invocation of the local exploration policy and to $-0.05 \cdot \text{number of waypoints}$ for the navigation policy. This results in a similar penalty per distance traveled.

V. EXPERIMENTAL EVALUATIONS

We extensively evaluate our approach both in simulation and real-world experiments. We aim to answer the following questions:

- I) Does the high-level policy learn to take decisions that lead to efficient exploration of the environment, improving over alternative decision rules?
- II) What is the impact of the different subpolicies, in particular the local and global exploration policies?
- III) Does the learned behavior transfer to the real world and to execution with different subpolicies?
- IV) Is the overall system capable of successfully solving extended tasks involving many physical interactions within a single episode in the real world?

A. Experimental Setup

We instantiate the task in the iGibson simulator [35]. Each scene contains three cabinets placed randomly across a set of feasible locations. All the doors in the scene are initially in a closed state. We then construct tasks of finding 1-6 target objects, matching the hardest setting in previous work [1], [3]. We randomly place up to three target objects across the free space of the entire apartment and up to three objects inside the cabinets. We use the same eight training scenes as the iGibson challenge and use the remaining seven apartments for evaluation. The embodied agent is a Fetch robot, equipped with a mobile base with a differential drive, a height-adjustable torso, and a 7-DoF arm. We scale the robot’s size by a factor of 0.85 to be able to navigate the narrow corridors of all apartments. Its raw 10-dimensional action space consists of a continuous linear and angular velocity for the base together with the torso and arm-joint velocities. The robot is equipped with an RGB-D camera with a field of view of 79 degrees and a maximum depth of 5.6 m.

B. Baselines

We compare our approach against different high-level decision-making modules and ablations of the action space.

Greedy: A greedy high-level decision-making strategy that immediately drives to any newly mapped task object (door, cabinet, target object) if available and otherwise selects either the local or global exploration policies with equal probability. *SGoLAM+:* SGoLAM [17] 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 and directly use the semantic camera for goal localization which further improves the performance. We then modify the action execution to open doors and cabinets when applicable.

HIMOS: The hierarchical reinforcement learning approach presented in Section IV.

w/o frontier removes global exploration from the subpolicy set.

w/o expl removes local exploration from the subpolicy set.

w/o IAM removes the invalid action masking and instead penalizes the agent with -2.5 for selecting invalid actions.

Metrics: We evaluate the models’ ability to find all the desired objects using the success rate and we evaluate the optimality of the search path with the success-weighted path length (SPL) [36]. In the simulation, we evaluate 25 episodes per scene, the number of target objects, and report the average over three random training seeds. This results in a total of $25 \cdot 6 \cdot 8 \cdot 3 = 3600$ episodes for seen and $25 \cdot 6 \cdot 7 \cdot 3 = 3150$ episodes for unseen apartments for each approach.

C. Simulation Experiments

To test the models’ abilities to learn to complete the tasks, we first evaluate them in the seen apartments for variable numbers of target objects. The results are reported in Table I. We find that all the compared models achieve good success

TABLE I
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	Greedy	93.7	92.6	91.5	91.8	88.3	85.9	90.6
	SGoLAM+	92.2	89.3	88.1	85.6	84.7	83.3	87.2
	w/o frontier	81.1	75.0	69.7	69.4	69.9	64.2	71.6
	w/o expl	93.2	90.8	89.6	90.1	89.6	84.4	89.6
	w/o IAM	95.1	95.1	93.5	92.5	90.8	90.2	92.9
	HIMOS	98.5	98.3	96.3	95.4	93.8	93.3	96.0
SPL	Greedy	44.7	44.6	45.4	45.5	46.2	47.3	45.6
	SGoLAM+	43.3	42.4	44.5	45.4	46.1	47.3	44.8
	w/o frontier	41.0	41.1	41.7	42.5	45.6	45.9	43.0
	w/o expl	43.3	44.5	44.8	44.0	44.9	44.9	44.4
	w/o IAM	46.9	47.7	49.1	46.5	49.4	49.7	48.2
	HIMOS	49.1	50.5	51.4	50.9	53.5	54.2	51.6

TABLE II
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	Greedy	94.8	94.8	92.4	92.3	89.2	87.2	91.8
	SGoLAM+	92.5	89.6	88.5	88.6	88.7	85.8	88.6
	w/o frontier	82.2	78.2	73.8	70.8	69.7	66.5	73.5
	w/o expl	93.4	91.8	89.8	87.8	84.4	83.6	88.5
	w/o IAM	96.5	95.4	94.3	91.4	90.2	88.2	92.7
	HIMOS	97.7	96.9	96.8	96.0	96.0	94.7	96.3
SPL	Greedy	44.6	46.2	47.9	50.3	49.9	50.4	48.2
	SGoLAM+	44.5	43.7	46.2	47.5	48.8	50.0	46.8
	w/o frontier	32.0	38.6	42.7	43.6	44.8	45.2	41.1
	w/o expl	40.9	42.0	46.7	48.7	48.1	48.8	45.9
	w/o IAM	45.9	46.1	45.8	48.1	49.3	49.8	47.5
	HIMOS	52.2	51.5	51.8	55.7	57.4	58.4	54.5



Fig. 3. Example trajectories of HIMOS in unseen apartments. Black: unexplored, blue: free space, green: walls, red: agent trace, grey: (found) target objects, other colors: miscellaneous objects. Bottom right: the agent failed to find the last object, marked by the red circle, in the given time.

rates. All three high-level decision-making variations, greedy, SGoLAM+, HIMOS, are able to make reasonable decisions, demonstrating the benefits of the design of the high-level abstractions discussed in Section IV-C. Furthermore, our proposed method, HIMOS, further improves over the baselines, consistently achieving the highest success rate and the most efficient paths, as measured by the SPL metric.

We then evaluate the models in the unseen apartments. Note

that neither the low- nor high-level policies have seen these scenes during training. The results are shown in Table II. We find that the models learned to generalize without any clear generalization gap. The performance is even slightly higher than on the seen apartments, this is in accordance to previous observations [3]. This may be due to the validation split containing potentially simpler scenes than the training scenes. The evaluations on unseen scenes confirm the observations from the training scenes: HIMOS consistently achieves the highest success rate and the best SPL across all the numbers of target objects. Finally, we find that our hierarchical approach scales very well to longer scenarios, with a very small drop in success rates as the number of target objects increases. To find all six objects, the agent often has to explore the majority of the apartments and interact with a large number of articulated objects.

D. Ablation Study

Exploration Subpolicies: To evaluate the impact of the exploration policies, we compare HIMOS to *w/o frontier* and *w/o expl*. We find frontier exploration to have a large impact on success rates. Removing this component reduces the success rate to 73.5%. Removing the local exploration policy leads to a smaller, but nonetheless significant drop of 7.7 ppt in average success rates as well as a clear drop in SPL. This indicates on one hand the different strengths of the two exploration behaviors, and on the other hand that HIMOS learned to use

TABLE III
REAL WORLD EXPERIMENTS ON THE HSR ROBOT.

Model	1-obj	2-obj	3-obj	4-obj	5-obj	6-obj	Total
Success	5	4	4	3	4	3	23
Collision	0	0	1	0	0	1	2
Interaction failure	0	1	0	0	1	1	3
Navigation failure	0	0	0	2	0	0	2
Total Episodes	5	5	5	5	5	5	30

the local exploration policy to increase search efficiency. Again, we find this effect to be consistent across both seen and unseen apartments.

Invalid Action Masking: We observe that removing the invalid action masking also leads to a drop in both success rates and SPL. Furthermore, we found that masking improves convergence speed by up to 2.5 times. Note that the action masking does not use any further privileged information beyond our perception assumptions (Section IV) of inferring object states.

Qualitatively, we find that the agent learned sensible behaviors for the task at hand. Figure 3 depicts example episodes in the unseen scenes. The agent learned to frequently invoke the local exploration policy while the apartment is still largely unexplored, to then use the global exploration policy to navigate to unexplored corners where target objects could be hidden. Generally, areas further away are being used in order to travel faster to certain areas of interest. The high-level policy also frequently uses frontier-based navigation when the exploration policy is stuck in some area. When a target object lies on the way to another relevant navigation point (a frontier, cabinet, or door), the high-level policy learned to navigate directly to the latter, instead of sequentially navigating to the target object and then proceeding. This behavior saves time and improves efficiency.

E. Real World Experiments

We transfer the trained policy to a *Toyota HSR* robot. The robot has a height-adjustable torso and a 5-DOF arm for environment interactions. It is equipped with an RGB-D camera used for mapping and a 2D lidar employed for localization in the pre-built map and object avoidance by the ROS navigation stack. Both the local exploration policy and the high-level behavior policy are transferred to this real-world setting without any further retraining or fine-tuning. The agent requires only minor adjustments to account for the differences in robot geometry and subpolicies. See Section IV-B and the supplementary material for details on the real-world subpolicies.

The experiments are performed in an office building covering three rooms connected by a hallway with a total of three doors. The operation space covers roughly 180 square meters. We place three articulated objects, two cabinets with a revolute door, and one drawer in the environment. These objects are never seen during training. We evaluate runs with 1-6 target objects in five different scenarios, for a total of 30 episodes. Each scenario defines new positions for the three articulated objects. We randomly chose which doors start in an open or closed state, and start each episode from the room that the

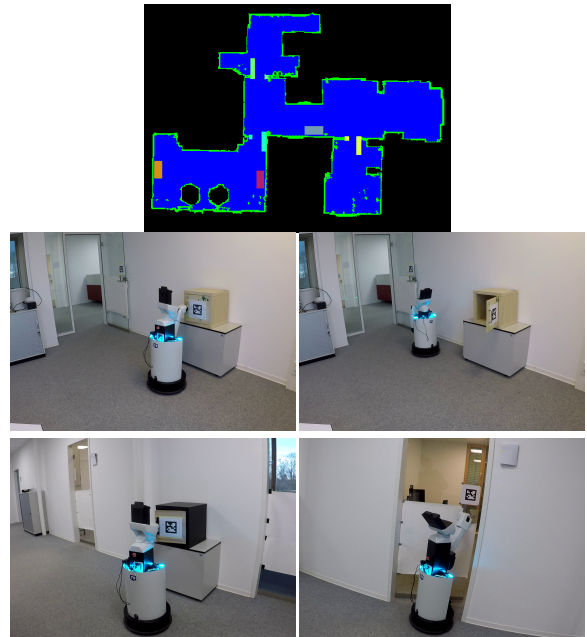


Fig. 4. Top: map of the real world environment. Initial door state and cabinet positions are randomized between episodes. Below: example trajectory in the real world. From top left to bottom right: the agent decides to look inside a target object, then navigates to the hallway, opens a different cabinet and finally opens and drives through a closed door.

last episode terminated in. We cover glass doors to prevent the agent from directly looking through them. This evaluation tests (i) the generalization abilities of both low- and high-level behaviors to the real world, (ii) the generalization abilities of the local exploration policy and high-level policy to a different robot, and (iii) the high-level policy's generalization to unseen subpolicies, as we change both the navigation and manipulation modules. This is an important ability for transfers to different robot models and execution requirements. Lastly, (iv) the map representation enables easy transfer to different objects as it only requires a mapping to known semantic and instance colors.

The results of the experiments are summarized in Table III and example episodes are shown in Figure 4 as well as in the supplementary video. The agent successfully completes 76.7% of the episodes, requiring long sequences of autonomous navigation and physical interactions. The high-level policy proves robust to failures in the subpolicies. These include navigation failures if the planner does not find a valid path to a frontier point and manipulation failures in which the mobile manipulation skill failed to grasp the handle of an articulated object. In this case, the agent is capable of re-triggering the interactions after detecting the failure. A few irrecoverable failures occurred that did not allow the agent to continue: reaching a safety limit of the wrist joint during door opening, base collisions, and in two cases repeated failures of the navigation stack from which the agent was unable to recover.

VI. CONCLUSION

We introduced the interactive multi-object search task in which the agent has to manipulate the environment in order to

fully explore it, resembling common household settings. We proposed a novel hierarchical reinforcement learning approach capable of solving this complex task in both simulation and the real world. By combining a high-level policy on abstract action spaces with low-level robot behaviors, we are able to perform long-term reasoning while acting in continuous action spaces. Our approach decouples the perception from decision making which allows a seamless transition to unknown and real-world environments on a differing embodiment. In extensive experiments, we demonstrated the capabilities of our approach and the importance of the individual components in ablation studies. In future work, we will investigate the benefits of jointly training the high- and low-behavior and integrate more sophisticated mapping modules to build a semantic map directly from the robot sensors. Further, additional low-level behaviors could extend environment interaction options or perform more goal-oriented active perception actions.

REFERENCES

- [1] S. Wani, S. Patel, U. Jain, A. Chang, and M. Savva, "Multion: Benchmarking semantic map memory using multi-object navigation," *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 9700–9712, 2020.
- [2] K. Fang, A. Toshev, L. Fei-Fei, and S. Savarese, "Scene memory transformer for embodied agents in long-horizon tasks," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2019, pp. 538–547.
- [3] F. Schmalstieg, D. Honerkamp, T. Welschhold, 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.
- [4] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. of the IEEE Int. Symp. on Comput. Intell. in Rob. and Aut. (CIRA)*, 1997, pp. 146–151.
- [5] D. Honerkamp, T. Welschhold, and A. Valada, "N²m²: Learning navigation for arbitrary mobile manipulation motions in unseen and dynamic environments," *IEEE Transactions on Robotics*, 2023.
- [6] J. Gu, D. S. Chaplot, H. Su, and J. Malik, "Multi-skill mobile manipulation for object rearrangement," *arXiv preprint arXiv:2209.02778*, 2022.
- [7] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. S. Chaplot, O. Maksymets *et al.*, "Habitat 2.0: Training home assistants to rearrange their habitat," *Advances in Neural Information Processing Systems*, vol. 34, pp. 251–266, 2021.
- [8] T. Chen, S. Gupta, and A. Gupta, "Learning exploration policies for navigation," in *International Conference on Learning Representations*, 2019.
- [9] 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*, 2020.
- [10] C. Chen, U. Jain, C. Schissler, S. V. A. Gari, Z. Al-Halah, V. K. Ithapu, P. Robinson, and K. Grauman, "Soundspaces: Audio-visual navigation in 3d environments," in *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020, pp. 17–36.
- [11] A. Younes, D. Honerkamp, T. Welschhold, and A. Valada, "Catch me if you hear me: Audio-visual navigation in complex unmapped environments with moving sounds," *IEEE Rob. and Automation Letters*, vol. 8, no. 2, pp. 928–935, 2023.
- [12] Y. Qiu, A. Pal, and H. I. Christensen, "Learning hierarchical relationships for object-goal navigation," in *2020 Conference on Robot Learning (CoRL)*, 2020.
- [13] R. Druon, Y. Yoshiiyasu, A. Kanazaki, and A. Watt, "Visual object search by learning spatial context," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1279–1286, 2020.
- [14] E. Beeching, J. Debangoye, O. Simonin, and C. Wolf, "Deep reinforcement learning on a budget: 3d control and reasoning without a supercomputer," in *25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 158–165.
- [15] F. Xia, W. B. Shen, C. Li, P. Kasimbeg, M. E. Tchampi, A. Toshev, R. Martín-Martín, and S. Savarese, "Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments," *IEEE Rob. and Automation Letters*, vol. 5, no. 2, pp. 713–720, 2020.
- [16] S. K. Ramakrishnan, D. S. Chaplot, Z. Al-Halah, J. Malik, and K. Grauman, "Poni: Potential functions for objectgoal navigation with interaction-free learning," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2022, pp. 18 890–18 900.
- [17] J. Kim, E. S. Lee, M. Lee, D. Zhang, and Y. M. Kim, "Sgolam: Simultaneous goal localization and mapping for multi-object goal navigation," *arXiv preprint arXiv:2110.07171*, 2021.
- [18] 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*, vol. 7, no. 2, pp. 3372–3379, 2022.
- [19] A. H. Qureshi and Y. Ayaz, "Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments," *Robotics and Autonomous Systems*, vol. 68, pp. 1–11, 2015.
- [20] D. Honerkamp, T. Welschhold, and A. Valada, "Learning kinematic feasibility for mobile manipulation through deep reinforcement learning," *IEEE Rob. and Automation Letters*, vol. 6, no. 4, pp. 6289–6296, 2021.
- [21] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," *Advances in neural information processing systems*, vol. 31, 2018.
- [22] M. Hutsebaut-Buysse, K. Mets, and S. Latré, "Hierarchical reinforcement learning: A survey and open research challenges," *Machine Learning and Knowledge Extraction*, vol. 4, no. 1, pp. 172–221, 2022.
- [23] J. Krantz, A. Gokaslan, D. Batra, S. Lee, and O. Maksymets, "Waypoint models for instruction-guided navigation in continuous environments," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2021, pp. 15 162–15 171.
- [24] C. Chen, S. Majumder, Z. Al-Halah, R. Gao, S. K. Ramakrishnan, and K. Grauman, "Learning to set waypoints for audio-visual navigation," in *International Conference on Learning Representations*, 2020.
- [25] N. Yokoyama, A. W. Clegg, E. Undersander, S. Ha, D. Batra, and A. Rai, "Adaptive skill coordination for robotic mobile manipulation," *arXiv preprint arXiv:2304.00410*, 2023.
- [26] M. Iovino, E. Scukins, J. Stryud, P. Ögren, and C. Smith, "A survey of behavior trees in robotics and ai," *Robotics and Autonomous Systems*, vol. 154, p. 104096, 2022.
- [27] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics*, 2011.
- [28] R. Mohan and A. Valada, "Amodal panoptic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 21 023–21 032.
- [29] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," in *FLAIRS Conference*, 2022.
- [30] 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, pp. 181–211, 1999.
- [31] S. Yan, J. Zhang, D. Büscher, and W. Burgard, "Efficiency and equity are both essential: A generalized traffic signal controller with deep reinforcement learning," in *Int. Conf. on Intelligent Robots and Systems*, 2020, pp. 5526–5533.
- [32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
- [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*, vol. 22, no. 268, pp. 1–8, 2021.
- [35] C. Li, F. Xia, R. Martín-Martín, M. Lingelbach *et al.*, "igibson 2.0: Object-centric simulation for robot learning of everyday household tasks," in *Conference on Robot Learning*, vol. 164, 2021, pp. 455–465.
- [36] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva *et al.*, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.

Learning Hierarchical Interactive Multi-Object Search for Mobile Manipulation

- Supplementary Material -

Fabian Schmalstieg*

Daniel Honerkamp*

Tim Welschehold

Abhinav Valada



Fig. 1. In order to explore unstructured environments, the agent has to autonomously manipulate the environment which may include opening doors or looking into articulated objects.

In this supplementary material, we provide additional details on the environment setup, the sub-policies as well as details on the training and architecture. Examples of the learned behavior are included in the video material.

I. ENVIRONMENT DETAILS

Figure 1 depicts an example simulation environment.

Doors: Doors leading out of the apartments are set to a locked state and cannot be opened by the agent as they lead to a fall into the abyss.

Cabinet positioning: In the simulation, we first mark feasible areas along the walls of the apartment in which the cabinets can be spawned without blocking doorways or narrow corridors. We then uniformly sample poses from these areas and reject any poses that would result in a collision with the environment.

Target object sampling: During training, we draw six objects randomly with replacements. If the same object gets drawn repeatedly, the resulting object slot is left empty, leading to a distribution over 1-6 target objects. During the evaluation,

the desired number of target objects is drawn uniformly from all target objects. Three out of the six target objects are set to always be placed in a drawer if they get selected. Leading to an average of half the target objects always being placed within a drawer.

II. INTERACTION AND MANIPULATION MOTIONS

As discussed in the assumptions in Section IV, we assume the capability to infer the poses of objects of interest. For doors that can open to either side, this includes knowledge of the direction that they open to. All manipulation motions start with the agent navigating to a given offset point in front of the respective object. The opening motions in the simulation are then implemented in two variations: "magic" actions for fast training and physical execution based on a motion planner during evaluation.

Magic actions: During training, the object joint positions are slowly increased to their maximum collision-free value by the simulator. These magic opening actions can fail with a probability of 15%, in which case the joint positions are left unchanged.

Motion planning: For realistic evaluation, at test time in simulation these motions are replaced with actual execution. The manipulation motions are implemented based on a BiRRT motion planner [19]. The motion planner creates a plan for opening the doors toward a desired direction. It selects a desired end-effector goal given the position and orientation of the object and plans a trajectory towards this goal by sampling collision-free arm joint configurations. Subsequently, the joints are set according to the plan. In addition, the end-effector closes to grasp the door knob. Finally, a push or pull operation manipulates the door by using the desired direction and computing the joint positions with inverse kinematics.

III. NAVIGATION MOTIONS

As discussed in Section IV, the navigation policy moves the agent along waypoints computed by an A* planning algorithm. The algorithm computes the path based on a prior known traversability map with an inflation radius of 0.2 meters. This map is used simply to avoid recomputing the navigation graph at every step. For frontier point selection, the semantic map is first converted to an occupancy map and then convolved with a 5×5 kernel for a single iteration.

*These authors contributed equally.
Department of Computer Science, University of Freiburg, Germany.
Project page: <http://himos.cs.uni-freiburg.de>

TABLE I
HYPERPARAMETERS USED FOR TRAINING.

Parameter	Value	Parameter	Value
clip param	0.1	γ	adaptive
ppo epoch	4	learning rate	0.0005
num mini batch	128	optimizer	Adam
entropy coef	0.005		

IV. REAL-WORLD ADAPTATIONS

In contrast to the Fetch robot that we train in simulation, the HSR has an omnidirectional drive. The pre-trained local exploration policy still executes its commands as pure differential drive motions (sending forward and angular velocity commands). The unseen N^2M^2 mobile manipulation policy and the ROS navigation module can make use of the robot’s omnidirectional movement, as the training procedure is agnostic to their internal workings. To account for differences in the robot geometry, we use a robot-specific inflation radius for the navigation policies and adjust the instance navigation module to select relative navigation goals that are further away from the object instances.

V. TRAINING DETAILS

Network architectures: The coarse map is encoded into a 256-dimensional feature vector with a ResNet-18 [32]. The

fine-grained map is encoded into a 128-dimensional feature vector using a simple three-layer CNN with 32, 64, and 64 channels and strides 4, 2, and 1. The local exploration policy then concatenates the map encodings together with the robot state and processes these features with fully connected layers, following the author’s original architecture [3]. The high-level policy only uses the coarse map encoder architecture (without weight-sharing). Both the local exploration policy and the high-level policy then use an actor and a critic parameterized by a two-layer MLP network with 64 hidden units each and a Categorical policy.

Hyperparameters: Table I lists the main hyperparameters used during training. The agents were implemented based on a public library [34]. Parameters not mentioned were left at their defaults. The $\gamma_{adaptive}$ parameter is calibrated to result in an average discount factor of 0.99 for a high-level policy step. This is done by setting $\gamma_{adaptive}^n = 0.99$ where n is the average subpolicy duration. We set n to 7 for HIMOS and *w/o frontier* and 10 for *w/o expl* based on a average subpolicy lengths in a training run. During training we early terminate the episodes if they exceed 500 steps.

F

EAGERx: Graph-Based Framework for Sim2real Robot Learning

Bas van der Heijden^{*1}, Jelle Luijckx^{*1}, Laura Ferranti¹, Jens Kober¹, Robert Babuska¹

Abstract—Sim2real, that is, the transfer of learned control policies from simulation to real world, is an area of growing interest in robotics due to its potential to efficiently handle complex tasks. The sim2real approach, however, is hampered by discrepancies between simulation and reality, inaccuracies in physical phenomena modeling, and asynchronous control, among others. To this end, we introduce EAGERx, a framework with a unified software pipeline for both real and simulated robot learning. It can support various simulators and aids in integrating state, action and time-scale abstractions to facilitate learning. EAGERx’s integrated delay simulation, domain randomization features, and proposed synchronization algorithm contribute to narrowing the sim2real gap. We demonstrate the efficacy of EAGERx in accommodating diverse robotic systems and maintaining consistent simulation behavior. EAGERx is open source and its code is available at <https://eagerx.readthedocs.io>.

