G. Ansaldo

A Multi-Agent Simulation Environment for Human-Robot Collaboration in an Industrial Setting

TU Delft

A Multi-Agent Simulation Environment for Human-Robot Collaboration in an Industrial Setting

By

G. Ansaldo

Research Assignment

in partial fulfilment of the requirements for the degree of

Master of Science
in Mechanical Engineering

at the Department Maritime and Transport Technology of Faculty Mechanical, Maritime and Materials Engineering of Delft University of Technology

Student Number:	5350859
MSc Track:	Multi-Machine Engineering
Report Number:	2022.MME.8594
Supervisors:	Dr. F. Schulte
	Dr. B. Heydari
Date:	January 12, 2021

TU Delft

Contents

List of Figures

List of Tables

Introduction

The study of collaborative processes where humans and robots jointly work to achieve a common goal is known as Human-Robot Collaboration (HRC). Alternatively stated, HRC is defined in the context where humans, robots, and the environment come to contact with each other and form a tightly coupled dynamical system to accomplish a task (Bauer, Wollherr, and Buss 2008). Collaborative robots, or cobots, are specifically designed robots that are intended to interact with human subjects. As opposed to typical industrial robot applications in which robots are isolated from humans, cobot applications involve human interaction. Manufacturing, industrial, and logistics environments make up the majority of cobot implementations, but cobots can also be used in a wide range of other sectors, from agriculture to medicine to pharmaceuticals. With an ever so increasing popularity in HRC, the cobot market is expected to significantly grow over the coming decade. According to global tech market advisory firm ABI Research, the market was valued at $475 million in 2020 and will expand to $600 million in 2021 and $8 billion by 2030, with a projected Compound Annual Growth Rate (CAGR) of 32.5% (ABI Research 2021).

(a) (b) (c)

Figure 1: Collaborative Robots (a) KUKA LBR iiwa (b) Sawyer Robot (c) ABB Yumi

Traditionally, industrial robots are used to replace laborers who are assigned to non-ergonomic tasks (Vysocky and Novak 2016). For instance, this could include manipulation with heavy payloads, dangerous tasks, or monotonous operations which are uncomfortably repetitive or demand high accuracy. However, there still exist operations that require humans to operate such as complex assembling. According to a report by A.T. Kearney and Drishti, factory workers perform 72% of tasks, and they generate almost an equal proportion of value (Hu et al. 2018). Indeed, complex assembly tasks require human dexterity and the ability to deal with unpredictable situations. Despite that, the rise of Reinforcement Learning (RL) methods and the surge of cobots in the industry could help humans to perform tasks faster and more efficiently, reducing cycle times as well as errors.

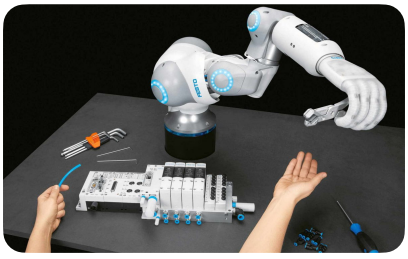


Figure 2: Example of Human-Robot Collaboration for an Assembly Task

RL is a machine learning technique that allows agents to learn from their actions and experiences in an interactive environment by trial and error. The RL process requires a tremendous amount of "trial and error" episodes, or interactions with an environment before a good policy can be learned. For this reason, the use of simulations is essential to achieve results in a cost-effective and timely manner. Many simulation environments are currently available for RL research. Some of the most common and popular environments are OpenAI Gym, Meta-World, and DeepMind Control Suite. These benchmarks are excellent for training RL policies in different scenarios. However, human-robot environments have yet to be fully explored and could be of enormous help in the field of HRC.
This project assists in expanding the current RL environment library by introducing an HRC setting. On a deeper note, this project makes an effort at designing and implementing a simulation environment where a cobot and a human work together in order to achieve a common goal. Aspects of HRC, RL, simulation, and order picking are combined in order to develop a so called 'reaching' task. Please refer to the following URL for details regarding the code developed for this project and video demonstrations of the designed task: `https://github.com/gansaldo/reaching-task`

This paper is divided into three parts as shown in Figure 3. Part I, the Conceptual & Technical Research Design, provides a clear picture of how the research is designed and carried out. A problem formulation is presented, followed by the objective of the research project. Consequently, research questions alongside methods of research are presented. Part II is the Design Stage, where the development of the simulation environment is achieved. Finally, Part III is the Implementation/Validation Stage, in which the designed environment is implemented in a real robot and thus validated.