Index Terms—Reinforcement Learning, Software-Hardware Integration for Robot Systems, Machine Learning for Robot Control, Software Tools for Robot Programming

I. INTRODUCTION

Transferring control policies trained in simulation to the real-world, known as sim2real, has gained considerable interest in the field of robotics due to its potential to address complex tasks with remarkable efficiency [1], [2], [3]. Simulations offer a safe, cost-effective, and controlled environment for training and testing robotic algorithms, allowing roboticists to refine their models and controllers without the risks and expenses associated with real-world experimentation. The sim2real approach, however, faces challenges due to the *sim2real gap*, that is, unaccounted discrepancies between simulation and reality. These disparities may stem from *inaccurate modeling of physical phenomena* (e.g., friction, deformations, and collisions), or from the use of *separate software implementations* for reality and simulation, which may lead to unintended mismatches as depicted in Fig. 1. Another subtle but significant source of discrepancy is the *asynchronous nature* of robotic systems [4]. While robotic systems are typically simulated sequentially [5], sensing, computation and acting happen concurrently in reality. Disregarding these differences can be detrimental to the real-world performance of a policy trained in simulation.

Inaccurate modeling of physical phenomena in simulation is typically mitigated by domain randomization [3]. However, this approach can make the simulation more challenging,

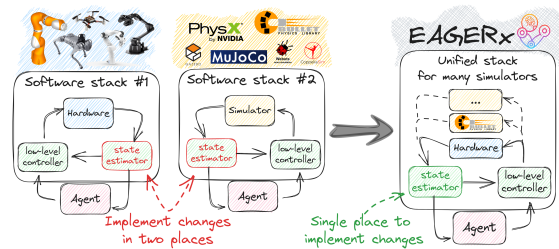


Fig. 1: Our framework offers a unified software pipeline for both simulated and real robot learning. It can support various simulators and aids in integrating state, action and time-scale abstractions.

which may lead to longer training times and suboptimal policies. Reformulating the task to the right level of abstraction may be more effective to alleviate the sim2real gap if the abstraction captures the task and can be extracted accurately both from simulated and real data [6]. Abstractions can take various forms, such as action abstraction that simplifies control issues using high-level actions [7], time-scale abstraction that uses macro-actions for multi-scale planning and learning [8], and state abstraction that condenses raw sensor data into key features [6]. Therefore, existing sim2real frameworks [9], [10], [11] have exploited the multi-rate graph-based design of ROS [12] to obtain a unified software pipeline that allows for the integration of various kinds of abstractions. However, these frameworks restrict users to the Gazebo simulator [13], which can be limiting as different tasks may require specific types of simulators. Additionally, these frameworks fall short in synchronizing components that operate in parallel within the simulation. At faster-than-real-time simulation speeds, this can exacerbate communication and processing delays, leading to inconsistencies, inaccuracies, and potential system instability. Such amplified delays can compromise the proper functioning of the simulated system, rendering learned policies ineffective when transferred to real-world environments. Conversely, naive synchronization may also widen the sim2real gap if it overlooks the concurrent nature of sensing, computation and acting in reality.

The main contribution of this paper is EAGERx (Engine Agnostic Graph Environments for Robotics), that is, a robot learning framework with a unified software pipeline compatible with both simulated and real robots that supports the integration of various abstractions and simulators as depicted in Fig. 1. EAGERx introduces a novel synchronization protocol that coordinates inter-node communication based on node rates and anticipated delays. By simulating delays, our protocol maintains asynchronous robotic

^{*}Equal contribution

¹ Dept. of Cognitive Robotics, Delft University of Technology, The Netherlands. d.s.vanderheijden@tudelft.nl

This work is funded by the EU’s H2020 OpenDR project (grant No 871449) and the Dutch Science Foundation NWO-TTW’s Veni project HARMONIA (18165).

system relationships *synchronously*, preserving the benefits of modular and synchronous simulation. Contrasting with sequential simulation, the protocol permits nodes to transmit messages asynchronously and perform tasks without waiting for immediate responses, thereby accelerating the simulation and allowing nodes to progress based on their processing capabilities and data availability. EAGERx is Python-based and offers high simulation accuracy without compromising speed, native support for domain randomization and delay simulation, and a modular structure for easy manual reset procedures and prior knowledge integration. Our framework features a consistent interface, an interactive GUI, unit tests with code coverage $> 95\%$, and a comprehensive documentation, including interactive tutorials, easing new user adoption. The documentation, tutorials, and our open-source code can be found at <https://eagerx.readthedocs.io>.

In summary, we make four key contributions:

- C1** EAGERx’s modular design supports various robotic systems and state, action and time-scale abstractions.
- C2** EAGERx’s agnostic design allows compatibility with multiple engines.
- C3** Integrated delay simulation and domain randomization in EAGERx can narrow the sim2real gap.
- C4** The proposed synchronization protocol ensures consistent simulation behavior even beyond real-time speeds.

An extensive experimental evaluation (together with the additional documentation) is provided to show the applicability of EAGERx for robotics tasks.

II. FRAMEWORK

This section provides an overview of EAGERx. Sec. II-A outlines the framework’s main components. Then, Sec. II-B discusses the package management system promoting modularity and versioned compatibility. Finally, Sec. II-C discusses the framework’s capabilities for domain randomization, simulator augmentation, and delay simulation, which are essential to minimize the sim2real gap.

A. Agnostic Framework

First, we provide a brief overview of the main components, followed by a code example.

1) *Graph*: EAGERx processes are represented as nodes within a graph structure, linked by directed edges from a node’s output to one or multiple node inputs. Nodes communicate via edges by exchanging messages. This versatile decentralized architecture, ideal for networked hardware and off-board computer interactions, is especially useful for robotics.

2) *Node*: Nodes, representing individual processes running at a fixed rate, are central to EAGERx, executing user-code to compute outputs based on inputs from other nodes. Robot control systems typically consist of multiple nodes (e.g., one node extracts camera images, another localizes using these images, and a final node directs robot movement based on localization). Nodes can be launched in various ways. For example, *I/O-bound nodes* may use separate threads to reduce communication overhead from

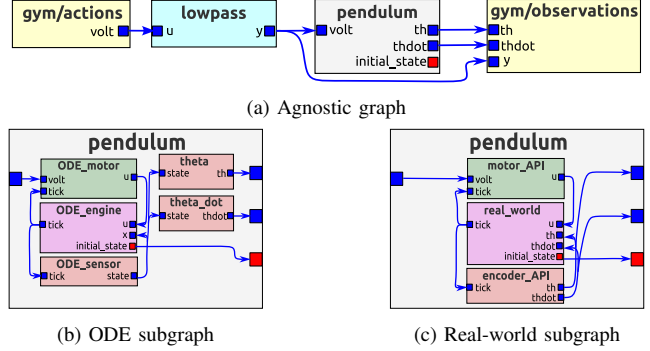


Fig. 2: (a) Displays the engine-agnostic graph of the pendulum environment from Code-Example 1 as generated by the GUI. The engine-specific subgraphs for replacing the object (i.e., pendulum) are depicted for the ODE (b) and real-world (c) engines. The yellow nodes, split for visualization clarity, symbolize the agent’s actions and observations. Blue squares represent I/O channels, while red squares indicate node states and/or parameters that can be randomized at the start of an episode.

message serialization, while *CPU-bound nodes* may utilize subprocesses. Computational loads can be distributed across machines by launching nodes as external processes.

3) *Object*: EAGERx objects enable flexible node replacement when transitioning a robotic system from simulation to reality. For instance, in reality, nodes for extracting sensor data from a physics-engine become obsolete, requiring replacement with nodes interfacing robot hardware EAGERx objects accommodate this adaptability.

Objects define abstract inputs and outputs, as well as subgraphs for each supported physics-engine. Users can add objects to graphs (Fig. 2a), and establish connections between nodes and objects. Upon selecting a physics-engine, abstract objects are replaced by corresponding subgraphs (Fig. 2b, Fig. 2c), rendering the node and object graph *engine-agnostic* (Fig. 2a), as it supports multiple physics-engines. Notice how the framework treats reality as just another physics-engine. Practically, objects represent entities interacting directly with the physical environment. For instance, a robot may have an abstract input and output for its motors and encoders, respectively. Depending on the chosen physics-engine, the robot’s subgraph comprises nodes interfacing real hardware or nodes communicating with a simulator.

4) *Engine*: Physics-engines (e.g., PyBullet [14], Gazebo [13]) are interfaced by a special node called the *engine*. The engine initiates the physics-engine, adds 3D meshes, and sets dynamic parameters (e.g., friction coefficients). It controls time passage and its rate defines the simulation step size.

5) *Backend*: Node processes, launched in various ways (i.e., subprocess, multi-threaded, distributed), communicate through edges and interact with a collective database called the parameter server. The backend facilitates low-level node-to-node communication (i.e., establishing connections and the serialization of messages) for every edge and controls the parameter server. EAGERx supports two backends (i.e., ROS1, SingleProcess), with an abstract backend API allowing users to implement custom backends. Defined graphs can

be initialized as distributed networks of subprocesses or run in a single process. EAGERx provides an abstraction layer over ROS, adding key features for robot learning such as synchronized faster-than-real-time simulation, domain randomization, and delay simulation.

6) *BaseEnv*: EAGERx favors composition over inheritance as a design principle, because robotic systems are more naturally constructed from various components than by finding commonalities and creating a family tree [15]. EAGERx environments consist of an engine, backend, and graph, which is composed of nodes and objects. This design promotes code reuse and handles future requirement changes better than an inheritance-based environment.

```

1 from .tutorials.pendulum import Pendulum # Make object
2 o = Pendulum.make(name="pendulum")
3 from .tutorials.low_pass import LowPass # Make node
4 n = LowPass.make(name="lowpass", rate=15, cutoff=7)
5
6 from eagerx import Graph # Make `agnostic` graph
7 g = Graph.create([o, n])
8 g.connect(action="volt", target=n.inputs.u)
9 g.connect(source=n.outputs.y, target=o.actuators.volt,
10          delay=0.1) # Simulates actuator delay
11 g.connect(source=n.outputs.y, observation="y",
12          skip=True) # Resolves cyclic dependency
13 g.connect(source=o.sensors.th, observation="th")
14 g.connect(source=o.sensors.thdot, observation="thdot")
15
16 from eagerx_ode.engine import OdeEngine # Select engine
17 e = OdeEngine.make(rate=30,
18                  real_time_factor=0, # 0 -> unlimited
19                  sync=True) # toggles synchronization
20 from eagerx.backends.single_process import SingleProcess
21 b = SingleProcess.make() # Make backend
22
23 from .tutorials.env import CustomEnv # Make env
24 env = CustomEnv(g, ode, b, name="env_id", rate=30)
25
26 obs, info = env.reset() # Start a new episode
27 a = env.action_space.sample() # Select an action
28 obs, reward, terminated, truncated, info = env.step(a)
29 env.shutdown() # Release resources

```

Code-Example 1: Environment creation for the swing-up problem.

Code-Example 1 showcases the steps to create an environment using EAGERx for the pendulum swing-up problem, a classic problem in both simulated [5] and real-world robotics scenarios [16]. It begins with the creation of a *pendulum* object and a *lowpass* node to filter the agent’s actions, thereby reducing wear and tear on the system (lines 1-4). Subsequently, an *agnostic* graph is constructed in which the various components are connected, anticipated delays are specified for simulation, and cyclical connections are handled (lines 6-14). The environment is set up with the *OdeEngine* physics-engine and a *SingleProcess* backend (lines 16-21). Equally, the *RealEngine* could be used to switch to real-world scenarios. Following initialization, an interaction is implemented by sampling an action and applying it to the environment (lines 23-28), with the environment being cleanly shut down at the end (line 29).

B. Support

Robotic system design often involves multiple cycles of design, implementation, evaluation, and refinement. EAGERx supports the users as follows:

1) *Visualization Tools*: EAGERx offers interactive visualization tools that aid in understanding and debugging robotic

systems. Users can visualize the graph of nodes and inspect parameter specifications of individual nodes with EAGERx’s interactive GUI. The ability to visualize a complex robotic system is a powerful tool for debugging and understanding the system’s behavior. Example visualizations of the GUI are shown in Figures 2a, 3d, and 4c.

2) *Package Management*: EAGERx incorporates a package management system that fosters modularity, versioned compatibility, and automated unit tests covering 95% of the code. This system allows users to easily share, import, and reuse code modules in different projects. By promoting modular design, EAGERx enables users to build complex robotic systems by combining smaller, well-tested components.

3) *Onboarding Resources*: EAGERx provides comprehensive onboarding resources, including interactive tutorials, code samples, and documentation, to help users quickly learn and adopt the framework.

C. Mitigating the Sim2Real gap

To address the sim2real gap, EAGERx’s modular design enables manual reset routines, simulator augmentation, and supports domain randomization and delay simulation. Resetting simulations is simple, but real-world resets require well-devised routines to return the system to its starting state. Specialized reset nodes can be incorporated into the graph to ease real-world resets between episodes. These nodes are solely used during resets and are absent during an episode. Simulator augmentation, supported by EAGERx’s node structure, increases simulation accuracy and fidelity. Incorporating nodes into physics-engine subgraphs seamlessly enhances simulator capabilities, mimicking real-world phenomena like wind force. Domain randomization varies the simulation by altering parameters like object shapes and lighting [3]. In EAGERx, nodes can register any parameter for randomization. Delay simulation is enabled by our synchronization protocol discussed in II-B, and emulates communication latency and computational delays encountered in real-world systems, yielding a more accurate simulation. Delays can be implemented across any graph edge, encompassing edges between nodes and objects, thus simulating sensor and actuator delays as demonstrated in Code-Example 1, line 10.

III. SYNCHRONIZATION

Parallel computation, used in robotic system simulations via ROS [12] in existing sim2real frameworks [9], [10], [11], can increase simulation speeds. However, these frameworks, when run at faster-than-real-time speeds, suffer from unsynchronized parallel components, unintentionally widening the sim2real gap. Here, the individual computation delays become more pronounced relative to the accelerated simulation clock. Without suitable synchronization at high speeds, certain components may struggle to match pace and gradually fall out of sync, leading to a deviation in the simulation from its real-world counterpart. Consequently, the learned control policy’s performance may deteriorate, as it could receive outdated or mismatched observations, yielding actions based

on inaccurate data. This may render the learned policies ineffective when transferred to the real-world environment.

A. Protocol

We developed a synchronization protocol for each of the nodes representing the robotic system that enables parallel computation and minimizes additional message-passing overhead, thereby enhancing system efficiency and accuracy. This is particularly apt for robotic systems represented as node-based graphs that benefit from parallel operation of multiple nodes. Properly constructed communication patterns and protocols can achieve global synchronization, whereby each node proceeds with its tasks once necessary input data or conditions have been satisfied.

Each node runs a local protocol version, depicted in Alg. 1. The conditions for a node to proceed with the next callback are based on the expected ordering of events, as dictated by assumed rates and delays of the system (lines 5-9). Executed with an event loop thread and dedicated input channel threads, the protocol compares received and expected message counts for input channels before executing subsequent callbacks (line 10-13). This comparison informs whether a node proceeds with the next callback or awaits more messages. Nodes perform tasks based on the protocol's decision and asynchronously transmit output to connected nodes (line 14). Only upon completion of the previous callback or receipt of a new input channel message does the event-driven protocol evaluate conditions for the subsequent callback (line 17). Consequently, task execution is entirely independent of any global clock or synchronization messages, thus minimizing additional message-passing overhead.

The protocol computes expected messages per input channel with node n executing its callback at rate f_n and receiving messages at rate f_i delayed by τ_i over input channels $i \in \mathcal{U}$ as summarized by Alg. 2. Assuming nodes maintain their rates, callbacks occur every $\Delta t_n = \frac{1}{f_n}$ seconds, and messages are received every $\Delta t_i = \frac{1}{f_i}$ seconds. The protocol expects the k th callback after $k\Delta t_n$ seconds, anticipating $\lfloor (k\Delta t_n - \tau_i)/\Delta t_i \rfloor$ messages from each input channel i , where $\lfloor a/b \rfloor$ denotes the integer division operator. While this intuition underpins the synchronization protocol, the implementation in Alg. 2 is more complex. Computations are recast in rates to improve numerical stability. The protocol sets every input channel's initial expected message count to 1, irrespective of τ_i , simplifying callback implementations.

The protocol also handles the special case of cyclical dependencies—common in robotics systems interacting with a physc-engine and can cause deadlocks otherwise—with Alg. 3. In EAGERx, users can designate input channels as cyclical, postponing dependency to the next callback. This strategy allows one node to execute first in a cycle, while others await this node's output.

B. Limitations

The protocol's limitations should be considered in the context of the underlying communication protocol, which must ensure preservation of message order and be lossless

Algorithm 1: Synchronization protocol executed by each node

Input: node rate f_n , input rates f_i , input delay τ_i , input channels $i \in \mathcal{U}$, output channels $j \in \mathcal{Y}$
Output: Processed data sent to downstream nodes
1 $k \leftarrow$ Initialize callback index to 0
2 $B_i \leftarrow$ Initialize empty buffers for every input channel i
3 Start eventLoopThread
4 Start inputChannelThread for every $i \in \mathcal{U}$
5 **eventLoopThread:**
6 **foreach** $i \in \mathcal{U}$ **do**
7 **if** channel i is cyclical **then**
8 $\delta_i \leftarrow$ Expected message count (Alg. 3)
9 **else**
10 $\delta_i \leftarrow$ Expected message count (Alg. 2)
11 **if** $\delta_i \leq \text{size}(B_i)$ for every $i \in \mathcal{U}$ **then**
12 **foreach** $i \in \mathcal{U}$ **do**
13 $u_{i,k} \leftarrow$ Pop last δ_i messages from B_i
14 $y_k \leftarrow$ Run callback with inputs $u_{i,k}, \forall i \in \mathcal{U}$
15 Send y_k to all output channels $j \in \mathcal{Y}$
16 $k \leftarrow$ Increment callback index to $k + 1$
17 Trigger event on eventLoopThread
18 WaitForEvent
19 **inputChannelThread i:**
20 $B_i \leftarrow$ Buffer received message
21 Trigger event on eventLoopThread

Algorithm 2: Expected number of messages to receive between the $k - 1$ th and k th callback

Input: callback index k , node rate f_n , input rate f_i , input delay τ_i
Output: Expected number of messages δ to receive between the $k - 1$ th and k th callback
1 **if** $k = 0$ **then**
2 $\delta \leftarrow 1$ // Set initial count to 1
3 **else**
4 /* Expected count between $k - 1$ and k */
5 $N_{k-1} \leftarrow \lfloor (f_i(k-1) - f_n f_i \tau_i) / f_n \rfloor$
6 $N_k \leftarrow \lfloor (f_i k - f_n f_i \tau_i) / f_n \rfloor$
7 $\Delta \leftarrow N_k - N_{k-1}$
8 /* Correct expected count with delay */
9 $c \leftarrow \lfloor (f_i k - f_n \Delta - f_n f_i \tau_i) / f_n \rfloor$
10 $\delta \leftarrow \Delta - \min(\Delta, \max(0, -c))$

(e.g., TCP instead of UDP [17]). The protocol assumes that the robotic system can be represented by nodes with fixed rates and at least one input. Although the protocol can be easily toggled between synchronous and asynchronous modes, it does not allow for a hybrid mode, where some nodes are synchronized and others are not. Finally, the protocol does not account for jitter and assumes deterministic delay; however, this limitation can be mitigated by varying the delay across episodes if needed.

IV. EXPERIMENTAL EVALUATION

This section presents experiments to show the capabilities of our framework and to support the four key contributions **C1-C4** presented in Sec. I.

A. Experimental Setup

EAGERx is validated using a pendulum swing-up and a vision-based box pushing task. These tasks validate **C1-C2**, involving distinct types of systems like pendulum and

Algorithm 3: Expected number of messages to receive between the $k - 1$ th and k th callback to resolve a cyclical dependency

Input: callback index k , node rate f_n , input rate f_i , input delay τ_i , fudge factor $\epsilon \approx 10^{-9}$

Output: Expected number of messages δ to receive between the $k - 1$ th and k th callback

```

1 if  $k = 0$  then
2    $\delta \leftarrow 0$  // Set initial count to 0
3 else
4   /* Calculate count as if  $k$  is shifted */
5   if  $f_n > f_i$  then
6      $o \leftarrow \lfloor (f_n - \epsilon) / f_i \rfloor$  // Forward
7   else
8      $o \leftarrow -1$  // Backward
9   /* Expected count between  $k - 1$  and  $k$  */
10   $N_{k-1} \leftarrow \lfloor (f_i(k - 1 + o) - f_n f_i \tau_i) / f_n \rfloor$ 
11   $N_k \leftarrow \lfloor (f_i(k + o) - f_n f_i \tau_i) / f_n \rfloor$ 
12   $\Delta \leftarrow N_k - N_{k-1}$ 
13  /* Correct expected count with delay */
14   $c \leftarrow \lfloor (f_i k - f_n(\Delta - 1) - f_n f_i \tau_i) / f_n \rfloor$ 
15   $\delta \leftarrow \Delta - \min(\Delta, \max(0, -c))$ 

```

manipulator robots, and different *engines*. The box-pushing task supports **C1**, using state and action abstractions for safe, efficient learning. The pendulum swing-up task is a sensitive nonlinear control problem, that validates **C3** by assessing delay simulation for effective sim2real. To validate **C4**, we experimentally assess Alg. 1 and employ it in accelerated, parallelized training for both tasks. In selecting these control problems, we favored those frequently used as benchmark tasks in the RL literature [5], [18], [19]. The experiments involved training agents in simulation using the soft actor-critic [20] algorithm from Stable Baselines3 [21] (with hindsight experience replay [22] for box pushing) and performing zero-shot evaluations on their real counterparts. Domain randomization and delay simulation values were kept constant throughout training episodes and selected from uniform distributions.

Inverted Pendulum The inverted pendulum task addresses the classic control problem of swinging up and stabilizing an underactuated pendulum. We used two simulators: the *Pendulum* environment from OpenAI Gym Classic Control [5] with modified parameters (Fig. 3a), and a set of ODEs (Fig. 3b). For a detailed understanding of the dynamics and parameters, readers are directed to [16]. The parameters were obtained using grey-box estimation, and fit based on data derived from a real pendulum system (see Fig. 3c). This real-world pendulum setup, a mass attached to a disk actuated by a DC motor, is used for the zero-shot evaluations.

Box Pushing We performed a box pushing experiment using simple hardware, such as a Viper 300x robotic manipulator (see Fig. 4b) and a consumer-grade webcam (Logitech C170) for ArUco marker detection to obtain the box’s position and orientation. The goal here is to push the box to a fixed goal configuration from varying start configurations. For evaluation, we selected six unique initial configurations (three positions approximately 30 cm from the goal for both a yaw angle of 0 and $\frac{\pi}{2}$ rad) and repeated them thrice per policy. Training was performed in PyBullet (see Fig. 4a).

B. Analysis

C1 To support the claim that the toolkit accommodates various robotic systems, both tasks involve two distinct robot systems, namely a pendulum system and manipulator. Additional demonstrations that use a quadruped and quadrotor are made publicly available. EAGERx’s graph-based design, enabling diverse abstractions, is demonstrated in the vision-based box pushing task. Rather than end-to-end training on raw images, an *aruco detector* is used for state abstraction as depicted in Fig. 4e, negating the need for photorealistic rendering. Action abstractions, visible in Fig. 4c, include an *inverse kinematics* node for task-space learning and a *safety filter* correcting hazardous commands. Nodes set at optimal rates ensure efficient resource use and learning. Finally, the pendulum task underlines the toolkit’s modularity using a *angle reset* node, visible in Fig. 3d, to position the pendulum at the initial angle via PID control, before a new episode.

C2 To support the claim that EAGERx is compatible with a variety of physics-engines and the real-world, we conducted experiments with four different *engines*—PyBullet [14], OpenAI Gym Classic Control [5], real-world, and simulations with sets of ODEs—showing the ability to switch between real and simulated counterparts. The box pushing task demonstrates how a division of the graph into engine-specific and engine-agnostic subgraphs resulted in a unified pipeline between PyBullet and reality. The *inverse kinematics* and *safety filter* nodes work with any simulator, as seen in the agnostic graph (Fig. 4c), while the *aruco detector* and *webcam* nodes are swapped with PyBullet-specific nodes in Figures 4d and 4e. Likewise, the agnostic graph in Fig. 3d was used in all pendulum experiments to display sim2real transfer across physics-engines.

C3 We show that the integrated delay simulation and domain randomization features can reduce the sim2real gap by demonstrating that the negative impacts of actuator delay can be counteracted using the delay simulation feature during training for two different simulated versions of the pendulum. In this task we supported **C3** by evaluating policies on the real system with an actuator delay set at the smallest value that led to a breakdown in baseline performance. When we progressively increased the actuator delay, it resulted in baseline policy failure for delays of 0.025,s and 0.035,s for the Gym and ODE pendulum, respectively. Our experiments studied the potential of training with domain randomization and/or delay simulation to mitigate the adverse effects of the actuator delay. For the Gym pendulum, we applied randomization within $\pm 10\%$ of the mean values (0.033,kg for mass and 0.1,m for length). For the ODE pendulum, randomization was limited to $\pm 5\%$, considering the higher accuracy of this model. Delay simulation involved randomization within ± 0.005 s around the set actuator delay. The results shown in Fig. 3e suggest that delay simulation can mitigate the adverse effects of actuator delay for zero-shot transfer from both the Gym and ODE simulator to the real pendulum system. In the ODE scenario, adding domain randomization to delay simulation further improved performance and resulted

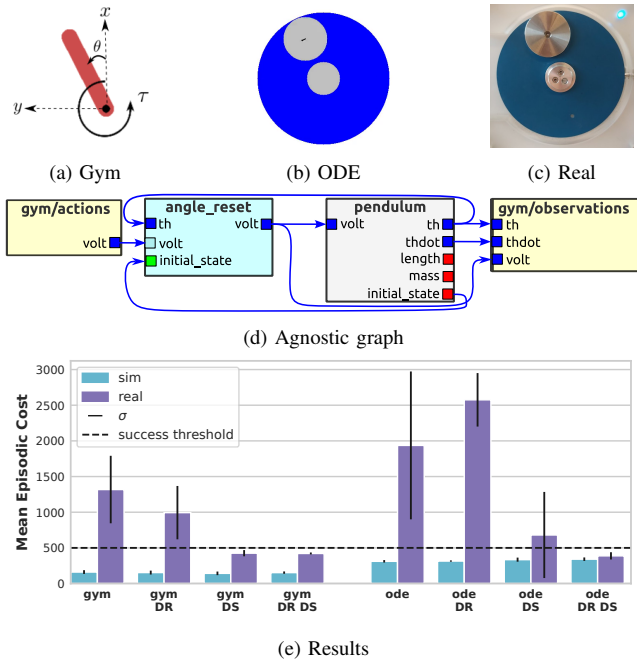


Fig. 3: Results (e) for the Pendulum (a)-(c), represented by the agnostic graph (d). The results show mean episodic cost for 5 policies (10 episodes per policy) and the impacts of domain randomization (DR) and delay simulation (DS). The *success threshold* indicates 100% success rate, meaning successful pendulum swing up and stabilization each episode for all evaluations below this threshold.

in successful transfer with the smallest performance gap between simulation and reality. The effectiveness of domain randomization is further highlighted in the box pushing task (Fig. 4f). We examined its impact through altering the box’s friction coefficient between 0.1 and 0.4. Fig. 4f shows that, compared to the baseline, friction randomization reduces the performance gap between simulation and reality, despite lowering overall performance, thereby illustrating that relying solely on domain randomization can increase task difficulty. Conversely, the incorporation of the inverse kinematics node combined with friction randomization enhances performance, while also reducing the gap between simulation and real-world execution.

C4 The claim that Alg. 1 ensures consistent simulation behavior when exceeding real-time speeds is supported by simulating the pendulum system described in Code-Example 1 and driving it with an identical action sequence for various real-time factors. The real-time factor is defined as the ratio between the simulation time and the real-world time, so a real-time factor of 1 means that the simulation runs in real-time. If a target real-time factor is set too high, some components in the simulation may struggle to match pace and start to lag behind. The effects of this lag in unsynchronized parallel operation within a simulation is demonstrated in Figures 5a and 5b. Fig. 5a shows the variation in angle $\sin(\theta)$ at $t = 2.0$ s over 5 runs of a simulated pendulum as a function of the real-time factor. The pendulum is driven by an identical feedforward voltage sequence over episodes.

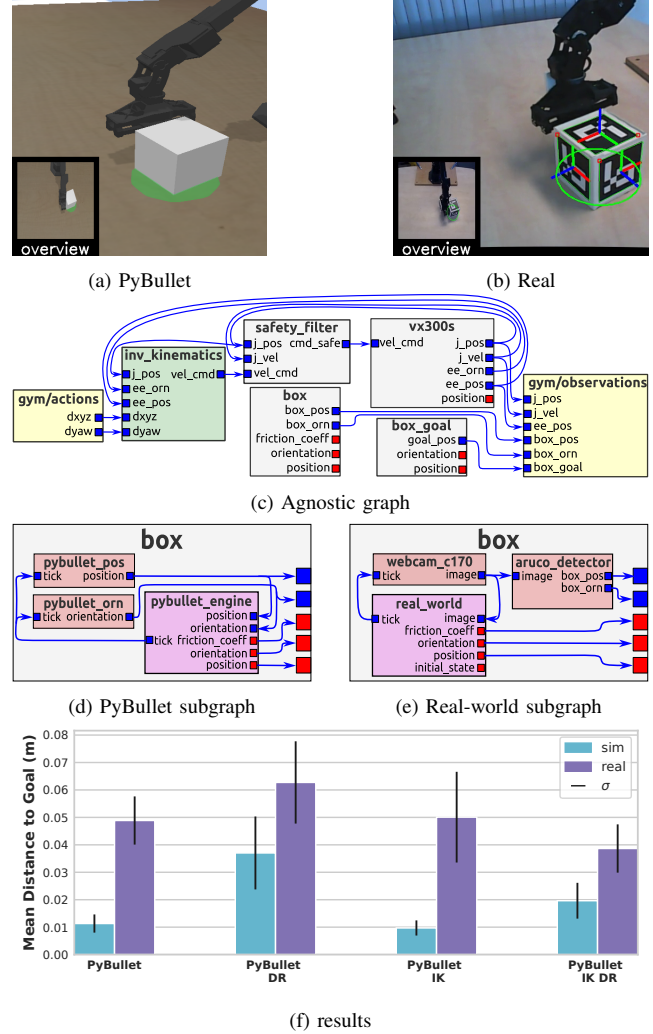


Fig. 4: Results (f) for the Box Pushing (a)-(b), represented by the agnostic graph (c). The engine-specific subgraphs for replacing the *box* object are depicted for the Pybullet (d) and real-world (e) engines. The results show mean distance from the goal at the end of 16 episodes for 3 policies and evaluate the benefits of an inverse kinematics (IK) node (c) (facilitating task space control) and domain randomization (DR) of the friction coefficient.

Commands and angle measurements are sent and received asynchronously. The figure illustrates the increasing variability in angle $\sin(\theta)$ differences with respect to a synchronized simulation. In contrast, a simulation synchronized with our protocol remains fully deterministic regardless of the real-time factor.

Fig. 5b presents the real-time factor of the simulation. The realized real-time factor for synchronous simulations plateaus naturally to maintain synchronization. However, the unsynchronized simulations exhibit a higher realized real-time factor, which is misleading. As shown in Fig. 5a, the increased speed comes at the expense of greater variability. Consequently, parallel components become increasingly out of sync, even if they achieve their target real-time factor individually. These figures collectively demonstrate the detri-

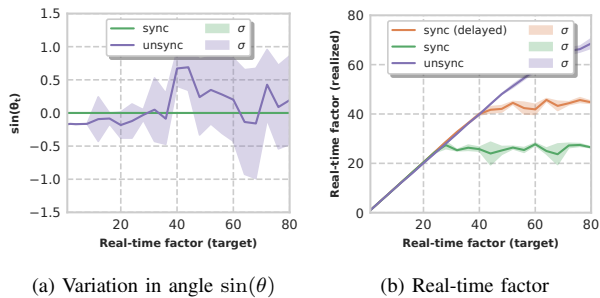


Fig. 5: A comparison between unsynchronous (*unsync*) and synchronous (*sync*) simulations of a pendulum at faster-than-real-time speeds. Fig. 5a shows the variation in angle $\sin(\theta)$ at $t = 2.0$ s over 5 runs of a simulated pendulum as a function of the real-time factor. Fig. 5b shows the realized real-time factor of the simulation for both synchronous and asynchronous cases.

mental effects of unsynchronized parallel operation within a simulation when combined with faster-than-real-time speeds.

The protocol, designed for robotic system synchronization, does not necessitate synchronous communication within the system. In fact, asynchronous communication permits nodes to transmit messages and perform tasks without waiting for immediate responses, thereby accelerating the simulation and allowing nodes to progress based on their processing capabilities and data availability, as Fig. 5b illustrates. We introduced a simulated delay between the pendulum actuator and the physics-engine. Consequently, the pendulum’s callback and physics-engine’s callback can be executed concurrently, as the physics-engine’s callback relies on the pendulum’s output from the previous timestep rather than the current one. Since each node’s protocol operates independently, this parallelization occurs naturally, resulting in approximately 50% increase in the realized real-time factor for the synchronized simulation compared to the case without delay.

V. DISCUSSION

Comparing EAGERx with ROS [12] might seem natural due to their modular structures and asynchronous communication. Nonetheless, such a comparison risks being misleading since EAGERx represents an abstraction based on the actor model [23] and can operate atop a backend like ROS. This abstraction layer offers vital functionality for robot learning, including synchronized faster-than-real-time simulation, domain randomization, and delay simulation, not inherently supported by ROS. Recent research [24] presented a reactive solution to ROS’s asynchronous programming challenges via an event-driven API, inspiring EAGERx’s synchronization approach. However, this API didn’t specifically aim to synchronize simulations using expected rates and delays, as demonstrated in our work. Importantly, EAGERx’s protocol extends beyond ROS to other backends as well.

The proposed synchronization protocol can be seen as an application of the actor model for computation [23]. It is a powerful and flexible model of concurrent computation where actors, the primary units, execute tasks concurrently and communicate by exchanging messages. The actor model

	EAGERx	[29]	[30]	[9]	[10]
Engine Agnostic	✓	✗	✗	—	✗
Specialized Reset Procedures	✓	✗	✗	✗	✗
Unified Pipeline Sim/Real	✓	—	—	—	—
Synchronized Simulation	✓	✓	✓	✗	✗
Distributed Computing	✓	✓	✓	✓	✓
GPU Accelerated	✗	✓	✗	✗	✗
Gradient Information Available	✗	✗	✓	✗	✗
Domain/Delay Randomization	✓/✓	✓/✗	✓/✓	✗/✗	✗/✗
Environment Visualization	✓	—	✓	—	—
Open Source / License-free	✓/✓	—/—	✓/✓	✓/✓	✓/✓
Documentation / Tutorials	✓/✓	✓/✓	✓/✓	—/✗	✗/✗
Last commit (age)	< 1 week	2 months	< 1 week	1 year	4 years

TABLE I: A comparison of various modular sim-to-real robot learning frameworks, where — indicates partial feature presence.

is well-suited for synchronizing robotic systems represented as graphs of nodes, where various nodes need to operate concurrently. Our protocol operates on an event-driven basis and circumvents dependence on a global/local clock, a central coordinator [25], or extra synchronization messages [26]. Instead, it assesses conditions for subsequent callbacks exclusively after finalizing the preceding one or obtaining a new input channel message. This can outperform busy-waiting techniques (or spinlock) [27] that continuously evaluate conditions at a fixed time interval.

Ptolemy II [28] constitutes a software framework for designing, modeling, and simulating heterogeneous systems. Like EAGERx, it applies the actor model of computation, enabling concurrency and asynchronous communication. Both frameworks offer graphical user interfaces for visualizing complex systems. Ptolemy II holds an advantage over EAGERx in its support for a wider range of computation models and the ability to combine them within a single system. Nevertheless, Ptolemy II serves as a general-purpose framework, while EAGERx specifically targets robot learning. Furthermore, Ptolemy II employs a Java-based structure, in contrast with EAGERx’s exclusive use of Python.

In comparison to Gym [5] — which offers a flexible API but lacks a unified sim2real framework — EAGERx addresses this deficiency. Unlike Gym’s default sequential simulation, EAGERx supports concurrent, distributed operations across devices within environments, enhancing its applicability to robot learning. Gym environments use object-oriented classes, frequently constructed via inheritance and extended with wrapper patterns. However, such object-oriented design in Gym environments with wrappers is ill-suited for complex robotic systems, as it can result in an unwieldy proliferation of classes and wrappers. Additionally, incorporating time-abstraction within Gym environments is challenging, often confining it to multiples of the environment’s step size. Conversely, EAGERx allows each node within the graph environment to operate at separate frequencies.

Various robot learning frameworks with connections to EAGERx have been introduced in the field. Among these, Isaac Orbit [29] and Drake [30] stand out as recent frameworks with shared design principles. In line with EAGERx, Orbit and Drake adopt a modular approach to constructing robot environments, enabling the execution of different nodes at varying rates to support both lower and higher level control for effective robot learning. However, these frameworks

exhibit three critical differences with EAGERx. Firstly, EAGERx is designed to be engine-agnostic, whereas Orbit relies on the proprietary Isaac Sim [31] simulator, and Drake incorporates an integrated multi-body dynamics simulator, hence restricting them to a single simulation platform. Secondly, EAGERx features dedicated reset procedures in the form of reset nodes. These nodes can be added to the *graph* and are only activated during environment resets. Thirdly, EAGERx offers a unified pipeline for both simulation and reality. Although Orbit and Drake promote component reusability in both simulation and reality, EAGERx enforces this more rigorously through *engine-agnostic* and *engine-specific graphs*. This effectively isolates the engine-agnostic code and minimizes the risk of discrepancies. Additional frameworks such as Robo-Gym [9] and Gym-Gazebo(2) [11], [10] aimed to exploit the node structure of ROS for robot learning and were primarily centered around the Gazebo simulator without synchronization. A comparative summary of the discussed robot learning frameworks is presented in Tab. I.

VI. CONCLUSION

This paper presented EAGERx, a novel framework to facilitate the transfer of robot learning policies from simulation to the real-world. Our unified framework is compatible with simulated and real robots, supporting various abstractions and simulators. The presented synchronization protocol simulates delays without sacrificing simulation speed or accuracy, enabling effective policy training in simulation and subsequent transfer to real robots. We evaluated our framework on two benchmark robotic tasks, demonstrating its effectiveness in reducing the sim2real gap.

REFERENCES

- [1] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning," in *Proc. of the Conf. Robot Learning (CoRL)*, ser. Proceedings of Machine Learning Research, vol. 164. PMLR, 08–11 Nov 2022, pp. 91–100. [Online]. Available: <https://proceedings.mlr.press/v164/rudin22a.html>
- [2] J. Kooi and R. Babuska, "Inclined Quadrotor Landing using Deep Reinforcement Learning," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 2361–2368.
- [3] 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*, 2018.
- [4] 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," *Intl. Journal of Robotics Research (IJRR)*, vol. 40, no. 4-5, pp. 698–721, 2021.
- [5] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv preprint*, 2016.
- [6] S. Höfer et al., "Sim2Real in Robotics and Automation: Applications and Challenges," *IEEE trans. on Automation Science and Engineering*, vol. 18, no. 2, pp. 398–400, 2021.
- [7] D. Kortenkamp, R. Simmons, and D. Brugali, "Robotic systems architectures and programming," *Springer Verlag*, pp. 283–306, 2016.
- [8] D. Precup, *Temporal abstraction in reinforcement learning*. University of Massachusetts Amherst, 2000.
- [9] M. Lucchi, F. Zindler, S. Mühlbacher-Karrer, and H. Pichler, "robogym – An Open Source Toolkit for Distributed Deep Reinforcement Learning on Real and Simulated Robots," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.
- [10] N. Lopez, Y. Leire, E. Nuin, E. Moral, L. Usategui, S. Juan, A. Rueda, M. Vilches, R. Kojcev, and A. Robotics, "gym-gazebo2, a toolkit for reinforcement learning using ROS 2 and Gazebo," *arXiv preprint*, 3 2019. [Online]. Available: <https://arxiv.org/abs/1903.06278v2>
- [11] I. Zamora, N. Lopez, V. Vilches, and A. Cordero, "Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo," *arXiv preprint*, 2016.
- [12] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, vol. 3. Kobe, Japan, 2009, p. 5.
- [13] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, vol. 3, 2004, pp. 2149–2154 vol.3.
- [14] E. Coumans and Y. Bai, "PyBullet, a Python module for physics simulation for games, robotics and machine learning," <http://pybullet.org>, 2016–2021.
- [15] E. Freeman, E. Robson, B. Bates, and K. Sierra, *Head First Design Patterns: A Brain-Friendly Guide*. "O'Reilly Media, Inc.", 2004.
- [16] E. Derner, J. Kubalik, N. Ancona, and R. Babuska, "Constructing parsimonious analytic models for dynamic systems via symbolic regression," *Applied Soft Computing*, vol. 94, p. 106432, 2020.
- [17] G. Xylomenos and G. Polyzos, "TCP and UDP performance over a wireless LAN," in *Proc. of the IEEE Conf. on Computer Communications (INFOCOM)*, vol. 2. IEEE, 1999, pp. 439–446.
- [18] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, and P. Welinder, "Multi-goal reinforcement learning: Challenging robotics environments and request for research," *arXiv preprint*, 2018.
- [19] S. James, Z. Ma, D. R. Arrojo, and A. Davison, "RLBench: The Robot Learning Benchmark & Learning Environment," *IEEE Robotics and Automation Letters (RA-L)*, 2020.
- [20] 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," *arXiv preprint*, 2019.
- [21] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal on Machine Learning Research (JMLR)*, 2021.
- [22] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, OpenAI, and W. Zaremba, "Hindsight experience replay," in *nips*, 2017.
- [23] C. Hewitt, P. Bishop, and R. Steiger, "Session 8 formalisms for artificial intelligence: a universal modular actor formalism for artificial intelligence," in *Advance Papers of the Conference*, vol. 3. Stanford Research Institute Menlo Park, CA, 1973, p. 235.
- [24] H. Larsen, G. van der Hoorn, and A. Wąsowski, *Reactive Programming of Robots with RxROS*, ser. Studies in Computational Intelligence. Springer Verlag, 2021, vol. 6, pp. 55–83.
- [25] M. Raynal, *Concurrent programming: algorithms, principles, and foundations*. Springer Verlag, 2013.
- [26] D. G. Messerschmitt, "Synchronization in digital system design," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 8, pp. 1404–1419, 1990.
- [27] T. Anderson, "The performance of spin lock alternatives for shared-memory multiprocessors," *IEEE trans. on Parallel and Distributed Systems*, vol. 1, no. 1, pp. 6–16, 1990.
- [28] C. Ptolemaeus, *System design, modeling, and simulation: using Ptolemy II*. Ptolemy.org Berkeley, 2014, vol. 1.
- [29] M. Mittal et al., "Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments," *IEEE Robotics and Automation Letters (RA-L)*, pp. 1–8, 2023.
- [30] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," <https://drake.mit.edu>, 2019.
- [31] NVIDIA, "NVIDIA Isaac Sim," 2022. [Online]. Available: [\url{https://developer.nvidia.com/isaac-sim}](https://developer.nvidia.com/isaac-sim)

G

Prioritizing States with Action Sensitive Return in Experience Replay

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 Experience replay for off-policy reinforcement learning has been shown to improve
 2 sample efficiency and stabilize training. However, typical uniformly sampled
 3 replay includes many irrelevant samples for the agent to reach good performance.
 4 We introduce Action Sensitive Experience Replay (ASER), a method to prioritize
 5 samples in the replay buffer and selectively model parts of the state-space more
 6 accurately where choosing sub-optimal actions has a larger effect on the return.
 7 We experimentally show that this can make training more sample efficient and that
 8 this allows smaller parametric function approximators – like neural networks with
 9 few neurons – to achieve good performance in environments where they would
 10 otherwise struggle.

11 1 Introduction

12 Reinforcement learning aims to find a policy
 13 that maximizes cumulative reward. This is of-
 14 ten done through learning a compact represen-
 15 tation of the value of states or state-action pairs
 16 in an environment by interacting with it. How-
 17 ever, environment interactions can be costly and
 18 therefore necessitate efficient data use. Lin’s
 19 introduction of experience replay in reinforce-
 20 ment learning [13] promotes efficient use and
 21 reuse of collected experience. A replay buffer
 22 stores transition samples from environment in-
 23 teractions, aiding sample efficiency and stabi-
 24 lizing training [14, 15]. Typically, learning of
 25 the value function in off-policy reinforcement
 26 learning is approached like a supervised learn-
 27 ing problem where stochastic gradient descent
 28 is used to train function approximations such as
 29 neural networks. In supervised learning, data is
 30 usually sampled uniformly, however, for imbal-
 31 anced datasets there are techniques to change
 32 the sampling distribution [7]. With reinforce-
 33 ment learning, we have a similar issue as not every
 34 sample is equally relevant for the performance of
 35 the agent. Lambert et al. [12] showed that in model-based reinforcement learning, a model that better
 36 explains the transition data (with higher likelihood) does not necessarily allow a better policy to be
 found on it. There is an objective mismatch when training. This is also the case for a value function

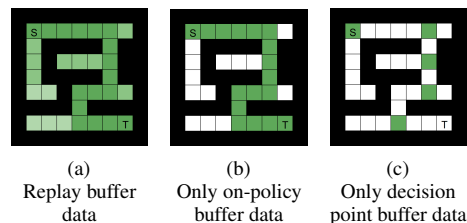


Figure 1: Toy maze environment where an agent moves from start S to termination T. The agent can choose which adjacent square to move to but can only move back to the square where it came from if there is no other valid action. The states sampled from a buffer of transitions are shown in green. After convergence, with normal experience replay the buffer will likely be filled with states from the whole maze. Previous work focused function approximator expressiveness on relevant states by prioritizing on-policy data. We introduce a method to prioritize “decision points” for the agent to further focus this expressiveness.

Submitted to 16th European Workshop on Reinforcement Learning (EWRL 2023). Do not distribute.

37 in off-policy reinforcement learning with function approximators. A lower value function error on the
38 contents of a uniformly sampled buffer does not necessarily mean a better policy will result from it.

39 When using localized representations for the value network, such as in tabular reinforcement learning,
40 replaying these irrelevant samples leads to computational overhead, but it does not impact the accuracy
41 in the vital parts of the state-action space. However, many popular off-policy algorithms use global
42 parametric function approximators like neural networks instead. Updates in irrelevant parts of the
43 state-action space then yield a global effect, as these are often trained with stochastic gradient descent
44 and minimize the mean square error across a batch of experiences [25]. Particularly in the case of
45 smaller function approximators, such as neural networks with few neurons, the limited expressiveness
46 that could have been allocated to model performance-critical areas is spent on inconsequential regions
47 of the state-action space.

48 Off-policy algorithms may therefore, despite their ability to learn from experiences under different
49 policies, still be influenced by the sampling distribution and buffer content when using global function
50 approximators. To illustrate this, consider an extreme case: an agent would not see any performance
51 improvement if it is trained using a replay buffer that contains states that the agent would never
52 come across during rollouts. Other works have limited replay of states and/or actions unlikely under
53 the current policy, the “off-policyness”, which will reduce the replay of irrelevant samples and can
54 improve performance [17, 23, 24]. However, just increased sampling of on-policy data does not
55 prioritize what is really relevant: the importance of modeling these experiences accurately for the
56 agent’s performance during evaluation. A toy example that shows how we would ideally focus the
57 approximators’ expressiveness is shown in Fig. 1. In this example, an optimal policy could be learned
58 with data from only 5 states while it is likely the replay buffer will contain data from all 30 states.

59 In this paper, we define a modeling importance criterion that measures sensitivity on return of taking
60 a suboptimal action and introduce Action Sensitive Experience Replay (ASER), a method to change
61 the replay distribution to match this criterion for off-policy reinforcement learning algorithms that use
62 a state-action value function. This allows us to focus the expressiveness of function approximators
63 on important parts of the state-space. We argue that while fitting the observation distribution (i.e.
64 minimizing the total modeling error of the data collected by the agent along its rollouts) might seem
65 logical, it is likely not the best strategy to achieve the maximum return efficiently. Since this leaves
66 deprioritized parts of the state-space with lower accuracy, we use n-step returns to bootstrap only to
67 states that are adequately modeled. We experimentally show that we find sample efficiency and/or
68 final performance gains: (i) in a simple case, such as the maze in Fig. 1 where importance is given,
69 this effective reduction of the state-space allows for significant improvements, (ii) when we transfer a
70 learned importance criterion from a previously trained policy, (iii) in some cases, when learning the
71 importance criterion during training.