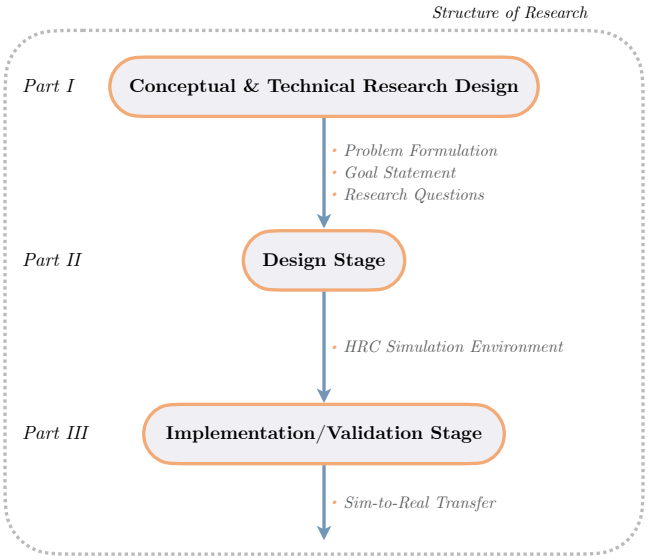


Figure 3: Research Structure and Paper Division

Part I

Conceptual & Technical Research Design

1 Problem Analysis

A concise literature review is provided followed by the problem statement. The literature review below provides an overview of current knowledge regarding HRC and simulation environments, sim-to-real, and order picking allowing to identify relevant theories, methods, and gaps in the existing literature.

1.1 Literature Review

1.1.1 HRC & Simulation Environments

Current manufacturing companies' shop floors are being revolutionized by the introduction of HRC, a key technology of Industry 4.0. In the industrial world, collaborative robots, or cobots, work alongside humans in a so-called cyber-physical production system, where human's unique abilities and smart machines' strengths are combined (Gualtieri, Rauch, and Vidoni 2021). Developing human-robot interaction scenarios strongly depends on the situation of study in addition to being an extremely dynamic process. Indeed, when a human interacts with a robot, its actions are dependent on the observed stimuli, thus, changes in the environment are often unavoidable. As a result, it is impossible to perform repeatable testing and validation of interaction scenarios in real environments (Schmitz, Hirth, and Berns 2010). For this reason simulation environments are required in order to allow robots to safely make and learn from mistakes without putting real people at risk (Erickson et al. 2020). In fact, physics-based simulations can perform thousands of human-robot trials in a few hours providing models that represent a wide range of human body shapes, weights, and physical capabilities. There are currently several toolkits available that show the significance and value of simulations for both robotic tasks as well as RL research (Savva et al. 2019; Kolve et al. 2017; Fan et al. 2018). However, a standard toolkit that is currently used for RL research is OpenAI Gym.

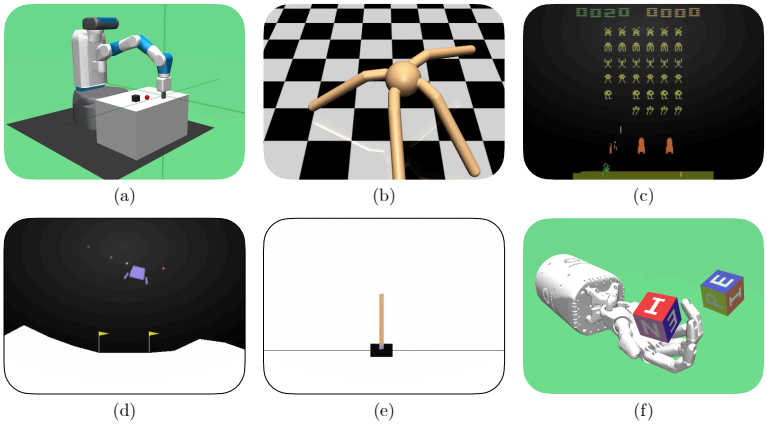


Figure 4: Examples of some OpenAI Gym Environments: (a) Fetch Pick and Place, (b) Ant, (c) Space Invaders, (d) Inverted Pendulum, (e) Hand Manipulate Block

OpenAI Gym includes a set of benchmark problems, a common interface, and comparison tools that facilitate the learning of control policies for single-agent simulations (Brock-

man et al. 2016). However, when it comes to HRC settings, multi-agent environments are necessary for simulating human-robot interactions. Besides being focused on single agents, OpenAI Gym has been used as a foundation for many multi-agent reinforcement learning toolkits such as Arena and PettingZoo (Wang et al. 2019; Terry et al. 2020). Additionally, in terms of robotic applications, OpenAI Gym has been extended to ROS Gazebo, a benchmark in robotics' simulations (Zamora et al. 2016). An interesting application of OpenAI Gym with ROS Gazebo is the one studied by Yamazaki et al. 2014 where a human-robot interaction occurs. However, utilizing ROS Gazebo often requires powerful graphics hardware components especially in the case where multiple agents are to be simulated. Besides the many efforts in including the human aspect in multi-agent reinforcement learning environments, an environment purely dedicated to HRC settings has not been developed until August 2020 when Erickson et al. 2020 introduced Assistive Gym. This is an open-source physics-based simulation integrated into the OpenAI framework for physical human-robot interaction and robotic assistance.