72 2 Related Work

73 There have been several different proposals for non-uniform sampling or reweighting of replay buffers.
74 Some, like CER [27], add extra samples to a uniform sampled batch while others change the sampling
75 distribution altogether. Most techniques are aimed to satisfy one of the following four objectives.

76 Firstly, a common aim is to reduce the off-policyness of the selected samples for replay. This may
77 improve learning speed and stability. Previous works estimate the off-policyness using importance
78 weights [17] or multiple buffers [23] and then clip gradients or reweight experiences. In [24],
79 off-policyness is instead reduced by sampling multiple batches and taking the most similar state
80 distribution. Possible drawbacks of selecting mostly on-policy data are reduced robustness to policy
81 or environment changes. Our approach does not prioritize on-policyness specifically, however, these
82 approaches may work well together by further reducing the amount of modeled states.

83 Another objective may be to prioritize samples that are modeled poorly by the value function.
84 Previous works [4, 22] propose prioritizing the replay of experiences based on their TD-error, which
85 is analogous to how unexpected the transition is to the value function. This may be beneficial to
86 integrating new experience faster and, like our approach, will prioritize states where the value function
87 is volatile but can potentially focus the limited expressiveness on irrelevant parts of the state-space
88 (reducing the final performance and convergence rate).

89 Heuristics can also be used to enforce good coverage of transition dynamics. For example, [20]
 90 introduce prioritized sampling based on Shannon’s entropy of the state space vector. Other variants
 91 include selecting on reward [10] and state coverage [9]. Some of these works may do the opposite
 92 of our approach and the works described before, as, instead of focusing replay, they often enforce a
 93 broader coverage of the state space. In some cases, this may be preferred to increase robustness to
 94 environment changes, avoid catastrophic collapse, or for transfer learning.

95 Lastly, instead of using rules-based strategies to do prioritization, a learning-based approach to select
 96 experience from the buffer can be used. In [18, 26], training a replay policy to maximize the increase
 97 in cumulative reward by selecting transitions to train the agents’ policy on is proposed. An interesting
 98 observation in these works is that these learning methods prefer replaying recent, likely on-policy,
 99 data. These methods, however, cannot learn a prioritization similar to our approach as the learning
 100 methods do not have the information needed as input.

101 Next to what the goal of prioritization is previous works differ in how they implement prioritization.
 102 The approach used by some works discussed here [17, 23] is reweighting errors to, for example, take
 103 larger update steps for more prioritized data. Instead, the sampling distribution can also be changed
 104 so more prioritized samples are seen more often [4, 20, 22]. With very large batch sizes and many
 105 update steps per new sample, these approaches will converge to effectively the same. Since this is
 106 often not the case we use the latter approach, however, reweighting may work too.

107 Most works discussed here prioritize samples from a first-in-first-out buffer. Instead, works such as
 108 [9] change in what order data is removed from the replay buffer as they often have a limited capacity.
 109 While this may also benefit our approach, we only change how a fixed-size buffer is sampled, not
 110 how data is removed.

111 Finally, our method makes use of n-step returns for off-policy reinforcement learning algorithms. A
 112 common way of accounting for the bias towards off-policy data introduced by n-step returns is by
 113 using importance sampling or tree backup [25]. We do not use these approaches as we argue our
 114 method naturally limits the impact of this bias (see Sect. 3.3).

115 3 Action Sensitive Experience Replay (ASER)

116 ASER aims to focus the expressiveness of function approximators used in off-policy reinforcement learning by
 117 changing the sampling distribution of the replay buffer. To do this, our method consists of three additions to standard
 118 algorithms. Firstly, a formal definition of the modeling importance criterion based on the Q-function of the algo-
 119 rithm is given in Sect. 3.1. Then, this criterion is used to change the sampling distribution as described in Sect. 3.2.
 120 Lastly, due to the changed sampling distribution, we need to skip over some states when bootstrapping as described
 121 in Sect. 3.3. In Sect. 3.4 we show the implementation of ASER for SAC [8].

128 3.1 Modeling Importance Criterion

129 The modeling importance could be formalized in many
 130 ways, depending on reward distribution, action authority,
 131 etc. What is explored here is a measure to find areas of the
 132 state-action space where choosing a wrong action greatly
 133 affects the expected return for continuous state and action
 134 spaces. We define a function $p: \mathbb{R}^n \rightarrow \mathbb{R}^+$ that maps
 135 the state space s to the modeling importance. A possible
 136 modeling importance could be the norm of the second
 137 derivative of the Q-function to the action a at the optimal
 138 action. A highly negative concavity at the peak of the
 139 Q-function gives a local proxy for the rate of decrease in
 140 expected return when choosing a sub-optimal action. For

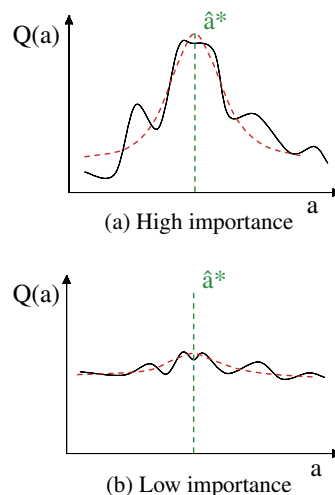


Figure 2: Visualization of Q-functions over actions where \hat{a}^* is the estimate of the optimal action according to the policy. A good importance metric differentiates these, even though metrics like a second derivative may locally be similar. The dotted red line shows an approximation of this function, where a scaled entropy or variance could be a good metric, for example.

141 multi-dimensional actions, a matrix norm of the Hessian could be used instead. However, there are
 142 multiple issues with using this definition as the modeling importance. It is often not trivial to find the
 143 peak of the Q function in continuous domains. Taking the estimated optimal action in actor-critic
 144 methods instead, for example, may not exactly coincide with the peak of the Q-function and therefore
 145 give inaccurate values. Even if the peak of the Q-function could be found, the concavity is a local
 146 metric. The second-order derivatives may be large while the total value decrease is small.

147 Therefore, a less local metric for this value loss is needed. A solution to this is to approximate the
 148 Q-function over actions for every state with a tractable function where we could use metrics like
 149 entropy or variance. Fig. 2 shows examples of Q-functions where we would like a high difference
 150 in importance. The assumption here, however, is that the reinforcement learning algorithm used
 151 will model the reduction in expected return accurately. From experiments, we found that this is not
 152 generally the case. For example, we see that with ϵ -greedy exploration, DQN may find what the
 153 best action is but will likely not accurately model the value of selecting a sub-optimal action. On the
 154 other hand, maximum entropy reinforcement learning algorithms have a built-in incentive to find the
 155 sensitivity of their actions and perform more randomly in parts of the state space where the effect on
 156 the expected return is smaller.

157 In Sec. 3.4, we show that the policy of a SAC agent already suits these needs well. However, a similar
 158 tractable approximation and exploration incentive could be implemented for other algorithms.

159 3.2 Sampling Prioritization

160 The modeling importance criterion is used to select what priority samples should be selected for
 161 replay. Similarly to PER [22], the selection is stochastic depending on the criterion and we introduce
 162 a hyperparameter α that scales the prioritization of samples. The chance of selecting a sample i is:

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}. \quad (1)$$

163 Sweeping the entire buffer to select data is too computationally expensive, therefore their prioritization
 164 will be updated with the current criterion when they are sampled. New experience samples are
 165 introduced into the buffer with the maximum prioritization that exists inside the buffer. This prioritizes
 166 integrating new experiences quickly, although this effect is limited.

167 Since PER’s prioritization criterion is based on the TD-error, which depends on what next state the
 168 transition ends up in, a bias will be introduced that needs to be annealed during training. PER does
 169 this using importance sampling. The modeling importance criterion we introduce only depends on
 170 the current state, which is independent of any stochasticity in the environment, which means we
 171 do not need this correction. We change the distribution of states that is replayed compared to the
 172 distribution of states that is seen by the agent during rollouts, this will introduce a bias in modeling
 173 errors but not in value convergence.

174 3.3 Bootstrapping

175 Typically, off-policy reinforcement learning algorithms use TD(0) updates to learn the Q-function.
 176 Here, the Q-function is bootstrapped to the value of the next state-action pair. The sampling
 177 prioritization may result in inaccurate modeling of this next state and these inaccuracies will then be
 178 pulled into the value estimation of states that do need accurate modeling. This is mainly an issue
 179 when there is a significant difference in modeling importance like in the maze environment from Fig.
 180 1.

181 To account for this, transitions can be unrolled further along the trajectory that they came from to
 182 areas where the Q-function is modeled accurately using n-step returns. Rewards along this trajectory
 183 are added up and discounted. The bootstrap target of the initially replayed transition to a transition n
 184 steps away is as follows:

$$Q_{\text{targ}}(\mathbf{s}_0, \mathbf{a}_0) = r_0 + \gamma r_1 + \gamma^2 r_2 + \dots + \gamma^{n-1} r_{n-1} + \gamma^n Q(\mathbf{s}_n, \pi(\mathbf{s}_n)) \quad (2)$$

185 where the subscripts indicate the steps away from the initial transition along the trajectory of the
 186 transition. A visualization of this bootstrapping can be found in Fig. 3.

187 The modeling importance criterion is used to
 188 decide whether the Q-function models a state
 189 well enough. We determine the criterion for an
 190 unprioritized batch and take a percentile of this
 191 batch as a threshold value b for bootstrapping.
 192 Since at the start of training the accuracy of the
 193 Q-values is low we increase this value during
 194 training.

195 The k -th percentile target is determined as fol-
 196 lows:

$$k = \min(k_m, tk_s) \quad (3)$$

197 where k_m is the maximum target percentile, t is
 198 the amount of training timesteps and k_s is the
 199 target percentile slope. This results in a sim-
 200 ple bounded linearly increasing threshold. If
 201 the criterion is lower than this threshold, the Q-
 202 function is assumed to be inaccurate and the next
 203 transition in the trajectory is taken until a transi-
 204 tion is equal to or higher than the threshold is found.

205 It is important to note this n-step bootstrapping will cause the bootstrap target to be dependent on the
 206 policy that collected it. A different policy may have taken a different trajectory by taking different
 207 actions and have collected different rewards. This will result in a bias towards older policies that are
 208 still in the buffer with off-policy reinforcement learning algorithms. There are ways to solve this, for
 209 example, with importance sampling or tree backup [25]. We argue, however, that this effect is limited
 210 since the policy dependency is only in areas where actions have a limited effect on the expected
 211 return. As soon as a state is encountered where a different policy would have a significant effect, it is
 212 used to bootstrap instead. A different policy would therefore only have a limited effect on the value
 213 of the bootstrap target.

214 3.4 Implementation

215 As explained in Sect. 3.1, we would like a tractable function to approximate the Q-function in order to
 216 find a more global metric for sensitivity. In SAC [8], the policy is modeled by a Gaussian distribution
 217 where the mean and standard deviation depend on the state. This mean and standard deviation are
 218 learned to minimize the KL divergence with a normalized exponential of the Q-function. This results
 219 in a global approximation of the sensitivity of the Q-function described by a Gaussian. Using the
 220 probability of the mode of this Gaussian works well to estimate the total sensitivity as it gives a
 221 combined metric when actions are multi-dimensional, but other metrics, like the determinant of the
 222 covariance matrix, may also work. A high probability of the mode will mean that the distribution is
 223 narrow, resulting in a fast reduction in value next to the optimal action.

224 This results in the following modeling importance:

$$p(\mathbf{s}) = \pi(\mathbf{a}|\mathbf{s}), \quad \mathbf{a} = \bar{\pi}(\mathbf{s}) \quad (4)$$

225 where $\bar{\pi}$ is the “greedy” version of the policy, in this case, the mode of the Gaussian. We, therefore,
 226 implement our method for SAC to evaluate its effectiveness. One thing to note is that in SAC the
 227 Q-value target includes an entropy term. When using n-step returns these entropy terms will need to
 228 be added in a similar way to the rewards in Equation (2) to get consistent results.

229 The complete algorithm for ASER is described in Algorithm 1. ASER can be used as a replacement
 230 for uniform sampling used in other reinforcement learning algorithms if a good importance criterion
 231 can be found. We use a replay buffer of transition tuples state \mathbf{s} , actions \mathbf{a} , reward r , next state \mathbf{s}'
 232 and end of episode (or done flag) d with prioritization p . For every gradient step, ASER will modify
 233 the batch of transitions before they are used in the normal gradient step of the algorithm. The only
 234 change that needs to be done to the algorithm it is implemented on is that any bootstrapping will need
 235 to be discounted according to the bootstrap length used by ASER, so with γ^n instead of simply γ .
 236 Other than updating the prioritization the buffer contents are not modified.

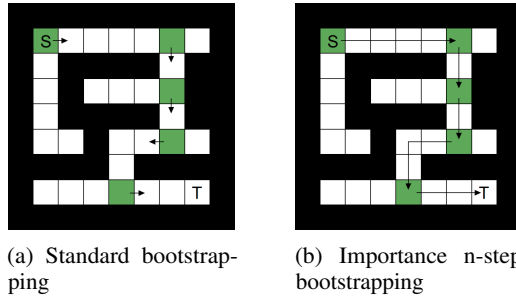


Figure 3: Bootstrapping visualized with arrows in the toy maze environment. White states, which are not modeled accurately due to the prioritized sampling, are skipped when bootstrapping to avoid pulling this badly modeled data into the well-modeled green states.

Algorithm 1: ASER

Input: Replay buffer \mathcal{D} of samples $T = (s, \mathbf{a}, r, s', d)$ with prioritization p , training timesteps t

Parameters: Maximum horizon h , maximum target percentile k_m , target percentile slope k_s , discount factor γ

```
1 for each gradient step do
2   Sample with  $P(i) = p_i^\alpha / \sum_j p_j^\alpha$  a batch of transitions  $B = \{(s, \mathbf{a}, r, s', d)_i\}$  from  $\mathcal{D}$ ;
3    $b \leftarrow$  top  $k$ -th percentile of  $p(s')$  for every  $T \in B$ , where  $k = \min(k_m, tk_s)$ ;
4   for each  $T = (s, \mathbf{a}, r, s', d)_i \in B$  do
5      $n \leftarrow 1$ ;
6     while  $p(s') < b$  and not  $d$  and  $n < h$  do
7        $(\hat{r}, \hat{s}', \hat{d}) \leftarrow$  from next transition in trajectory from buffer  $\mathcal{D}$ ;
8       Modify  $T$  with:  $r \leftarrow r + \gamma^n \hat{r}$ ;  $s' \leftarrow \hat{s}'$ ;  $d \leftarrow \hat{d}$ ;
9        $n \leftarrow n + 1$ ;
10    end
11  end
12  Normal gradient step with batch  $B$  of modified transitions  $T$  with bootstrap discount  $\gamma^n$ ;
13  Update in  $\mathcal{D}$  prioritization  $p$  for every  $T \in B$  with  $p(s)$ ;
14 end
```

237 4 Results

238 In the following section, we introduce the setup of our experiments and the Maze environment in
239 Sect. 4.1 and then show the effect of ASER on this environment in Sect. 4.2. Results that show
240 the dependency on reward function or environment definition and an ablation study of the sampling
241 prioritization and n-step bootstrapping are given in Sect. 4.3.

242 4.1 Maze environment and experiment setup

243 We test ASER within a maze four times larger than that represented in Fig. 1 and 3. In this environment,
244 the agent must move from start S to termination T. The agent can move to adjacent squares, but may
245 only return to the square it came from if there is no other valid action, i.e., when reaching a dead-end.
246 Therefore, in many states, the chosen action will have no effect, as every action leads to the same
247 next state, but in some choosing the wrong action will result in the agent eventually reaching a dead
248 end. This means that the neural network only needs to model these decision-sensitive areas and can
249 ignore the other states. SAC is continuous in state and action space and this environment is discrete.
250 Therefore, states are mapped uniformly onto a continuous axis between 0 and 1. For the action space,
251 four valid actions [UP, DOWN, LEFT, RIGHT] are defined and mapped to the values [0, 0.25, 0.5,
252 0.75]. The agent can opt for an action as a continuous value within the range [0, 1], which is then
253 mapped to the nearest valid action. For instance, should the agent choose a value of 0.4 with only
254 [UP, RIGHT] as valid options, the outcome will be a RIGHT action. We initially show the maze
255 with a reward function where a penalty is applied every time the agent reaches a dead end. Since
256 the reward function has a significant effect on ASER, in Sect. 4.3 we also show the effect when the
257 reward is given when reaching the end of the maze linearly decreasing with the timesteps spent to get
258 to the end.

259 To show the effectiveness of only learning action-sensitive parts of the state space, we also compare
260 the effect of our method with prior information about the sensitivity of actions. To do this, we create
261 an “oracle” that has perfect information about the sensitivity of actions. This oracle is used instead of
262 the modeling importance metric $p(s)$ from Equation (4), where a decision point has an importance of
263 1 and all other states have an importance of 0. Another way to use prior information is to use the
264 importance criterion from a previously trained policy instead of learning it while training.

265 We build upon the algorithm implementations of Stable Baselines 3 [19]. To create a fair comparison
266 we run hyperparameter optimization for every algorithm on the shown environments. For this, we use
267 Optuna [2] and maximize the sum of the average total reward of multiple rollouts in the evaluation
268 environment during training to optimize for speed of convergence and final performance. Tables
269 with all hyperparameters, both optimized values and SB3 defaults, can be found in Appendix A. We

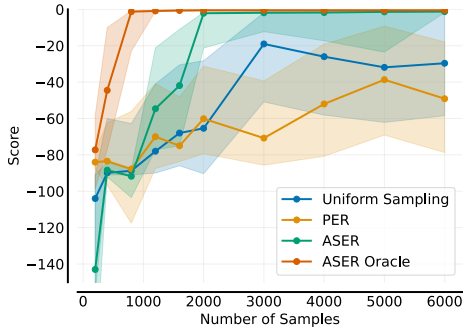


Figure 4: Sample efficiency comparison between SAC with standard replay, PER, ASER with an online learned importance criterion, and ASER with an importance criterion from an “oracle” on the Maze environment with a small, 24-neuron actor and critic network

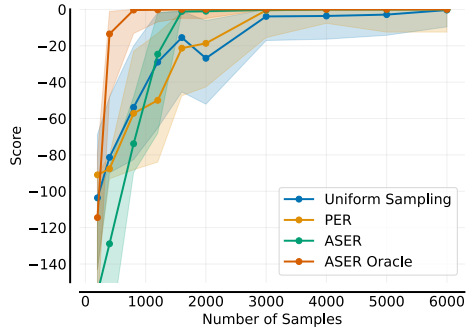


Figure 5: Sample efficiency comparison between SAC with uniform sampling, PER, and ASER with an online learned importance criterion on the Maze environment with a two-layer 256-neuron actor and critic network.

270 present all the sample efficiency graphs according to the recommendations of Agarwal et al. [1],
 271 using the interquartile mean with a 95% stratified bootstrap confidence interval from 16 training runs.

272 4.2 ASER on Maze environment

273 In Fig. 4 we show the sample efficiency curves for SAC with standard experience replay, Prioritised
 274 Experience Replay, ASER with a learned selection criterion, and ASER with the oracle. All of these
 275 agents use a small, 24-neuron policy and critic network. SAC with standard replay struggles to
 276 reliably converge in the Maze environment with these small networks, even after hyperparameter
 277 optimization. It appears that at some point additional training samples no longer improve the agent’s
 278 performance. A possible reason for this is that too much of the limited expressiveness of the neural
 279 network is used to model parts of the state-space with no impact on performance, as actions do
 280 not affect return there. Filling the replay buffer with more data will not be beneficial, as the neural
 281 network needs to filter out portions of these data to effectively incorporate it. When ASER has access
 282 to the oracle it only replays and bootstraps to decision points in the maze. With this oracle, we see a
 283 large increase in sample efficiency.

284 When the modeling importance criterion is not
 285 given, and therefore learned while training, we
 286 still see an improvement in performance against
 287 uniform sampling. ASER more quickly and con-
 288 sistentlly converges to the best policy. Since the
 289 importance now needs to be estimated by inter-
 290 acting with the environment performance picks
 291 up later compared to the oracle. PER struggles
 292 in this environment, with performance slightly
 293 below uniform sampling. Fig. 5 shows that
 294 when training a larger network of two layers of
 295 256 neurons there is still a significant sample ef-
 296 ficiency, stability, and final performance gain for
 297 ASER compared to standard experience replay
 298 and PER, but the effect is smaller than with a
 299 small network. In general, this increased expres-
 300 siveness allows for more sample-efficient and
 301 stable learning compared to the smaller network.

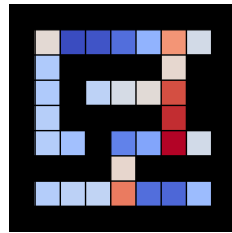


Figure 6: Sampling count relative to uniform with ASER in the Maze environment where blue is less sampling and red is more sampling than uniform.

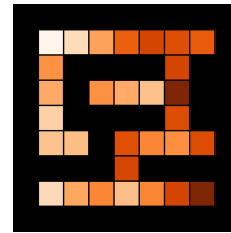


Figure 7: Bootstrap distance with ASER for the Maze environment where darker is a shorter distance. E.g. in the top row, bootstrapping distance decreases with the distance to the decision point.

302 In Fig. 6 and 7 we show, respectively, the sampling prioritization and the variable step bootstrapping of ASER with learned importance. In the best case, these should reflect Fig. 1c and 3b. We see very similar behavior in the sampling, where decision points have been learned and are prioritized, although with some effect on nearby states. The bootstrapping figure is less clear but gives a similar picture, especially in the bottom and top rows it is clear that it is bootstrapping to decision points and termination as the distance lowers the closer we get to these points.

314 4.3 Ablation and limitations

315 Sample efficiency curves for ablation of both prioritization and n-step bootstrapping with a predetermined importance criterion are shown in Fig. 8. We see that the most significant efficiency improvement comes from variable step bootstrapping, with a smaller improvement from the prioritization. Both features are needed to properly “compress” the environment and achieve stable convergence. We also show the effect of fixed 10-step returns to make sure the performance improvements shown are not only caused by this longer return. 10-step returns were chosen using parameter optimization as explained in Sect. 4.1. Somewhat surprisingly 10-step SAC with uniform sampling still performs well even though this will introduce bias in the value estimation to older policies in the buffer without importance sampling or tree backup.

333 In order for ASER to learn a good importance, there must be a significant change in value between actions. This can depend heavily on how the reward function is defined. In Fig. 9 we show two of the same mazes, where one receives only a reward at the end and one receives a penalty each time it needs to turn around. In the former, ASER does not improve performance as shown in Fig. 10 due to the smaller difference in value, resulting in less clear differences in the importance criterion.

342 In the Pendulum environment from OpenAI Gym [5], performance differences are smaller, however, some increased sample efficiency can still be achieved with pretraining, as shown in Fig. 11. Online learned ASER is unable to achieve performance gains, possibly due to being unable to stably learn the importance criterion. In classical control environments, such as the shown Pendulum environment, the difference in importance between states is less clear than in the Maze environment. In these environments, the importance can also depend more on the policy. Especially when pretraining this may have a detrimental effect on performance as the newly trained policy may find a different path through the state space making different states important to the agent.

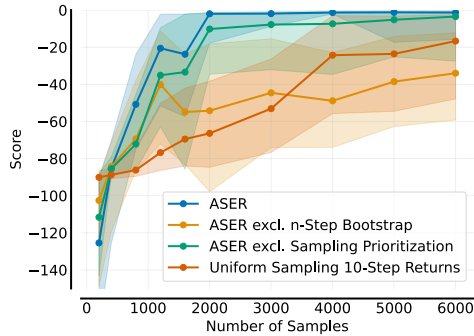


Figure 8: Sample efficiency comparison between ablations of SAC ASER on the Maze environment.

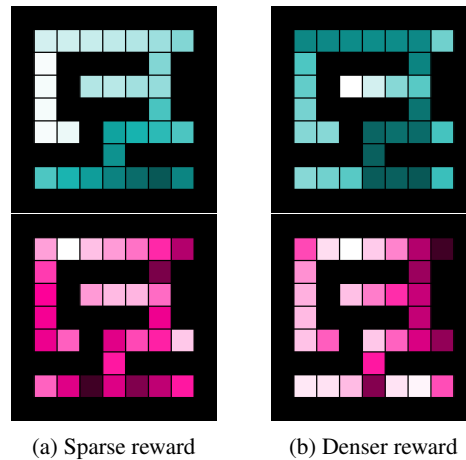


Figure 9: Comparison of the value function (top) and probability of the mode (bottom) between a sparse discounted reward at termination and a denser reward with a penalty each time the agent encounters a dead end. In the denser reward scenario figures (right), there is more immediate feedback resulting in a clearer difference in next state value at decision points and therefore a more accurate importance metric.

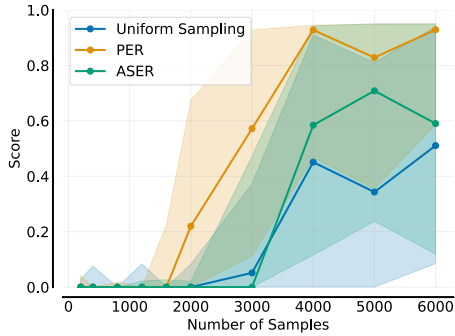


Figure 10: Sample efficiency comparison between SAC with uniform sampling, PER, and ASER with an online learned importance criterion on the Maze environment with a sparse reward function.

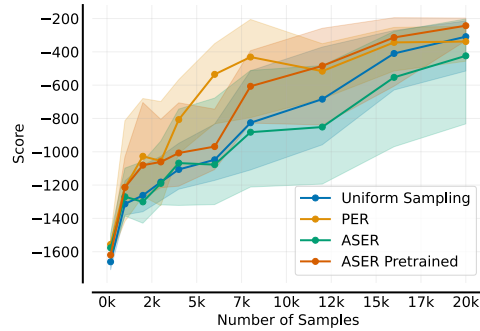


Figure 11: Sample efficiency comparison between SAC with uniform sampling, PER, ASER with an online learned importance criterion, and ASER with a pre-trained importance criterion on the Pendulum environment with a small, 24-neuron actor and critic network.

350 5 Conclusion

351 In this paper, we have presented Action Sensitive Experience Replay (ASER) which changes the
 352 experience replay distribution to prioritize states where the return is especially sensitive to non-optimal
 353 actions. This strategic reallocation of modeling resources enables parametric function approximators
 354 – such as neural networks – to focus their limited expressiveness on regions of the state-space that are
 355 most relevant for the agent’s performance.

356 Our findings show improvements in sample efficiency, stability, and final performance, particularly
 357 when the action’s sensitivity is either known in advance. When learning the sensitivity during training
 358 or carrying the sensitivity over from a previously trained policy we still observed some improvements.
 359 The improvements are more pronounced when the number of parameters in the function approximator
 360 is constrained. In order to achieve these performance improvements, our method does require
 361 environments to have rewards shaped in a way that results in clear changes in the value in some states

362 By making reinforcement learning agents model just the important parts of the state-space it could
 363 be possible for them to learn good policies in larger state-spaces without increasing the size of
 364 the function approximators used in the learning algorithm. Allowing smaller neural networks to
 365 achieve good performance may help with multi-task reinforcement learning. For example, in policy
 366 distillation [21] the policy of a large network is extracted to train a smaller, more efficient network,
 367 a step that may be omitted with a method similar to ours. Another possible application of our method
 368 is in transfer learning and specifically sim-to-real transfer. It is possible that transferring a pretrained
 369 importance criterion is more robust to environmental changes than the learned policy which can be
 370 prone to overfitting to the simulator dynamics [6, 11, 16].

371 We realize that the evaluation of our method is limited to a small number of environments in this
 372 paper. A larger study may give a better overview of the implications and possible limitations of our
 373 method. As shown, environments with clear differences in importance benefit most from our method.
 374 Therefore, we expect that some environments in the Atari 2600 benchmark suite [3] would work well
 375 with our method. This, along with an implementation on a discrete action-space algorithm like DQN,
 376 would be an interesting direction for future development.

377 **References**

- 378 [1] Rishabh Agarwal et al. “Deep Reinforcement Learning at the Edge of the Statistical Precipice”.
379 In: *Advances in Neural Information Processing Systems* (2021).
- 380 [2] Takuya Akiba et al. “Optuna: A Next-generation Hyperparameter Optimization Framework”.
381 In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery*
382 *and Data Mining*. 2019.
- 383 [3] Marc G. Bellemare et al. “The Arcade Learning Environment: An Evaluation Platform for
384 General Agents”. In: *CoRR* abs/1207.4708 (2012). arXiv: 1207.4708. URL: <http://arxiv.org/abs/1207.4708>.
- 385 [4] Marc Brittain et al. “Prioritized Sequence Experience Replay”. In: *CoRR* abs/1905.12726
386 (2019). arXiv: 1905.12726. URL: <http://arxiv.org/abs/1905.12726>.
- 387 [5] Greg Brockman et al. *OpenAI Gym*. 2016. eprint: arXiv:1606.01540.
- 388 [6] Rodney A Brooks. “Artificial life and real robots”. In: *Proceedings of the First European*
389 *Conference on artificial life*. 1992, pp. 3–10.
- 390 [7] Mikel Galar et al. “A Review on Ensembles for the Class Imbalance Problem: Bagging-,
391 Boosting-, and Hybrid-Based Approaches”. In: *IEEE Transactions on Systems, Man, and*
392 *Cybernetics, Part C (Applications and Reviews)* 42.4 (2012), pp. 463–484.
- 393 [8] Tuomas Haarnoja et al. “Soft actor-critic: Off-policy maximum entropy deep reinforcement
394 learning with a stochastic actor”. In: *International conference on machine learning*. PMLR.
395 2018, pp. 1861–1870.
- 396 [9] David Isele and Akansel Cosgun. “Selective experience replay for lifelong learning”. In:
397 *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.
- 398 [10] Max Jaderberg et al. “Reinforcement Learning with Unsupervised Auxiliary Tasks”. In: *Inter-*
399 *national Conference on Learning Representations*. 2017.
- 400 [11] Nick Jakobi, Phil Husbands, and Inman Harvey. “Noise and the reality gap: The use of simula-
401 tion in evolutionary robotics”. In: *Advances in Artificial Life: Third European Conference on*
402 *Artificial Life Granada, Spain, June 4–6, 1995 Proceedings 3*. Springer. 1995, pp. 704–720.
- 403 [12] Nathan Lambert et al. “Objective Mismatch in Model-based Reinforcement Learning”. In:
404 *Learning for Dynamics and Control*. PMLR. 2020, pp. 761–770.
- 405 [13] Long-Ji Lin. “Self-Improving Reactive Agents Based On Reinforcement Learning, Planning
406 and Teaching”. In: *Machine learning* 8 (1992), pp. 293–321.
- 407 [14] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature*
408 518 (2015).
- 409 [15] Volodymyr Mnih et al. “Playing Atari with Deep Reinforcement Learning”. In: *CoRR*
410 abs/1312.5602 (2013). arXiv: 1312.5602. URL: <http://arxiv.org/abs/1312.5602>.
- 411 [16] Stefano Nolfi and Dario Floreano. *Evolutionary robotics: The biology, intelligence, and*
412 *technology of self-organizing machines*. MIT press, 2000.
- 413 [17] Guido Novati and Petros Koumoutsakos. “Remember and forget for experience replay”. In:
414 *International Conference on Machine Learning*. PMLR. 2019, pp. 4851–4860.
- 415 [18] Youngmin Oh et al. “Learning to Sample with Local and Global Contexts in Experience Replay
416 Buffer”. In: *CoRR* abs/2007.07358 (2020). arXiv: 2007.07358. URL: <https://arxiv.org/abs/2007.07358>.
- 417 [19] Antonin Raffin et al. “Stable-Baselines3: Reliable Reinforcement Learning Implementations”.
418 In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8.
- 419 [20] Mirza Ramicic and Andrea Bonarini. “Entropy-based prioritized sampling in deep Q-learning”.
420 In: *2017 2nd international conference on image, vision and computing (ICIVC)*. IEEE. 2017,
421 pp. 1068–1072.
- 422 [21] Andrei A Rusu et al. “Policy distillation”. In: *arXiv preprint arXiv:1511.06295* (2015).
- 423 [22] Tom Schaul et al. “Prioritized Experience Replay”. In: *International Conference on Learning*
424 *Representations (ICLR)*. 2016.
- 425 [23] Samarth Sinha et al. “Experience replay with likelihood-free importance weights”. In: *Learning*
426 *for Dynamics and Control Conference*. PMLR. 2022, pp. 110–123.
- 427 [24] Peiquan Sun, Wengang Zhou, and Houqiang Li. “Attentive experience replay”. In: *Proceedings*
428 *of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 5900–5907.

- 431 [25] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press,
432 2018.
- 433 [26] Daochen Zha et al. “Experience replay optimization”. In: *Proceedings of the 28th International
434 Joint Conference on Artificial Intelligence*. 2019, pp. 4243–4249.
- 435 [27] Shangdong Zhang and Richard S. Sutton. “A Deeper Look at Experience Replay”. In: *CoRR*
436 abs/1712.01275 (2017). arXiv: 1712.01275. URL: <http://arxiv.org/abs/1712.01275>.

437 A Hyperparameters

Table 1: Optimized Hyperparameters Values Maze Small Network

		Standard	PER	ASER	Oracle ASER
λ	Learning Rate	2.480e−2	5.192e−2	1.594e−2	3.215e−2
γ	Discount Factor	9.770e−1	9.549e−1	9.358e−1	9.5267e−1
τ	Target Smoothing Coefficient	1.914e−3	9.203e−2	1.965e−2	7.399e−2
α	Sample Prioritization	-	3.414e−1	1.409e−1	1.032
h	Max Bootstrap Length	-	-	6	8
k_m	Bootstrap Target Percentile Maximum	-	-	73.54%	85.06%
k_s	Bootstrap Target Percentile Slope	-	-	5.457e−1	4.457e−1

Table 2: Optimized Hyperparameters Values Maze Large Network

		Standard	PER	ASER	Oracle ASER
λ	Learning Rate	1.439e−2	1.350e−2	3.904e−3	1.725e−2
γ	Discount Factor	9.696e−1	9.097e−1	9.827e−1	9.226e−1
τ	Target Smoothing Coefficient	1.013e−2	3.151e−2	7.655e−2	5.741e−2
α	Sample Prioritization	-	9.473e−2	3.593e−1	5.384e−1
h	Max Bootstrap Length	-	-	8	11
k_m	Bootstrap Target Percentile Maximum	-	-	71.87%	39.73%
k_s	Bootstrap Target Percentile Slope	-	-	1.739e−2	9.736e−1

Table 3: Optimized Hyperparameters Values Maze Reward at Termination

		Standard	PER	ASER
λ	Learning Rate	1.246e−2	3.106e−2	1.935e−3
γ	Discount Factor	9.599e−1	9.821e−1	9.720e−1
τ	Target Smoothing Coefficient	1.536e−2	8.151e−2	6.833e−2
α	Sample Prioritization	-	3.908e−1	1.501e−1
h	Max Bootstrap Length	-	-	17
k_m	Bootstrap Target Percentile Maximum	-	-	43.31%
k_s	Bootstrap Target Percentile Slope	-	-	6.508e−4

Table 4: Optimized Hyperparameters Values Pendulum

		Standard	PER	ASER	Pretrained ASER
λ	Learning Rate	$3.281e-2$	$1.200e-2$	$2.196e-2$	$5.365e-2$
γ	Discount Factor	$9.357e-1$	$9.447e-1$	$9.244e-1$	$9.273e-1$
τ	Target Smoothing Coefficient	$7.518e-2$	$2.538e-3$	$6.764e-2$	$4.648e-2$
α	Sample Prioritization	-	$2.538e-1$	$1.829e-1$	$1.961e-1$
h	Max Bootstrap Length	-	-	2	1
k_m	Bootstrap Target Percentile Maximum	-	-	77.05%	69.57%
k_s	Bootstrap Target Percentile Slope	-	-	$1.444e-3$	$1.064e-4$

Table 5: Stable Baselines 3 Defaults

Replay Buffer Size	1e6
Env. Steps Before Learning Starts	100
Batch Size	256
Gradient Steps Per Env. Step	1
Entropy Coefficient	Learned
Target Entropy	-1

H EValueAction: a proposal for policy evaluation in simulation to support interactive imitation learning

EValueAction: a proposal for policy evaluation in simulation to support interactive imitation learning

Fiorella Sibona*, Jelle Luijckx[†], Bas van der Heijden[†], Laura Ferranti[†], Marina Indri*

* Dipartimento di Elettronica e Telecomunicazioni (DET) - Politecnico di Torino, Italy
{fiorella.sibona, marina.indri}@polito.it

[†] Cognitive Robotics department (CoR) - Delft University of Technology, The Netherlands
{j.d.luijckx, d.s.vanderheijden, l.ferranti}@tudelft.nl

Abstract—The up-and-coming concept of Industry 5.0 foresees human-centric flexible production lines, where collaborative robots support human workforce. In order to allow a seamless collaboration between intelligent robots and human workers, designing solutions for non-expert users is crucial. Learning from demonstration emerged as the enabling approach to address such a problem. However, more focus should be put on finding safe solutions which optimize the cost associated with the demonstrations collection process. This paper introduces a preliminary outline of a system, namely EValueAction (EVA), designed to assist the human in the process of collecting interactive demonstrations taking advantage of simulation to safely avoid failures. A policy is pre-trained with human-demonstrations and, where needed, new informative data are interactively gathered and aggregated to iteratively improve the initial policy. A trial case study further reinforces the relevance of the work by demonstrating the crucial role of informative demonstrations for generalization.

Index Terms—Human-centered manufacturing, Learning from Demonstration, Interactive imitation learning, Simulation

I. INTRODUCTION AND STATE OF THE ART

In recent years, the *Industry 4.0* paradigm introduced some impactful technologies for the industrial workflow. Among these, Artificial Intelligence (AI) at large laid the foundations for implementing intelligent autonomous agents. Indeed, machine and deep learning algorithms coupled with smart sensors led to advanced human-robot perception, enabling new ways to attain safe and effective collaboration [1]. Also, simulation and virtual representations of physical systems are becoming a valuable tool to get insights on assets behaviour and to enable powerful industrial digital twins [2], [3].

Moreover, collaborative robots can help humans streamline the manufacturing process. However, human complex and creative reasoning is yet to be achieved by machines. Thereby, current and fore-coming research is investigating a human-centric vision of production lines, the so-called *Industry 5.0*, where more sustainable and value-driven production processes are created giving greater relevance to human skills [4]. When considering human-robot collaborative applications, the cognitive mismatch between a collaborative robot (*cobot*) and a human worker can be narrowed using AI, enabling the robot to interpret and adapt to the worker's behaviour. Nevertheless, the user is typically required to have the expertise necessary to understand and change the robot's behaviour. This limitation can be overcome by exploiting Learning from Demonstration

(LfD), or *imitation learning* (IL), where a human teacher demonstrates to the robot how the task should be executed.

Among the issues affecting LfD we have: (i) *dataset bias*, linked to the teacher-specific behaviour or the lack of variety of demonstrated situations and, (ii) *overfitting*, caused by too fine-tuned models or data scarcity in terms of the number of provided demonstrations. These issues hamper the generalization, or *extrapolation*, capability of LfD algorithms, resulting in undesired behaviour whenever new situations are met. To avoid such problems, the human would be required to (i) know how to offer informative and unbiased demonstrations, provided in heterogeneous set of situations, and (ii) identify and provide a sufficient number of demonstrations. This results in additional mental and physical effort on the human teacher side. The time that the human spends on performing demonstrations can be foreseen to be included in the future non-value adding activities, i.e., activities hindering the transition towards lean manufacturing paradigms [5]. Also, the authors of [6] point out that robot learning metrics should focus on time efficiency, to better reflect the *true cost for humans*.

Therefore, achieving generalization with few informative demonstrations is one of the main drivers of research in the field of LfD. What emerged from the analysed literature is that, independently of the inherent generalization capability of the proposed methods, the quantity and quality of demonstrations greatly impact the achievable extrapolation. It is then possible to derive two main objectives to reduce the *cost of generalization* (from the human point of view): reduce the number and improve the quality of demonstrations.

The following works aim at improving generalization while keeping a low number of demonstrations. Some works try to achieve adaptability by incorporating into the model a set of task variables describing the context under which demonstrations were performed [7], or conditioning the learning process on different information sources conveying the task [8]. Other works exploit dataset augmentation to achieve policy improvements for learning task-parameterized skills without increasing the number of demonstrations, such as [9], in which noise is added on recorded paths, and [10] where generated synthetic data are added to the dataset. Several other works, as [11] and [12], take advantage of reinforcement learning (RL) to explore and adapt the model to new situations, after a first phase learning from few demonstrations. In [13], goal proximity is used as a dense reward for the agent training.

Moreover, human demonstrations quality can affect the ability to achieve good extrapolation [14]. In fact, obtaining high-quality demonstrations and providing a definition for

The authors would like to thank Anna Mészáros and Giovanni Franzese, both with the CoR department at Delft University of Technology, for the invaluable support provided during algorithm adaptation to the case study. This work was supported by the European Union's H2020 project Open Deep Learning Toolkit for Robotics (OpenDR) under grant agreement #87144.

such quality appear among the most critical challenges when learning from offline human data [15], [16]. Nevertheless, some works try to exploit all available demonstrations, independently of their quality, to exploit a larger number of data. In [17], the proposed framework learns a well-performing policy also including confidence-reweighted non-optimal demonstrations. The method proposed in [18] computes a generalized trajectory from all demonstrations, while the authors of [19] fully exploit the demonstrations dataset to build a compositional task latent representation space.

The similarity-aware framework presented in [20], evaluates different representations based on a defined similarity, and then provides the user with the most similar reproduction of the demonstrated skill for unseen conditions. Also, the authors of [21] aim at extending the extrapolation capabilities of IL methods by means of virtual demonstrations, generated with the invariants method, to represent a better consistency and quality alternative to human demonstrations.

The quality and quantity of demonstration goals for good generalizing policies can be attained by a system seeking to guide the user's demonstrations interactively. In Interactive Imitation Learning (IIL) human demonstrations are periodically provided during robot execution. Compared to standard IL training, IIL can be more sample-efficient since demonstrations, in this context known as *feedbacks*, *corrections*, or *interventions*, are also collected while executing the novice policy, rather than the teacher's policy alone [22]. The method in [23] exploits the epistemic and aleatoric uncertainty information to detect ambiguities in the feedbacks to support the process of finding the best learning samples. The algorithm in [24] exploits topological persistence to detect ambiguity in a trained policy, in order to query user demonstrations only if needed, avoiding to gather demonstrations for known situations. A family of robot-gated IIL methods, stemming from DAGger [25], allows the agent to query a teacher intervention according to estimations of quantities related to task performance and uncertainty [26], [27]. In particular, ThriftyDAGger [28], considers the state novelty and the state risk of task failure to trigger corrections, given a budget of human interventions.

The EValueAction (EVA) framework, whose concept idea is introduced in this paper, has been designed to represent a support for the interactive learning process by guiding the user towards the most informative demonstrations. A state risk of failure is inferred by computing in simulation an estimate of its value function, given the executed policy. This way, new demonstrations are queried where needed, and failures are foreseen and prevented before acting in the real world, improving safety. Also, the case study thoroughly analysed in this paper provides a clear display of the dependence of generalization on the teacher expertise and resulting demonstrations' execution. The remainder of this paper is organized as follows: Section II describes the case study that brought to the research idea. Then Section III outlines the concept EVA system, followed by a possible solution for policy evaluation in simulation. Finally, Section IV draws some conclusions and sketches future research directions.

II. PRELIMINARIES

Before outlining the proposed EVA system, it is worth describing the case study that brought to the concept idea.

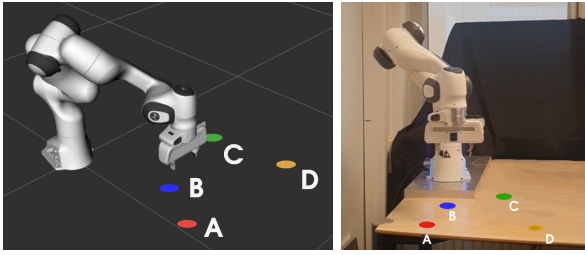
Given the following problem contextualization, the assumptions and trials outcomes are reported. In the context of flexible manufacturing, the collaborative assembly task represents a manufacturing operation of high practical relevance: the robot is typically allocated repetitive or power demanding tasks, while the human performs highly dexterous operations [29]. In such tasks, being able to adapt to new situations is crucial, as the workspace configuration and the positions and shapes of assembly parts often vary. Following the assembly task subdivision introduced in [30], we consider an *approaching phase* and an *assembling phase*. The authors perform such division in order to prevent the potential under-fitting caused by the variability of the demonstration data between the two assembly stages. In particular, the approaching phase is the one most affected by environment constraints and configuration of parts. As such, this stage would take the most advantage from an IL algorithm with good generalization capabilities learning from few demonstrations. Therefore, we consider a human teaching a cobot how to perform a desired approaching phase. For example, we can imagine a peg-in-hole collaborative assembly task, where the approaching phase can be reduced to a pick-and-place or hovering task.

As extensively reported, improving generalization is widely tackled by researchers, as poor generalization capabilities is a common issue in IL. The case study described hereafter provides a clear idea of how generalization capabilities of IL methods can be greatly affected by the way demonstrations are performed.

1) *A simplified learning scenario*: For the execution of the trials, we have taken advantage of the algorithm presented in [31]. Specifically, we exploited the Interactive Learning of Stiffness and Attractors (ILoSA) framework to learn attractors only from kinesthetic demonstration, without taking advantage of the interactive part. In ILoSA, the confidence level provided by the use of Gaussian Processes (GPs) allows to detect when the cobot lands on an unknown state, and is exploited to implement a stabilizing attractive field to lead the robot towards minimum variance regions. As ILoSA recorded demonstrations within a global frame, generalization was limited around the visited states. Hence, we borrowed the general idea of using local reference frames, as hinted in [32], to be able to test over different final positions.

In summary, we exploited ILoSA to learn from human demonstrations the attractor distance for the robot impedance control, learned with respect to the final position reference frame. The focus of the trials was on reaching as many new goals as possible with one single informative demonstration.

2) *Demonstrations setup*: The demonstrations and trials have been performed on a 7 DOF Franka-Emika Panda with an impedance controller and a ROS communication network. The demonstration consisted in moving the robot from a reference home position to a final position of interest. For our approaching phase scenario, we assume that an assembly part is picked from some location and brought to the reference home position at each assembly task iteration. Then the final desired position (before assembly stage) is reached exploiting the learned policy. As expected, the employed IL algorithm was able to generalize in the neighbourhood of a demonstrated trajectory task. Therefore, to check the generalization behaviour over



(a) Points of interest in *rviz*. (b) Real setup.

Fig. 1: Four goals of interest on the working plane have been chosen for the generalization check trials.

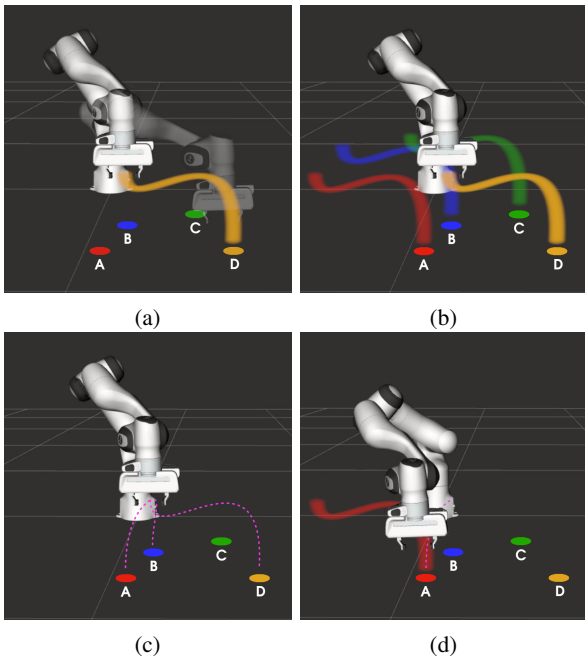


Fig. 2: Medium expertise demonstration on D.

destination points relatively far from the demonstrated one, we have chosen four goals of interest (Figure 1).

a) Automatic demonstration recording: The ILoSA Jupyter interactive Python code has been modified so as to interactively input a final goal point to accordingly name saved data and trained GPs models files/structures, while automatically generating a `.txt` file to keep track of the tests and relative outcomes. Also, after recording, the produced code lets iteratively test the learned policy over the set of interest points. This procedure allowed for faster data collection and consultation.