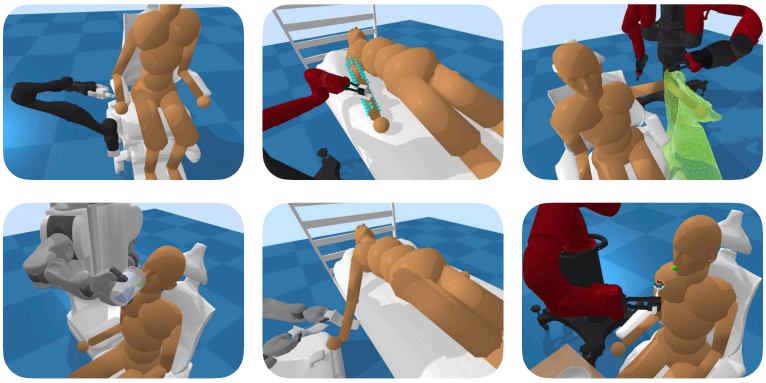


Figure 5: Examples of some Assistive Gym Environments

Assistive Gym's environments are specifically designed for robots to assist people with tasks associated with activities of daily living. Apart from that, Assistive Gym is unique among other robotic simulation environments in that it emphasizes human-robot interaction. Nevertheless, a gap is still present in terms of developing an industrial/logistics HRC environment. For instance, such an environment could depict a complex assembly task where a human and a robot collaborate to assemble a specific object. Besides the current limitations to only six different environments which are solely focused on assisting people in daily tasks such as itch scratching, drinking, and feeding, Assistive Gym can be used as a benchmark to compare control algorithms for robots that interact with people. Therefore, Assistive Gym is a suitable toolkit to expand the HRC library and design a new multi-agent environment.

1.1.2 Sim-To-Real Transfer in RL

As previously stated, training on real robots is extremely time-consuming. Indeed, it is not trivial to set up a system in which a robot can learn a task while also avoiding damaging itself or any items within its reach. As a result, simulations are required for robots to learn effective policies safely. When policies are learned, it is imperative to validate and

apply them in a real environment. Thus, the idea of sim-to-real transfer was introduced. Table 1 below lists some of the most relevant and recent works in this field.

Reference	**Description**	**Simulation Environment**	**Field**		**Task**	**Robot Used**	**Learning Algorithm**	**Human-Robot Collaboration**
			Unspecified	Other				
Kaspar et al. 2014	Sim-to-real transfer for RL without Dynamics Randomization	PyBullet	✓		Peg in Hole	KUKA LBR iiwa	SAC	✗
Matas et al. 2018	Sim-to-real RL for deformable object manipulation	PyBullet	✓		Folding Towel	7DOF Kinova Mico Arm	DDPGfD	✗
Jeong et al. 2019	Modeling Generalized Forces with RL for Sim2Real Transfer	MuJoCo	✓		Manipulation Task	Rethink Robotics Sawyer	MPO	✗
Arndt et al. 2020	Meta Reinforcement Learning for Sim2Real Domain Adaptation	MuJoCo	✓		Hitting Hockey Puck	KUKA LBR	PPO	✗
Breyer et al. 2018	Flexible robotic grasping with Sim2Real RL	PyBullet	✓		Pick and Lift Task	ABB YuMi	TRPO	✗
Vacaro et al. 2019	Sim-to-Real in Reinforcement Learning for Everyone	Unity3D	✓		Target Reach	Sainsmart Robotic Arm	IMPALA	✗
Ding et al. 2020	Sim-to-Real Transfer for Gripper Pose Estimation with GAN	PyBulet	✓		Tactile Sensing Task	Rethink Robotics Sawyer	CNN	✗
Schaffer 2021	Investigating Sim-to-Real Transfer and Multi-Agent Learning in Assistive Gym	Assistive Gym		Healthcare	Drinking Assistance	PR2	PPO	✗
Puthuveetil et al. 2021	Learning to Manipulate Real Blankets Around People via Physics Simulations	Assistive Gym		Healthcare	Uncover Blanket	Stretch RE1	PPO	✓
Peng et al. 2018	Sim-to-Real Transfer of Robotic Control with Dynamics Randomization	MuJoCo	✓		Pushing Task	Fetch Robotics Arm	RDPG	✗
This Paper	Human-Robot Collaboration in an Industrial Setting	Assistive Gym		**Logistics/ Industry**	Reaching Task	Rethink Robotics Sawyer	PPO	✓