3) Trials outcomes: The execution of several trials allowed to identify three relevant cases that we describe hereafter. Assumption: after trials, demonstrations with points A and B as final destination turned out to be the least generalizing thus are not considered as demonstration points.

a) Medium expertise demonstration: Destination hovering point is one among A, B, C, and D (refer to Figure 2). A human teacher performed a demonstration with the position above point D as the final position to be reached (Figure 2a). Note that the height recorded during the single demonstration

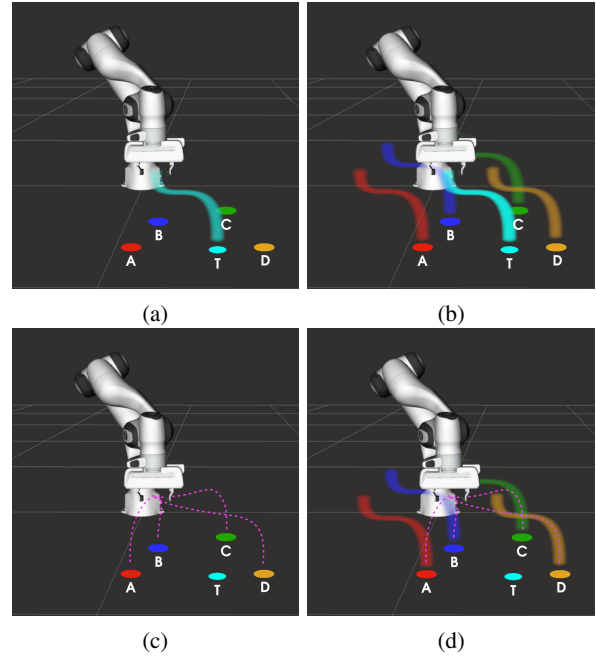


Fig. 3: Expert demonstration on T.

was kept as a hovering height for all worktable points set as desired goals. After the policy training, the motion was learned in the local reference frame leading to areas with high confidence for each final goal, as shown in Figure 2b. When the policy was tested on all the points of interest, it generalized well on A and B, since the IL algorithm attracted the robot towards the high confidence areas thanks to its stabilization prior (Figure 2c). Then, when A was the input for policy testing, the cobot was able to reach it (Figure 2d).

b) Expert knowledge demonstration: Destination hovering point is not among the points of interest (Figure 3). An expert human teacher, well aware of the IL algorithm behind the training process, demonstrated a motion to reach the position above an empirically chosen test point T (Figure 3a), which intuitively would have allowed the trained policy to generalize on all points of interest. Indeed, after GPs training, the policy generalized on A, B, C and D (Figure 3b–3d). Note that, in this case, the generalization capability is influenced by the specific motion shape and the stabilization fields, since, depending on both, the robot may or may not be attracted to the locally learned motion.

c) Little knowledge demonstration: We then let a human teacher kept unaware of the underlying IL algorithm demonstrate a motion towards D. As the human teacher did not have any information on the influence the motion shape would have had on the generalization capability, the resulting policy surprisingly didn't generalize on the other points of interest, learning the motion to hover on D only. Note that the human teacher had expertise in robotics but no information on the learning process.

4) Main takeaways: The performed trials then brought out that with an expert teacher, able to infer the most informative demonstrations with the aim of generalization, a single demonstration could be sufficient to generalize over four points

of interest and their neighbourhoods (plus the testing point for demonstration and relative neighbourhoods). Conversely, if the user is non-expert, with one demonstration only one task would be learned. Independently of the used LfD method and the simplicity of the learning scenario, this gave some intuitions on how robots can help to improve demonstration quality and reduce demonstration quantity. Namely, as the machine is well aware of the policy, it can give some suggestions on where new demonstrations would be most informative and potentially perform virtual demonstrations, thereby reducing the number of demonstrations demanded to the human. This led to the idea of exploiting simulation for policy evaluation, which in turn would also improve safety during collaboration.

III. THE EVALUEACTION (EVA) SYSTEM

The proposed system, EVA, seeks to provide a framework to guide the human teacher during the interactive demonstrations to improve the generalization over new states of a learned policy. The main elements of the system would be: (i) the LfD method of choice, (ii) automatic recording of demonstrations to populate a dataset, (iii) a policy evaluation algorithm exploiting a digital twin, (iv) human-robot interface for bidirectional information exchange.

The overall goal is to let the robot successfully perform a task by taking actions $\mathbf{a} \in \mathcal{A}$, given observations $\mathbf{o} \in \mathcal{O}$, where \mathcal{A} and \mathcal{O} are the robot's action and observation spaces, respectively. These actions could, for example, be reference end-effector configurations, while observations could comprise joint angle measurements and camera images. These observations result from states $\mathbf{s} \in \mathcal{S}$, i.e., $\mathbf{o} = O(\mathbf{s})$ where \mathcal{S} is the state space and O the observation mapping $O: \mathcal{S} \rightarrow \mathcal{O}$. We make this distinction between states and observations, since full state information is often not available in real world scenarios. Since actions follow from observations, the aim is to find a policy π (which is a mapping from observations to actions, i.e., $\pi: \mathcal{O} \rightarrow \mathcal{A}$) such that a task is performed successfully. Furthermore, we would like this policy to be successful starting from a set of initial states $\mathbf{s}_0 \in \mathcal{S}_0 \subset \mathcal{S}$. That is to say, the robot policy does not have to be successful starting from all possible states, but should be performing well for the actual distribution over initial states it encounters, which we denote by $p(\mathbf{s}_0)$. Furthermore, successful completion of a task can be determined by defining a goal state set $\mathcal{S}_g \subset \mathcal{S}$. Note that goals could be dynamic and part of the state \mathbf{s} . We can quantify the optimality of a policy by defining a reward function $R(\mathbf{s})$. Given the reward function, we can define the value of a state [33]:

$$V_\pi(\mathbf{s}) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R(\mathbf{s}_{t+k+1}) \middle| \mathbf{s}_t = \mathbf{s} \right], \quad (1)$$

where $\mathbb{E}_\pi[\cdot]$ is the expected value given that policy π is executed, t is any time step and γ is the discount rate with $0 \leq \gamma \leq 1$. Intuitively, the value of a state says how well the policy performs on average starting from that state following policy π . We define the optimal policy π^* as the one that has the highest expected value given our distribution of initial states:

$$\pi^* = \arg \max_{\pi \in \Pi} \mathbb{E}_{\mathbf{s}_0 \sim p(\mathbf{s}_0)} [V_\pi(\mathbf{s}_0)], \quad (2)$$

where Π is the policy space and $\mathbb{E}_{\mathbf{s}_0 \sim p(\mathbf{s}_0)}[\cdot]$ denotes the expectation given that \mathbf{s}_0 is drawn from distribution $p(\mathbf{s}_0)$. However, in practice it is often not trivial to come up with an appropriate reward function $R(\mathbf{s})$. This would particularly be the case if we were to solve this problem with RL. In that case, the reward function should ideally guide the robot towards successful behaviour [34], which would require *reward shaping* [35]. This can be a time consuming process that could also lead to suboptimal behaviour. Robotic RL also results in challenges related to data efficiency and safety [34], [36]. Therefore, we choose to find π^* by LfD. In this setting, it is not required that $R(\mathbf{s})$ guides the agent and we can only reward success:

$$R(\mathbf{s}) = \begin{cases} 1 & \text{if } \mathbf{s} \in \mathcal{S}_g \\ 0 & \text{else} \end{cases}. \quad (3)$$

Worth noting is that with this reward function and $\gamma = 1$, the optimal policy π^* will simply be the policy with the highest success rate. By setting $0 < \gamma < 1$, not only success will be rewarded, but the robot learner will also be stimulated to solve the task in minimum-time. This is desirable in our scenario, as time is related to cost in most industrial applications.

Our method falls within the realm of IIL, hence demonstrations are also interactively gathered while executing the learner policy. This results in demonstrations for states that the robot learner actually encounters, rather than only demonstrations for states the teacher encounters. We can collect a dataset with N demonstration trajectories $\mathcal{D} = \{\tau_0, \tau_1, \dots, \tau_N\}$ and try to imitate the expert policy. We consider demonstrations that consist of observation vectors, i.e., $\tau = [\mathbf{o}_0, \mathbf{o}_1, \dots, \mathbf{o}_T]$ where T is the last time step of demonstration τ . An estimate of the optimal policy can be obtained by minimizing a loss between the demonstrated trajectories and trajectories resulting from the policy:

$$\hat{\pi}^* = \arg \min_{\pi \in \Pi} L(\pi, \mathcal{D}). \quad (4)$$

A popular choice for L is the Kullback–Leibler divergence [37] between the distribution of observations induced by the learner policy and teacher policy [38]. We should however take into consideration that demonstrations are costly, since they can be time consuming for the human and ideally we would like the system to have a high level of autonomy to maximize efficiency. Therefore, we wish to optimize task success while minimizing the number of demonstrations:

$$\begin{aligned} \max_{\mathcal{D}} \mathbb{E}_{\mathbf{s}_0 \sim p(\mathbf{s}_0)} [V_{\hat{\pi}^*}(\mathbf{s}_0)] - \lambda |\mathcal{D}| \\ \text{s.t. } \hat{\pi}^* = \arg \min_{\pi \in \Pi} L(\pi, \mathcal{D}), \end{aligned} \quad (5)$$

where $|\mathcal{D}|$ is the cardinality of \mathcal{D} (the number of demonstrated trajectories) and λ a regularization parameter. In this way, we have arrived at an expression for the optimal set of demonstrations. The maximization of the expectation of the state values in (5) ensures that we maximize task success, while the regularization term penalizes the number of demonstrations in our dataset. We choose to penalize the number of trajectories τ rather than their length, since a longer demonstration is preferred over multiple short ones, both by the human and considering the potential cost per demonstration for preparing and processing it.

Finding the optimal solution to the optimization problem in (5) is difficult, because the problem does not have an analytical solution or gradient, and it can quickly become intractable, because it may take a large number of evaluations to find a good solution. In the context of the optimization problem in (5), evaluating the performance of a candidate solution (i.e., a dataset of demonstrations) requires collecting human demonstrations and physical experimentation to evaluate the performance, which can be time-consuming and expensive. Then, it is important to design an optimization algorithm that balances exploration and exploitation and is efficient in terms of the number of evaluations required to find a good solution.

Simulation is a cost-effective means of evaluating candidate solutions without the risks and expenses associated with real-world experimentation. Recent advancements in parallelized physics simulation on accelerated platforms have enabled fast simulations [39], [40]. However, the dissimilarities between the simulator and real-world environments can hinder the transferability of policies learned in simulation to real-world settings. As learning methods tend to exploit these differences to maximize simulated rewards, simulators' conventional use results in overestimation of real-world performance. This may lead to safety hazards and unexpected failures.

To address this issue, we propose to swap the real-world and simulator's roles to synthesize policies using human demonstrations and evaluate them using accelerated physics simulation. By doing so, discrepancies between simulation and reality lead to an underestimation of real-world performance. Failures in simulation may be attributed to either a sub-optimal policy or discrepancies. In case of success, the policy was robust enough to achieve the goal despite the discrepancies. The proposed solution does not rely on gradient information and instead uses a simulation based search strategy to find a good solution for the problem in (5). The system comprises two phases: pre-training and lifelong learning. During the pre-training phase, real-world demonstrations are continuously provided by the human operator, until a certain success rate is achieved in simulation or a maximum number of iterations is reached (see Algorithm 1). After the pre-training phase, the robot enters the lifelong learning phase, where it executes the policy independently. However, it will halt and request a human-demonstration if it encounters a state that resulted in a low success rate in simulation (see Algorithm 2).

Algorithms 1 and 2 provide the workflow of these two phases, while the involved functions are detailed below.

`evaluate`: this function estimates the policy's performance by computing the value function V_π , as defined in (1), which reflects the policy's performance over the induced state distribution, given a reward function R and the initial state distribution $p(s_0)$. Computing V_π may be impractical due to the large number of real-world evaluations required and limited access to the full state s . Monte Carlo sampling and an accelerated physics simulator can overcome such challenges (e.g., [39], [40]), as both are suitable for parallel implementation, to estimate an approximate value function $\hat{V}_\pi(o)$ that is a function of the observations o , serving as a proxy for the real value function. If there are significant differences between the real-world and simulator, a policy trained with real-world demonstrations may, in some cases,

Algorithm 1: Pre-training phase.

Input: Reward function: $R(s)$ // See Eq. (3)
 LfD method: $L(\pi, \mathcal{D})$ // See Eq. (4)
 Initial states: $s_0 \sim p(s_0)$
 Initial policy: π_0
 Success threshold: α
Output: Pre-trained policy: π
 Dataset: \mathcal{D}

```

1  $\mathcal{D} \leftarrow \emptyset$  // Initialize empty dataset
2  $\pi \leftarrow \pi_0$  // Initialize policy
3  $\hat{V}_\pi \leftarrow \text{evaluate}(R, p(s_0), \pi)$  // In simulation
4 do
5    $s_{\text{start}} \leftarrow \text{suggest}(\hat{V}_\pi)$  // Start state of demo
6    $\tau \leftarrow \text{demonstrate}(s_{\text{start}})$  // In real-world
7    $\mathcal{D} \leftarrow \mathcal{D} \cup \{\tau\}$  // Aggregate data
8    $\pi \leftarrow \text{optimize}(L, \mathcal{D})$  // Update policy
9    $\hat{V}_\pi \leftarrow \text{evaluate}(R, p(s_0), \pi)$  // In simulation
10 until  $\mathbb{E}_{s_0 \sim p(s_0)}[\hat{V}_\pi(O(s_0))] > \alpha$  // Success rate
```

Algorithm 2: Lifelong learning phase.

Input: Reward function: $R(s)$ // See Eq. (3)
 LfD method: $L(\pi, \mathcal{D})$ // See Eq. (4)
 Initial states: $s_0 \sim p(s_0)$
 Pre-trained policy: π // See Alg. 1
 Dataset: \mathcal{D} // See Alg. 1

```

1  $\hat{V}_\pi \leftarrow \text{evaluate}(R, \mathcal{S}_{\text{start}}, \pi)$  // In simulation
2  $s \leftarrow \text{reset}(p(s_0))$  // In Real-world
3 while running do
4    $o \leftarrow O(s)$  // Read sensor observations
5   if  $\hat{V}_\pi(o) > \alpha$  then run
6      $a \leftarrow \pi(o)$  // Get action
7      $s \leftarrow \text{act}(a)$  // In real-world
8   else request demonstration
9      $s_{\text{start}} \leftarrow s$  // Demo from current state
10     $\tau \leftarrow \text{demonstrate}(s_{\text{start}})$  // In real-world
11     $\mathcal{D} \leftarrow \mathcal{D} \cup \{\tau\}$  // Aggregate data
12     $\pi \leftarrow \text{optimize}(L, \mathcal{D})$  // Update policy
13     $\hat{V}_\pi \leftarrow \text{evaluate}(R, p(s_0), \pi)$  // In simulation
14  if done then
15     $s \leftarrow \text{reset}(p(s_0))$  // In real-world
```

always fail to solve the task in simulation. In this case, we propose to use on-policy RL algorithms such as [41].

`suggest`: This function proposes a new starting state s_{start} for the next demonstration based on the policy's simulated performance, which is reflected by the estimated value function $\hat{V}_\pi(o)$. Note that the starting state does not necessarily have to lie in the set of initial states \mathcal{S}_0 . Proper selection of starting states can significantly reduce the number of demonstrations needed to achieve adequate performance, a task that is typically performed by an expert user with knowledge of the chosen LfD method. Alternatively, meta learning, a technique for learning to learn and adapt to new tasks, can be employed [42]. In this case, a meta learning model would be trained to suggest the next starting state for a demonstration expected to improve the current value function $\hat{V}_\pi(o)$ the most.

`demonstrate`: This function requests the human to provide a human-demonstration from a given starting state s_{start} . Depending on the task, this could occur via, for example, kinesthetic teaching or teleoperation [22]. Then, `optimize` estimates a policy $\hat{\pi}^*$, as defined in (4), by minimizing the loss function determined by the selected method. Finally, `reset` resets the real-world system to an initial state, and `act` applies the action proposed by the robot's policy π to the system.

IV. CONCLUSIONS

This paper conceptually introduced EVA, a system conceived to make the IIL demonstrations process less costly and more safe for humans, while optimizing the exploration and exploitation balance for good generalization.

Motivations for the potential relevance of EVA as a support to IIL applications have been laid down, by analysing state-of-the-art works in the field. The provided core algorithms bring forward the system working flow. Future works shall fully implement EVA, and investigate its actual functionality by assessing its performance with respect to other IIL methods.

REFERENCES

- [1] A. Bonci, P. D. Cen Cheng, M. Indri, G. Nabissi, and F. Sibona, "Human-robot perception in industrial environments: A survey," *Sensors*, vol. 21, no. 5, p. 1571, 2021.
- [2] G. Castañé, A. Dolgui, N. Kousi, B. Meyers, S. Thevenin, E. Vyhmeister, and P.-O. Östberg, "The ASSISTANT project: AI for high level decisions in manufacturing," *International Journal of Production Research*, pp. 1–19, 2022.
- [3] "NVIDIA Omniverse," Available online: <https://developer.nvidia.com/nvidia-omniverse>.
- [4] P. K. R. Maddikunta, Q.-V. Pham, B. Prabadevi, N. Deepa, K. Dev, T. R. Gadekallu, R. Ruby, and M. Liyanage, "Industry 5.0: A survey on enabling technologies and potential applications," *Journal of Industrial Information Integration*, vol. 26, p. 100257, 2022.
- [5] D. Ramesh Kumar, S. Devadasan, and D. Elangovan, "Mapping of non-value adding activities occurring in classical manufacturing companies with lean strategies," *Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering*, vol. 236, no. 3, pp. 894–906, 2022.
- [6] E. Johns, "Back to reality for imitation learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 1764–1768.
- [7] M. Arduengo, A. Colomé, J. Borràs, L. Sentis, and C. Torras, "Task-adaptive robot learning from demonstration with gaussian process models under replication," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 966–973, 2021.
- [8] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn, "Bc-z: Zero-shot task generalization with robotic imitation learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 991–1002.
- [9] Y. Wang, Y. Hu, S. El Zaatari, W. Li, and Y. Zhou, "Optimised learning from demonstrations for collaborative robots," *Robotics and Computer-Integrated Manufacturing*, vol. 71, p. 102169, 2021.
- [10] J. Zhu, M. Gienger, and J. Kober, "Learning task-parameterized skills from few demonstrations," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4063–4070, 2022.
- [11] M. Akbulut, E. Oztop, M. Y. Seker, X. Hh, A. Tekden, and E. Ugur, "Acnmp: Skill transfer and task extrapolation through learning from demonstration and reinforcement learning via representation sharing," in *Conference on Robot Learning*. PMLR, 2021, pp. 1896–1907.
- [12] Y. Wang, C. C. Beltran-Hernandez, W. Wan, and K. Harada, "An adaptive imitation learning framework for robotic complex contact-rich insertion tasks," *Frontiers in Robotics and AI*, p. 414, 2022.
- [13] Y. Lee, A. Szot, S.-H. Sun, and J. J. Lim, "Generalizable imitation learning from observation via inferring goal proximity," *Advances in neural information processing systems*, vol. 34, pp. 16 118–16 130, 2021.
- [14] T. Xue, H. Girgin, T. S. Lembono, and S. Calinon, "Guided optimal control for long-term non-prehensile planar manipulation," *arXiv preprint arXiv:2212.12814*, 2022.
- [15] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," *arXiv preprint arXiv:2108.03298*, 2021.
- [16] M. Sakr, Z. J. Li, H. M. Van der Loos, D. Kulić, and E. A. Croft, "Quantifying demonstration quality for robot learning and generalization," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9659–9666, 2022.
- [17] S. Zhang, Z. Cao, D. Sadigh, and Y. Sui, "Confidence-aware imitation learning from demonstrations with varying optimality," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 340–12 350, 2021.
- [18] L. Panchetti, J. Zheng, M. Bouri, and M. Mielle, "Team: a parameter-free algorithm to teach collaborative robots motions from user demonstrations," *arXiv preprint arXiv:2209.06940*, 2022.
- [19] X. Bian, O. Mendez, and S. Hadfield, "Generalizing to new tasks via one-shot compositional subgoals," *arXiv preprint arXiv:2205.07716*, 2022.
- [20] B. Hertel and S. R. Ahmadzadeh, "Similarity-aware skill reproduction based on multi-representational learning from demonstration," in *2021 20th International Conference on Advanced Robotics (ICAR)*. IEEE, 2021, pp. 652–657.
- [21] R. Burlizzi, M. Vochten, J. De Schutter, and E. Aertbeliën, "Extending extrapolation capabilities of probabilistic motion models learned from human demonstrations using shape-preserving virtual demonstrations," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 10 772–10 779.
- [22] C. Celemin, R. Pérez-Dattari, E. Chisari, G. Franzese, L. de Souza Rosa, R. Prakash, Z. Ajanović, M. Ferraz, A. Valada, J. Kober *et al.*, "Interactive imitation learning in robotics: A survey," *Foundations and Trends® in Robotics*, vol. 10, no. 1-2, pp. 1–197, 2022.
- [23] C. Celemin and J. Kober, "Knowledge-and ambiguity-aware robot learning from corrective and evaluative feedback," *Neural Computing and Applications*, pp. 1–19, 2023.
- [24] J. Luijckx, Z. Ajanovic, L. Ferranti, and J. Kober, "Partnr: Pick and place ambiguity resolving by trustworthy interactive learning," *arXiv preprint arXiv:2211.08304*, 2022.
- [25] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.
- [26] J. Zhang and K. Cho, "Query-efficient imitation learning for end-to-end autonomous driving," *arXiv preprint arXiv:1605.06450*, 2016.
- [27] R. Hoque, A. Balakrishna, C. Putterman, M. Luo, D. S. Brown, D. Seita, B. Thananjeyan, E. Novoseller, and K. Goldberg, "Lazydagger: Reducing context switching in interactive imitation learning," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2021, pp. 502–509.
- [28] R. Hoque, A. Balakrishna, E. Novoseller, A. Wilcox, D. S. Brown, and K. Goldberg, "Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning," *arXiv preprint arXiv:2109.08273*, 2021.
- [29] D. K. Jha, S. Jain, D. Romeres, W. Yezunidis, and D. Nikovski, "Generalizable human-robot collaborative assembly using imitation learning and force control," *arXiv preprint arXiv:2212.01434*, 2022.
- [30] H. Hu, X. Yang, and Y. Lou, "A robot learning from demonstration framework for skillful small parts assembly," *The International Journal of Advanced Manufacturing Technology*, vol. 119, no. 9-10, pp. 6775–6787, 2022.
- [31] G. Franzese, A. Mészáros, L. Peternel, and J. Kober, "Ilosa: Interactive learning of stiffness and attractors," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 7778–7785.
- [32] A. Mészáros, G. Franzese, and J. Kober, "Learning to pick at non-zero-velocity from interactive demonstrations," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6052–6059, 2022.
- [33] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [34] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.
- [35] A. D. Laud, *Theory and application of reward shaping in reinforcement learning*. University of Illinois at Urbana-Champaign, 2004.
- [36] 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*, vol. 40, no. 4-5, pp. 698–721, 2021.
- [37] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.
- [38] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.
- [39] V. Makovychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.
- [40] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem, "Brax—a differentiable physics engine for large scale rigid body simulation," *arXiv preprint arXiv:2106.13281*, 2021.
- [41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
- [42] F. Alet, T. Lozano-Pérez, and L. P. Kaelbling, "Modular meta-learning," in *Conference on robot learning*, 2018, pp. 856–868.

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,*}

^aUnit 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 therefore 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 [57]. 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 [62]: 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 [57]. 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 [62]. 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 [49]. Therefore, we additionally provide a general HRC software framework, based on ROS [40], which can be utilized to replicate our developments. The framework is build around OpenDR [38], 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

*Corresponding author

 roel.pieters@tuni.fi (R. Pieters)

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 [29], i.e., perception model selection and training data collection. The last two identified challenges relate to the application 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 [26] or humans [34, 56]. 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 [23]), 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 [36]. 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 [50].
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 [31].

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 [54]. 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 [45]. 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 [59]. 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 [57]. Formal definitions of collaboration, working zones and operating modes are common [53] and standards provide requirements and design guidelines to ensure safety for operators. [55] 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. Other overviews of human-robot collaboration approaches can be easily found, for example, towards the topics of robotic vision [43] and machine learning [45], indicating the popularity of the topics, either individually, or combined. Proactive collaboration between human and robot is highlighted in [22], with emphasis on cognitive, predictable and self-organizing perspectives. Current challenges are found, which call for future research direction that address real-world applications.

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 [34].

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 [63]. Different visual modalities can be used to detect humans [24]. In [35], 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 [28] 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 [61], which utilizes a learning-based approach for human classification. The work, however, targets large indoor public spaces and a mobile service robot. In [24] 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 [39], 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 [51], which can utilize single as well as multiple cameras.

Gesture detection - Detection and recognition of human gestures has also been of interest to robotics. In [25], 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. [33] demonstrates real-time human-robot interaction 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 [56, 48]. 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 [60]. In relation to human-robot collaboration, research on action recognition has also focused on industrial activities [10, 8] and pose forecasting [44], 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 [26]. Real-time object detection is an active research problem to allow adoption to robotics applications, and many works can be found that have utilized detectors for tasks such as robot grasping [11]. Popular approaches are for example, Faster R-CNN [42], Yolo [41] and SSD [27]. 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 [64], template-based methods such as PoseCNN [7] and voting based methods such as DenseFusion [55]. For both object detection and pose estimation, datasets can be found, for example, Pascal VOC [12] and COCO [23] 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 [32]. 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 [52]. 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 [30]. 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) [53]. 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 part 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.

2.5. Comparison to our approach

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 (see Section 2.2-2.3). 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 adoption of perception tools is often possible by open-source software, details on integration are usually limited to just the tool itself [33] and not to a robotics framework [35] (e.g., ROS). This is also found in other works, where different perception tools are reviewed to detail the state of the art, e.g. in terms of robotic vision [43] and machine learning [45]. What these works do not cover is the challenges and issues faced with respect to data collection and the practical integration of the tools to a robot. While [14] and [22] do include challenges, these are not related to technical integration. Our work aims to fill this gap, by detailing three different visual perception tools for human-robot collaboration. For all three, we provide details on how to replicate our work, from dataset generation and training tools, to code examples (Python, ROS) and integration with a collaborative robot. All developments are open-source in the OpenDR toolkit¹.

¹<https://github.com/opendr-eu/opendr>

3. Visual recognition modules

All three integrated visual recognition modules utilize color images for perception. Depth perception was intentionally excluded such that models can 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 short 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. The lightweight version of OpenPose is selected [37], as it achieves detections in realtime. 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 [37], which was trained and evaluated with the COCO 2017 dataset [23] under default training parameters.

3.2. Human action recognition

Method - Recognition of human actions is done with ST-GCN [60], a real-time skeleton-based human action recognition framework, as it can utilize the lightweight OpenPose model [37]. The method takes the location of the human joints in every image, and generates a sequence of detected human skeleton graphs, connected both spatially and temporally. Depending on the dataset the method can detect a large number of different human actions, ranging from daily activities to complex actions with interactions.

Data generation and model training - The smallest training dataset is selected (NTU-RGB+D [46]), as it contains the most relevant human action classes (60 classes in 56,000 human action clips). 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 - Mask R-CNN from Detectron2 [58] was selected for object and target detection in the scene, as performance was preferred over inference time. Mask R-CNN combines a Region Proposal Network (RPN) with the CNN model, to simultaneously predict object bounds and objectness scores at each position. After detection, orientations are 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

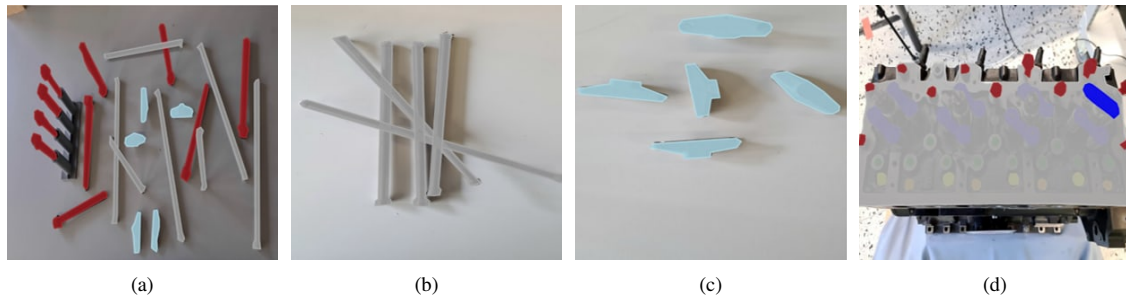


Figure 1: Image annotations for assembly objects, including bolts (red), pushrods (grey) and rocker arms (light blue); (a), (b) and (c), and targets objects, including Diesel engine (grey), small (yellow) and big (orange) pushrod holes, bolt holes (green) and rocker arm locations (dark blue); (d). Annotations 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.

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 around 280,000 images [47]. The methods for data generation and annotation are available in the OpenDR toolkit¹.

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 ergonomics 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 (object and target detection) and for human task coordination (human skeleton detection and human action recognition).

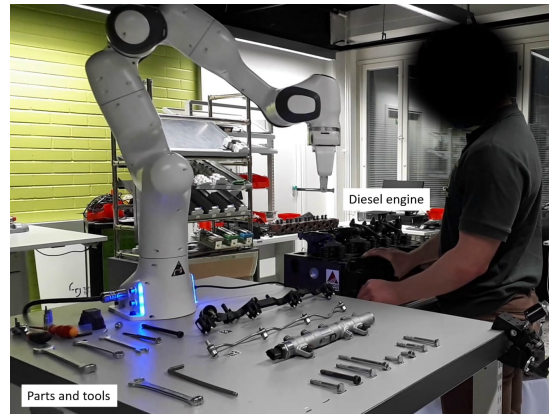


Figure 2: Experimental setup with a collaborative robot (Franka Emika), Diesel engine and parts for assembly tasks.

4.2. Integration

All developments are integrated in the OpenDR¹ toolkit [38] with ROS/ROS2² nodes of the perception tools and ROS moveit2³ scripts for the human-robot collaboration scenarios. A description of the use case, and the individual perception modules, has been documented⁴, enabling to easily replicate (and extend) our work. For robot and perception hardware, we utilize the Franka Emika collaborative robot⁵ and two Intel Realsense D435 cameras, one on the end-effector of the robot and one front-facing to the person for human perception. Computations are done on a Ubuntu PC with Nvidia GTX 1080 Ti GPU, running ROS Noetic.

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

²<https://www.ros.org/>

³<https://moveit.picknik.ai/>

⁴<https://trinityrobotics.eu/use-cases/sensor-based-human-robot-collaboration/>

⁵<https://franka.de/>

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 utilized in a similar manner [38]. 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 Studio⁶ and Albumentations⁷, respectively. These are also integrated into the OpenDR perception tools, in form of Python scripts and Jupyter notebooks⁸.

Listing 1: Object and target detections script in OpenDR¹

```
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 Moveit2³ 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 grasping. Additional actions include end-effector rotations (rotate_EE) and gripper actions (move_gripper) and can take input from visual modules, as shown by the inclusion of object and place.

Listing 2: Robot actions script in OpenDR¹

```
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)
```

⁶<https://labelstud.io/>

⁷<https://albumentations.ai/>

⁸<https://jupyter.org/>

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 OpenDR¹

```
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 and 2 provides details of the different perception modules, their corresponding datasets for training and inference, and their prediction accuracy results. 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, is that the datasets cannot be easily extended by adding additional data and/or classes. We explain this and other practical limitations in more detail for each recognition module.

Table 1

Perception models and datasets utilized to enable human-robot collaboration. Performance is reported in terms of frames per second (FPS) and prediction accuracy on custom test data, recorded for evaluation.

Perception module	Training			Inference (GTX 1080 Ti)			Prediction accuracy (%)
	Method	Dataset	Dataset size	Model size	Image size	FPS	
Human skeleton detection	Lightweight OpenPose [37]	COCO 2017 [23]	25 GB	1.2 GB	1920x1080	30	91
					1280x720	30	
					960x540	60	
Human action recognition	ST-GCN [60]	NTU-RGB+D [46]	1.3 TB	47 MB	1920x1080	20	87
					1280x720	30	
					960x540	31	
Object and target detection	Detectron2 [58]	Custom [47]	65 GB	0.5 GB	1920x1080	2.6	93
					1280x720	4.5	
					960x540	6.0	

Table 2

Confusion matrix of the object and target detection tool, evaluated on 2340 images with 39530 instances of the 8 classes. Actual classes are shown as column heads and predicted classes as row heads. The prediction accuracy is shown as last column.

Classes	Rockerarm target	Bolt hole	Big pushrod hole	Small pushrod hole	Engine	Bolt	Pushrod	Rockerarm object	Background	Prediction accuracy
Rockerarm target	4802	0	0	0	0	0	0	0	13	99.7
Bolt hole	0	12398	0	1	0	0	0	0	157	98.7
Big pushrod hole	0	0	2234	47	0	0	0	0	193	90.3
Small pushrod hole	0	11	28	2453	0	0	0	0	196	91.3
Engine	0	0	0	0	995	0	0	0	0	100
Bolt	0	0	0	2	0	6451	24	63	504	91.6
Pushrod	0	0	0	1	0	269	2579	47	517	75.6
Rockerarm object	0	0	0	0	0	0	0	3817	25	99.3

Human skeleton detection

The human skeleton detection method (LightWeight OpenPose [37]) with pretrained model [23] is evaluated on a custom test dataset of 1950 images, in which a person performs different actions in the field of view. Human actions included are similar to actions to be recognized in the human action recognition tool. The prediction accuracy of a human skeleton detected correctly, such that it performs human action recognition, was found to be 91%. Fig. 3 depicts the skeleton detection and draws it over the person in the scene. 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 [60]) with pretrained model [46] is evaluated on a custom test dataset of 1950 images, in which a person performs different actions in the field of view, i.e., 'salute' (ID:37), 'put the palms together' (ID:38) and 'cross hands in front' (ID:39). Each action was performed for 30 seconds, leading to >600 images per action. Recognition results were evaluated manually afterwards. Results indicate that a reasonably high prediction accuracy can be achieved (89%, 81% and 91% for the three actions, respectively).

Fig. 3c and 3d depict 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 20 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 datasets for human action recognition mostly cover daily actions [46], not relevant for industrial tasks.

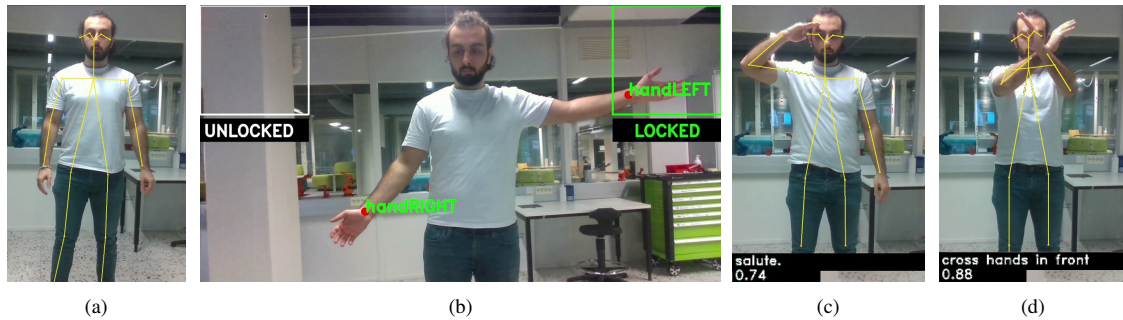


Figure 3: Human visual recognition modules. (a) and (b) depict results of human skeleton detection with the skeleton-based tracker Lightweight OpenPose [37]. (b) demonstrates that skeleton detection can be used for human-robot collaboration by detecting human wrists (handLeft and handRight) in certain image areas. (c) and (d) depict results of human action recognition with the real-time skeleton-based human action recognition framework ST-GCN [60]. Recognized actions are 'salute' (c) and 'cross hands in front' (d), with their corresponding confidence score.

Object and target detection

The object and target detection method (Detectron2 [58]) 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 [47], 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 dataset of around 280,000 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 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 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 in the recorded video⁹. 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.

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, as well

⁹<https://youtu.be/3z3yilDznrY>

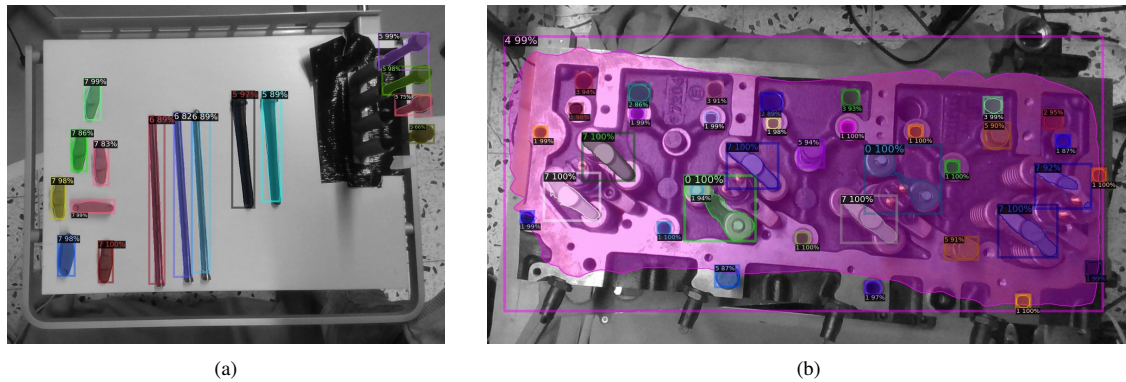


Figure 4: Results of visual perception for object and target detection utilizes Detectron2 [58]. (a) depicts detection of objects (three classes): rocker arms, bolts and pushrods, and (b) depicts detection of targets (five classes): engine, bolt holes, pushrod holes and rocker arm location. Each detection is labeled with the detected class and their corresponding confidence score.

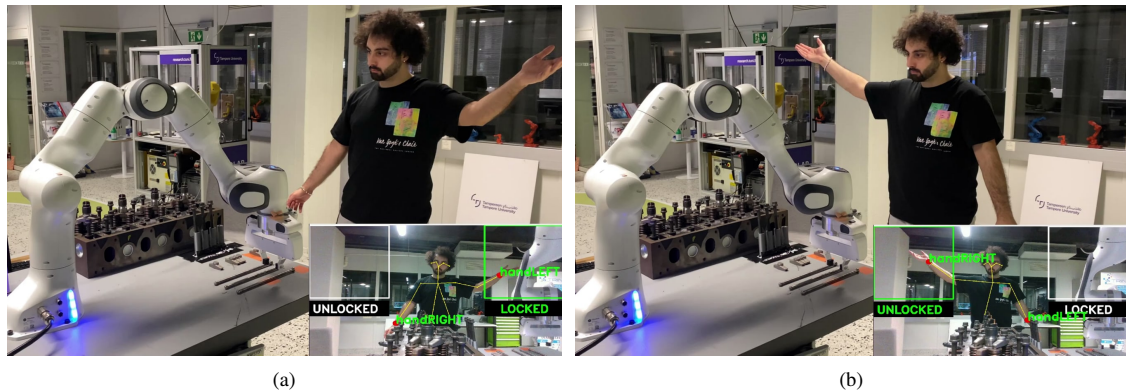


Figure 5: Results of human-robot collaboration experiments. (a) and (b) depict human task coordination by visual detection of the left wrist (handLeft), for halting the robot and performing manual assembly actions (a), followed by right wrist detection (handRight) for resuming robot actions (b).

as in the recorded video⁹, 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, six of the eight rocker arms are placed, however, only five are detected (class 7), while two rocker

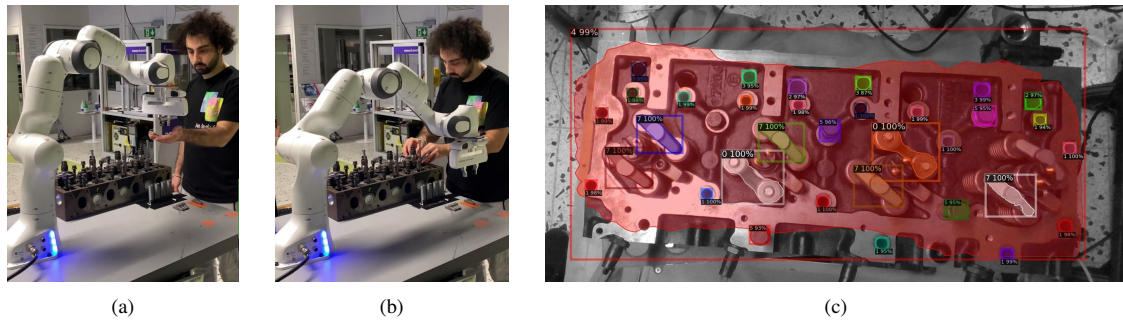


Figure 6: Results of robot-human hand-over and assembly tracking experiments. (a) depicts the hand-over of a rocker arm from robot to human. (b) depicts the human assembly action of the rocker arm by the human, while the robot fetches another rocker arm. (c) depicts the assembly tracking results, with 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). Each detection is labeled with the detected class and their corresponding confidence score.

arm locations are detected (class 0) and thus one is not detected. In addition, four bolts are placed and detected (class 5), while sixteen bolts holes are detected (class 1) and thus empty. In total, 22 bolts should be assembled to the engine block, meaning two bolt holes or bolts are not detected.

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 [29], [49]. Collecting data is complex, and expensive in resources and equipment, even when synthetic data generation approaches exist [36] [50]. 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 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 [60]) and/or rely on another detection tool as input (e.g., skeleton detection in the case of [60]). 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 [53, 63].

Hardware limitations concern the computation hardware and the visual sensors utilized. Naturally, a GPU similar to the ours (Nvidia GTX 1080 Ti) needs to be used to achieve the same performance as reported in Table 1. However, the toolkit is compatible for both GPU and CPU systems to train and run all models, limiting only the run-time performance. Placement of the visual sensors is challenging to accommodate due to the different moving parts in the scene, i.e., robot and human. In our case, the visual sensors were placed on the robot end-effector and behind the robot facing to the person. This led to situations where objects are either not in the camera's field of view or humans are occluded by the robot, limiting the time that suitable perception can occur. While different solutions can be developed that would better distribute cameras or avoid occlusion [43, 62], our camera setup did not cause limitations in performance or drawbacks in fluency of collaboration, as demonstrated in the recorded video⁹.

Integration effort - The resources and effort needed to develop, train and deploy perception models for industrial use, is considerable. Even when robust and reliable pre-trained models are to be integrated, still effort is needed to comply tools to existing software frameworks with its own datatypes and formatting. While ROS² has taken first steps to enable this for robotics, computer vision tools are typically disconnected from this. OpenDR [38] 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 [47]. A collection of

200 RGB images were 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 on a Nvidia GTX 1080 Ti GPU), 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 than 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] Alcér, 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., Obrdzalek, S., Matas, J., Lourakis, M., Zabulis, X., 2017. T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects, in: IEEE Winter Conference on Applications of Computer Vision (WACV), 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] Li, S., Zheng, P., Liu, S., Wang, Z., Wang, X.V., Zheng, L., Wang, L., 2023. Proactive human-robot collaboration: Mutual-cognitive, predictable, and self-organising perspectives. Robotics and Computer-Integrated Manufacturing 81, 102510. doi:10.1016/j.rcim.2022.102510.
- [23] 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), pp. 740–755.
- [24] 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.
- [25] 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.
- [26] 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.
- [27] 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, pp. 21–37.
- [28] 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.
- [29] Marcus, G., 2018. Deep learning: A critical appraisal. arXiv preprint arXiv:1801.00631.
- [30] 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., Rudnick, 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.cs1.2021.101255.
- [31] 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.
- [32] 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.
- [33] 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.
- [34] 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.
- [35] 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.
- [36] Nowruzi, F.E., Kapoor, P., Kolhatkar, D., Hassanat, F.A., Laganieri, 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.
- [37] Osokin, D., 2018. Real-time 2D multi-person pose estimation on cpu: Lightweight openpose. doi:10.48550/ARXIV.1811.12004.
- [38] 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), pp. 12479–12484. doi:10.1109/IROS47612.2022.9981703.
- [39] 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).
- [40] 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.
- [41] 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).
- [42] 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.
- [43] Robinson, N., Tidd, B., Campbell, D., Kulić, D., Corke, P., 2022. Robotic vision for human-robot interaction and collaboration: A survey and systematic review. ACM Journal of Human-Robot Interaction 12, 1–66. doi:10.1145/3570731.
- [44] 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.
- [45] 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.
- [46] 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.
- [47] Sharma, G., Pieters, R., Angleraud, A., 2023. Engine assembly dataset. doi:10.5281/zenodo.7669593.
- [48] 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.

- [49] S nderhauf, 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.
- [50] 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.
- [51] 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)*, pp. 197–212.
- [52] 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.
- [53] 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.
- [54] 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.
- [55] Wang, L., Gao, R., Vancza, J., Kr ger, J., Wang, X., Makris, S., Chrysolouris, G., 2019. Symbiotic human-robot collaborative assembly. *CIRP Annals* 68, 701–726. doi:10.1016/j.cirp.2019.05.002.
- [56] 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.
- [57] 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.
- [58] Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R., 2019. Detectron2. <https://github.com/facebookresearch/detectron2>.
- [59] 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.
- [60] 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*.
- [61] 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.
- [62] 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.
- [63] 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.
- [64] Zeng, A., Song, S., Nie ner, 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

Co-speech Gestures for Human-Robot Collaboration

Akif Ekeekli¹, Alexandre Angleraud¹, Gaurang Sharma¹ and Roel Pieters¹

Abstract— Collaboration between human and robot requires effective modes of communication to assign robot tasks and coordinate activities. As communication can utilize different modalities, a multi-modal approach can be more expressive than single modal models alone. In this work we propose a co-speech gesture model that can assign robot tasks for human-robot collaboration. Human gestures and speech, detected by computer vision and speech recognition, can thus refer to objects in the scene and apply robot actions to them. We present an experimental evaluation of the multi-modal co-speech model with a real-world industrial use case. Results demonstrate that multi-modal communication is easy to achieve and can provide benefits for collaboration with respect to single modal tools.

I. INTRODUCTION

Fluent interaction between human and robot requires reliable perception to capture the commands of a person. While recent approaches in deep learning [1] have established impressive tools to detect e.g., human pose, gestures and speech, single tools alone can not always convey easily the commands intended [2]. Reasons for this are the limited expressions available for different modes of communication and the limitations in perception performance. Human hand gestures, for example, contain much less information content than speech. On the other hand, gesture detection can be done much quicker than speech recognition, leading to a faster response time. These conflicting properties motivate to combine multiple perception tools into a single multi-modal detection model that utilizes communication from human to robot for assigning tasks and coordinating the collaboration. In this work we compare different perception tools and analyse them with respect to their suitability for human-robot collaboration. A co-speech gesture model is then developed that combines speech, human hand gestures and object detection to achieve effective communication of desired robot tasks, such as picking human-specified objects and robot to human hand-overs (see Fig. 1). The developments are intended for industrial human-robot collaboration where a collaborative robot shares its tasks, and works in close collaboration with, a human operator. Our contributions are:

- Human speech and hand gesture perception methods to command robot actions
- Co-speech gesture model that combines human natural speech and hand gestures to command robot actions
- Experimental evaluation of the co-speech gesture model in an industrial human-robot collaborative use case

¹Cognitive Robotics group, Unit of Automation Technology and Mechanical Engineering, Tampere University, 33720, Tampere, Finland; firstname.surname@tuni.fi

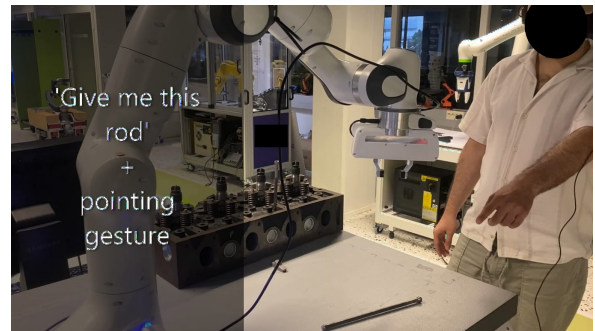


Fig. 1: Co-speech gesture model that combines a speech phrase, human gesture detection and object perception to command robot actions.

II. RELATED WORK

A. Human-Robot Collaboration

Collaboration between human and robot is often targeted for industrial manufacturing [3], as both robot and human have unique skills that complement each other. Different interfaces that enable the collaboration have been analyzed, providing clear directions on how the collaboration benefits the tasks [4]. Approaches include voice processing, gesture recognition, haptic interaction, and even brainwave perception. Often machine [5] and deep [6] learning are used as enabling perception tool [1] to classify and recognize the person and objects in the environment [7].

B. Human Perception

Visual detection of a person in the scene has been an active area of research [8]. Different visual modalities have been utilized [9], such as RGB and depth information [10]. Multi-modal approaches that utilize RGB-D data are popular as well [11]. Human pose estimation goes a step further than human detection by estimating the 3D pose of a human and their individual skeleton joints [12], which can be used as input for gesture detection. Utilizing speech for commanding robots has been demonstrated with short verbal commands for task coordination [13] and task programming [14]. As extension to short speech commands, natural language as instructions to robots has been used for planning [15] and allocation [16] of tasks to be performed by the robot.

Multi-modal human-robot collaboration using gestures and speech simultaneously has been demonstrated for a human interacting with the humanoid robot NAO in [17], where short phrases and gestures are utilized to indicate human actions. Collaboration between a robot arm and a human

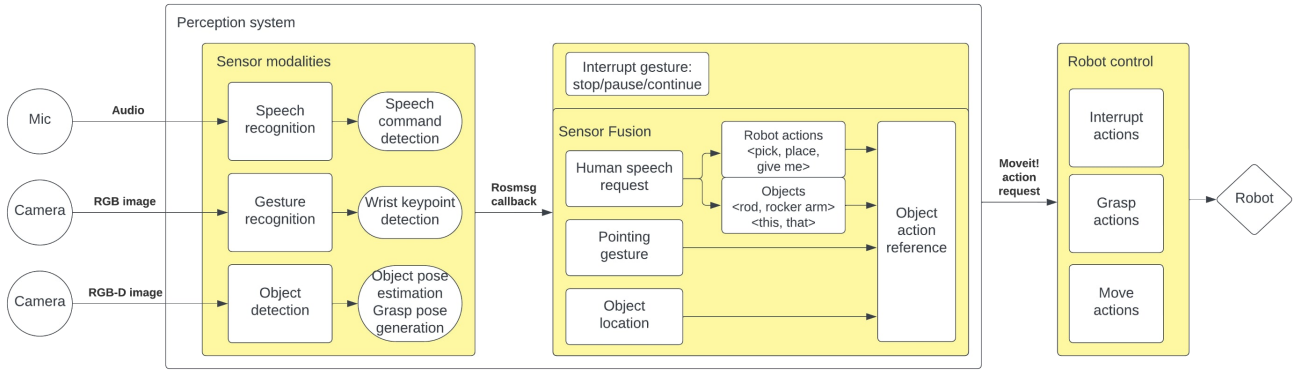


Fig. 2: Co-speech gesture model that takes input from speech commands, gesture recognition and object detection to generate robot actions for human-robot collaboration. Sensor fusion enables the human to refer to specific objects (<rod, rocker, arm, this, that>) and apply actions to them (<pick, place, give me>).

worker is also demonstrated in [18], where a set of gestures and speech commands are perceived individually to produce the same input for robot actions. As comparison, our work considers an industrial scenario with a collaborative robot where speech phrases and gestures are combined to assign tasks to the robot.

III. METHODS AND TOOLS

A. Perception Tools

The perception tools utilized in this work are integrated in a common framework for isolated and human-robot collaborative tasks. For human perception, Lightweight OpenPose, a human skeleton detection tool [12] is used, which takes images (RGB) as input and returns skeleton node points as output. For interaction, the wrist node of the skeleton is taken and, when presented in a certain image area, serves as trigger for robot actions (e.g., stop, continue) or refers to certain objects in the scene (i.e., detected objects pointed to). In the latter case, the detected object that is closest to the wrist node is selected for robot action execution. Speech recognition is enabled by Vosk [19] for the detection of predefined input commands and phrases. This set of words and sentences relate to available actions of the robot and locations in the scene, as described in Table I. The model is configured by filtering out unnecessary words that are unsuitable for robot instructions. Objects in the scene are detected by a neural network model (Detectron2 [20]) trained on a custom dataset collected for the use case [21].

B. Multi-modal Perception Methods

The perception tools can be used in different ways to allow for sensor redundancy, sensor multi-modality and sensor information fusion, as follows.

- **Sensor redundancy** - multiple sensors are used to command the same robot actions, e.g., speech or hand gesture to stop robot motion
- **Sensor multi-modality** - different sensor modalities are used to command individual robot actions, e.g.,

speech provides the robot actions, vision provides object locations in the scene and detects human gestures

- **Sensor-fusion** - different sensor modalities are combined to command a single robot action, e.g., speech provides robot action, vision provides specific object location as pointed to by the human

While sensor redundancy and multi-modality is supported and demonstrated in Section IV, we emphasize our contributions to the fusion of multiple sensor outputs into a single robot command, as explained in the following section.

C. Co-speech Gesture Model

The single-modal visual and speech perception models are fused into a multi-modal perception model by combining speech commands, pointing gestures and object detection (see Fig. 2). Several examples of these co-speech gestures are described in Table I. The human can refer to individual objects in the scene by speech (e.g., <rod>, <rocker arm>) and pointing to them, and apply specific robot actions by speech commands (e.g., picking with <pick>, placing with <place>, robot to human hand-over with <give>).

Depending on the object, different robot actions are possible, as specified beforehand. For example, objects can be picked up from the table, placed in specified locations and handed over to the person. Object detection returns a list of objects in the scene, which can be verbally referred to by their class. Pointing gesture detection allows to refer to specific objects in the scene by relating the pointing gesture location to detected object locations. Robot actions are therefore commanded by specific action verbs and object classes, complimented by gestures to provide fine-grained object references (see Fig. 2).

IV. EXPERIMENTAL RESULTS

A. Industrial Use Case

The considered use case replicates an industrial assembly task that in current situation is done manually by human operators. The solution we propose introduces a collaborative robot as assistive tool to the assembly station, under control

TABLE I: Perception methods' input and output

Method	Input	Output
Wrist detection	RGB image of the scene (human front-facing) Human gesture by moving wrist to certain image location	Robot stop/continue actions
Speech recognition	Robot action commands: <pick, place, give, go, stop, pause, continue> Workspace commands: <rod, home, arm, me> Human speech requests: <place rod>, <go home>, <give me another rocker arm>, <pick up the last rod>	Robot motion Gripper actions Robot to human hand-over Robot stop/continue actions
Object detection	RGB image of the scene (top-down)	Detected objects in the scene Valid target location for robot
Co-speech gesture	<pick rod> + pointing gesture + object detection <give me this rod> + pointing gesture + object detection <give me that rocker arm> + pointing gesture + object detection	Robot motion Gripper actions Robot to human hand-over

of the person. This means that the assembly work is coordinated by the human, with the robot assisting in tasks that the human decides. Available robot actions are to move to certain locations in the work space, pick objects that are detected on the table, place objects to specified locations or hand them over to the human. In addition, coordinated actions include the stopping and continuing of robot actions during execution, for human visual inspection of the objects placed by the robot. Human commands can be communicated by hand gestures and/or speech, with different levels of functionality as described in Table I. The setup for experiments is depicted in Fig. 3 and includes two cameras (Intel Realsense D435) for visual perception (one front-facing for wrist detection; Fig. 3(b) and one top-down for object detection; Fig. 3(c)) and a microphone for speech recognition. Computation is performed on a standard Desktop PC running Ubuntu Linux with Nvidia GTX 1080 Ti GPU, and all robot (Franka Emika) communication and control utilizes ROS. All tools are open-source available to utilize or replicate: <https://github.com/opendr-eu/opendr>.

B. Human Gesture Detection

Results for the visual wrist detection tool are depicted in Fig. 3(b), which highlights both detected human wrists. When one of the wrists is detected inside one of the squares, this is taken as trigger for referring to certain robot actions or objects in the scene. For example, to stop robot motion, the left wrist should be detected in the top left square and to continue robot motion, the right wrist should be detected in the top right square. Pointing gestures are interpreted in a similar manner. When the human points to a certain object, first the left or right wrist needs to be detected in either of the lower two squares in the image, after which the location to the closest detected object is determined. Performance of the skeleton detection tool has been reported in the original paper [12]. In our use case the detection accuracy of the wrists inside a square is consistent around 90%, as assessed from 20-second interval tests for different squares. This is satisfactory for effective collaboration.

C. Object Detection

Results of visual object detection are depicted in Fig. 3(c), which has the different detected objects annotated by colored bounding boxes (yellow for the rocker arms and blue for the rods). As objects are detected in image space, careful

calibration of both cameras ensures the detected objects can be picked from the table and that pointing gestures can refer to the same object in both camera frames. In our use case the detection accuracy of all classes is over 90%.

D. Speech Recognition

Results of speech recognition were found satisfactory, as in most cases the spoken commands are recognized correctly. Performance, as reported in the original paper [19], depends on the language skills of the person giving commands, as in certain cases non-native English speakers had to speak more clear to achieve correct speech recognition. Besides the speech recognition itself, the speech tool was improved by including a voice activity detector and a time-delay filter (0.5 seconds) to consider the natural pause in human speech. This resulted in a delay of ≈ 1.9 seconds between a verbal command and the recognized speech (average of 50 trials with different commands).

E. Co-speech Gesture Model Performance

The co-speech gesture model has all three perception models running in parallel, decreasing slightly the running performance of the skeleton detection tool (i.e., 24 fps with image size of 1920×1080). Object detection achieves a frame rate of 4.5 fps with image size of 1280×720 . Extended experiments were performed to test the co-speech tool in a collaborative assembly scenario. This included a human and robot performing assembly steps to an engine, with parts that are either mounted by the person or by the robot. Parts assembled by the person are picked by the robot from the table and handed over to the human, and parts assembled by the robot are picked by the robot from the table and directly mounted to the engine. Coordination of the tasks and requesting robot actions is done by the person via the co-speech gesture model. In addition, the human can halt and continue robot tasks at any time, by both gesture (i.e., raise left/right wrist) and speech commands (<stop>, <pause>, <continue>).

Single commands - Fig. 4(a) and (b) depict the human commanding a stop gesture and a continue gesture, respectively. Fig. 4(c) shows the human commanding the robot to move to its 'home' configuration by the phrase <ok, go home>. For this, the home location is preprogrammed in the software scripts.

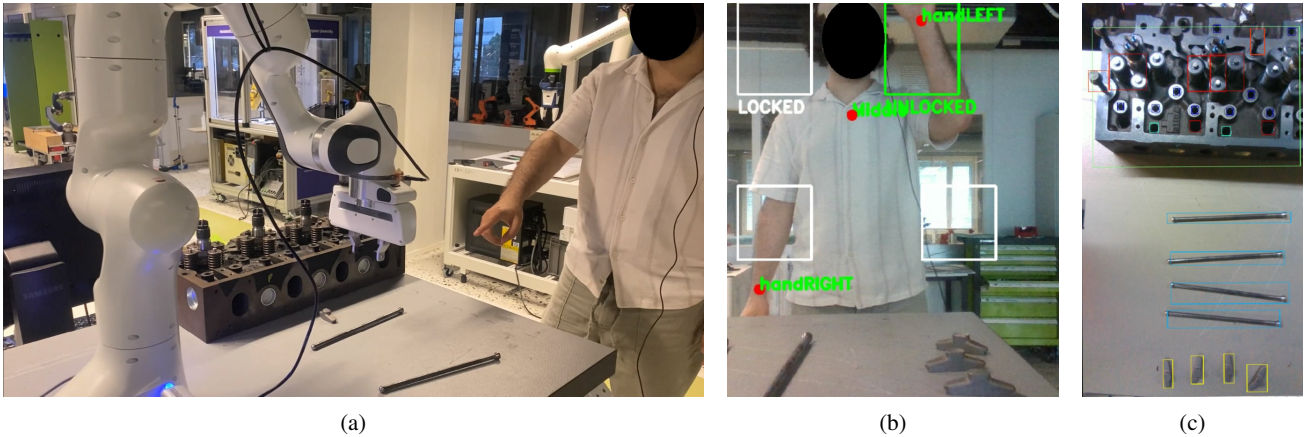


Fig. 3: Experimental setup with a human pointing at an object for robot picking (a). One camera is human front-facing to capture human hand gestures (b), one camera is mounted on the robot (eye-in-hand) for object detection on the table (c).

Speech phrases - Fig. 5 depicts how human speech alone can be utilized to command robot actions, by the phrase <give me another rocker arm>. From the recognized speech, the tool extracts relevant words and connects these to robot actions and objects in the scene. In this case <give me> refers to a robot to human hand-over, <rocker arm> refers to the rocker arm class in the object detection model, and <another> implies any of the detected rocker arms, meaning the first in the returned detection list. As a result, the command phrase initiates all required robot actions and starts executing them one-by-one, as shown in Fig. 5(a)-(c).

Co-speech commands - Fig. 6 depicts examples of the co-speech gesture model that utilizes a human speech phrase and pointing gesture to achieve robot actions applied to specified objects in the scene. In this case, as a pointing gesture is detected by the wrist detection tool, the closest specified object to the human wrist is selected for the robot actions. A video¹ of the co-speech gesture model demonstrates all commands from Fig. 4-6. This shows the collaborative tasks, where the human coordinates the actions of the robot with four pick and place actions and four robot to human hand-overs. Human visual inspection is done after object placement by stopping robot motion with a speech command. In total, the experiment includes over 20 speech commands and seven co-speech gestures to coordinate the shared task.

V. DISCUSSION AND LIMITATIONS

Sensor redundancy enables different modalities to command the same robot action. This was demonstrated for stopping and continuing robot motion and actions by hand gestures (see Fig. 4) and by speech commands. Few differences were observed resulting from the experimental evaluations. While hand gestures can be detected at relatively high rate (>24 FPS), it can take several image frames before a correct prediction occurs. On the other hand, speech commands can

have considerable delay even when a first verbal command is correctly recognized. Regardless, the benefit of utilizing both modalities, even with such delay, is evident in situations when an operator is doing manual actions.

While in most cases the co-speech gesture model achieves the intended robot commands and collaboration, some limitations are identified. First, detection of the human wrist in a specific image location requires careful human hand motion. As alternative, human actions [22] or hand gestures could be recognized directly from a dedicated model [23]. In our case, inference time and detection accuracy were the main reasons for utilizing a skeleton detection model instead. Second, the relation between human pointing and objects in the scene needs precise camera calibration, such that the same object is referred to in both images. This can be circumvented by using a single camera for both visual perception tools, with RGB and depth perception functionalities.

VI. CONCLUSIONS

This work investigated how multiple perception tools can be utilized and combined for effective human-robot collaboration. Human hand gestures and speech, as well as object detection, provide the input for robot actions, as coordinated by a human operator. Single modal perception serves to command basic robot actions (stop, continue) by gesture or speech. A co-speech gesture model is developed that combines human speech phrases, pointing gestures and object detection to command robot actions (pick and place, robot to human hand-overs) to specified objects in the scene. Experimental results demonstrate that co-speech gestures can be easily utilized for coordinating a shared collaborative task between human and robot.

ACKNOWLEDGEMENTS

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

¹https://youtu.be/b_ISrh0lcC8

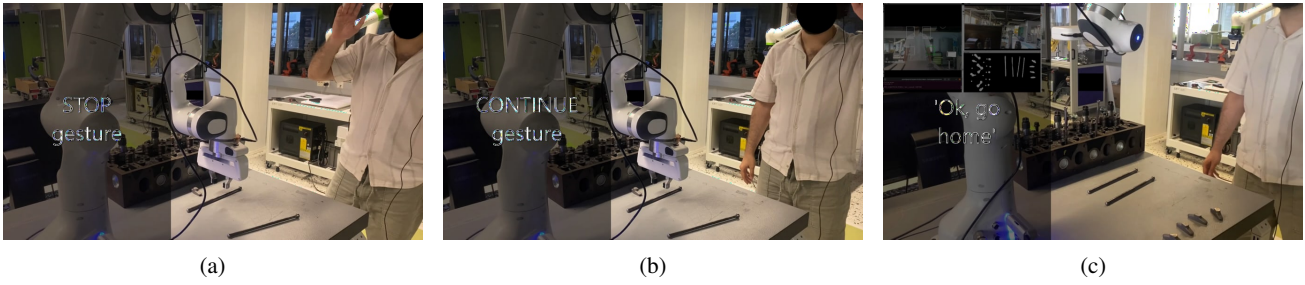


Fig. 4: Single command gestures stop (a), continue (b) and speech (c).

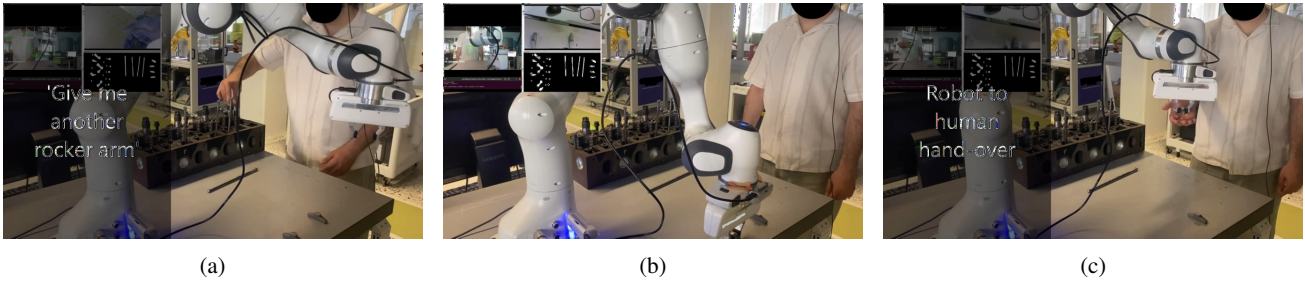


Fig. 5: Speech phrase to achieve robot to human hand-over.

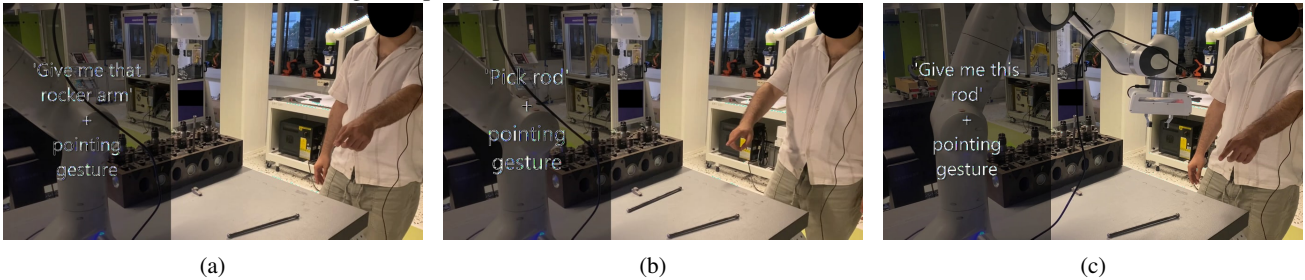


Fig. 6: Co-speech gestures to achieve specified robot actions to objects.

REFERENCES

- [1] N. Robinson *et al.*, “Robotic vision for human-robot interaction and collaboration: A survey and systematic review,” *ACM Trans. Hum.-Robot Interact.*, vol. 12, no. 1, pp. 1–66, 2023.
- [2] S. Gross and B. Krenn, “A communicative perspective on human-robot collaboration in industry: Mapping communicative modes on collaborative scenarios,” *Int. J. of Social Robotics*, pp. 1–18, 2023.
- [3] V. Villani, F. Pini, F. Leali, and C. Secchi, “Survey on human-robot collaboration in industrial settings: Safety, intuitive interfaces and applications,” *Mechatronics*, vol. 55, pp. 248–266, 11 2018.
- [4] L. Wang *et al.*, “Symbiotic human-robot collaborative assembly,” *CIRP Annals*, vol. 68, pp. 701–726, 2019.
- [5] F. Semeraro, A. Griffiths, and A. Cangelosi, “Human-robot collaboration and machine learning: A systematic review of recent research,” *Robot. Comput. Integr. Manuf.*, vol. 79, p. 102432, 2023.
- [6] N. Sünderhauf *et al.*, “The limits and potentials of deep learning for robotics,” *Int. J. Rob. Res.*, vol. 37, pp. 405–420, 4 2018.
- [7] J. Fan, P. Zheng, and S. Li, “Vision-based holistic scene understanding towards proactive human-robot collaboration,” *Robot. Comput. Integr. Manuf.*, vol. 75, p. 102304, 6 2022.
- [8] A. Zacharaki, I. Kostavelis, A. Gasteratos, and I. Dokas, “Safety bounds in human robot interaction: A survey,” *Safety Science*, vol. 127, p. 104667, 7 2020.
- [9] T. Linder, N. Vaskevicius, R. Schirmer, and K. O. Arras, “Cross-modal analysis of human detection for robotics: An industrial case study,” in *IEEE Int. Conf. Intell. Robots Syst.*, 9 2021, pp. 971–978.
- [10] E. Magrini *et al.*, “Human-robot coexistence and interaction in open industrial cells,” *Robot. Comput. Integr. Manuf.*, vol. 61, p. 101846, 2 2020.
- [11] C. R. Qi *et al.*, “Frustum pointnets for 3D object detection from RGB-D data,” in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.
- [12] D. Osokin, “Real-time 2D multi-person pose estimation on CPU: Lightweight OpenPose,” *arXiv preprint arXiv:1811.12004*, 2018.
- [13] A. Angleraud *et al.*, “Coordinating shared tasks in human-robot collaboration by commands,” *Front. Robot. AI*, vol. 8, 10 2021.
- [14] T. B. Ionescu and S. Schlund, “Programming cobots by voice: A human-centered, web-based approach,” *Procedia CIRP*, vol. 97, pp. 123–129, 2021.
- [15] A. Boteanu *et al.*, “A model for verifiable grounding and execution of complex natural language instructions,” in *IEEE Int. Conf. Intell. Robots Syst.*, 2016, pp. 2649–2654.
- [16] J. K. Behrens *et al.*, “Specifying dual-arm robot planning problems through natural language and demonstration,” *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2622–2629, 2019.
- [17] P. Bremner and U. Leonards, “Efficiency of speech and iconic gesture integration for robotic and human communicators—a direct comparison,” in *IEEE Int. Conf. Robot. Autom.*, 2015, pp. 1999–2006.
- [18] H. Chen, M. C. Leu, and Z. Yin, “Real-time multi-modal human-robot collaboration using gestures and speech,” *J. Manuf. Sci. Eng.*, vol. 144, no. 10, p. 101007, 2022.
- [19] Alpha Cephei, “Vosk Speech Recognition Toolkit,” <https://github.com/alphacep/vosk-api>, 2023.
- [20] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, “Detectron2,” <https://github.com/facebookresearch/detectron2>, 2019.
- [21] G. Sharma, R. Pieters, and A. Angleraud, “Engine assembly dataset,” <http://dx.doi.org/10.5281/zenodo.7669593>, Feb. 2023.
- [22] S. Yan, Y. Xiong, and D. Lin, “Spatial temporal graph convolutional networks for skeleton-based action recognition,” in *AAAI Conf. on Artificial Intelligence*, 2018.
- [23] O. Mazhar *et al.*, “A real-time human-robot interaction framework with robust background invariant hand gesture detection,” *Robot. Comput. Integr. Manuf.*, vol. 60, pp. 34–48, 12 2019.