Table 1: Analysis of Articles Published for Simulation to Real Robotic Transfer

Currently, models based on reinforcement learning often utilize simulated data to obtain vast quantities of labeled experiences. However, due to the mismatch between simulation environments and real-world scenarios, research has shown that a need to focus on methods for transferring knowledge from simulation to reality is required. As a way to improve simulation realism and better prepare for the real world, domain randomization has been identified as the most widely adopted technique (W. Zhao, Queralta, and Westerlund 2020). Rather than carefully modeling every parameter of the real world, domain randomization emphasizes the need to randomize the simulation in order to cover the real distribution of data regardless of the bias between the model and the real world (Tobin 2019). Alternatively, this technique involves randomizing or introducing noise into the simulation in order to ensure that learned policies are robust to changing environments.

1.1.3 Order Picking Systems & HRC

Order Picking Systems (OPS) are methods designed to increase the efficiency, speed, and accuracy of picking activities. Various companies and warehouses can utilize one or more of these systems to streamline order fulfillment.

In recent years, there has been a growing interest in the area of automation in OPSs, which may be due to its importance, technological progress and the increased use of automation in practice (Jaghbeer, Hanson, and Johansson 2020). More specifically, the categories of throughput, lead time, and operational efficiency have gained much attention (ibid.). Nevertheless, the human characteristics that are often a major determinant of the performance of OPSs have generally been neglected in such research (Grosse, Glock, and Neumann 2017). In spite of this, the advancement of cobots could reintroduce and highlight the importance of Human Factors (HFs) in OPSs. Various papers have shown that

HRC is helpful in increasing the performance of an OPSs (Fager, Sgarbossa, and Calzavara 2021; Coelho, Relvas, and Barbosa-Póvoa 2018). Pasparakis, Vries, and Koster 2021 have made a particular effort in this field and conducted a unique real-effort experiment in a warehouse. In this experiment, a scenario with the *human leading* the robot and one with the *human following* the robot were compared with respect to collaborative productivity, collaborative accuracy, and human pick speed. According to the results, while *human leading* results in better collaborative order picking productivity, *human following* leads to greater collaborative order picking accuracy. Lastly, and most importantly, Pasparakis, Vries, and Koster confirm that competence in manual order picking is directly transferrable to the HRC environment.

1.1.4 Research Gap & Contribution

As mentioned in Section 1.1.1, Assistive Gym is one of the only tools that can be used for benchmarking and developing simulation environments for assistive/collaborative HRC tasks. However, Assistive Gym currently focuses only on environments related to assisting people with impairments. Indeed, extending such an environment to an industrial/logistics HRC tasks has not been explored yet. Moreover as seen in Section 1.1.2 and Table 1, sim-to-real research with regards to Assistive Gym and specifically in the HRC field still needs to be explored. Puthuveetil et al. 2021 looked into transferring an Assistive Gym environment to a real scenario, however a static manikin instead of a rational human was used for the research. Lastly, in Section 1.1.3 it is shown how recent research in OPSs has neglected the importance of human collaboration in system performance. For this purpose, this project aims at bridging the gap between HRC simulation environments, particularly Assistive Gym, and the sim-to-real domain while merging this into OPSs.

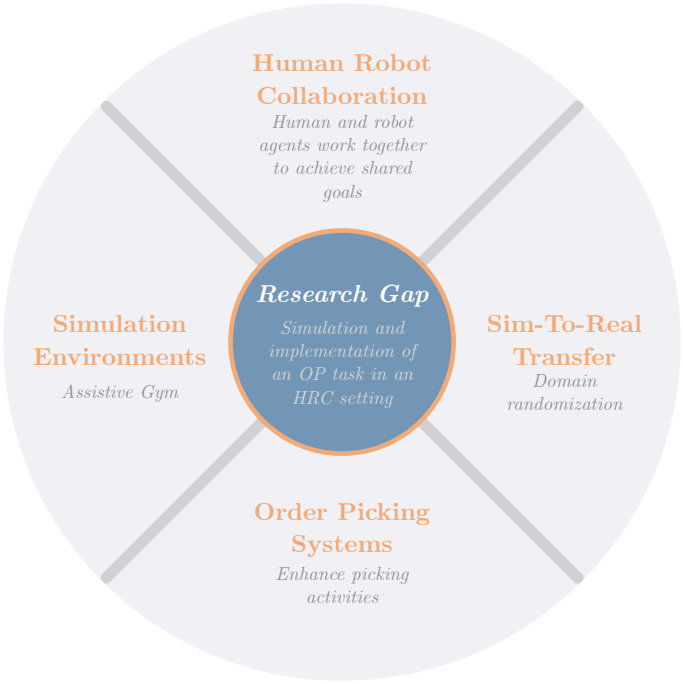


Figure 6: Research Gap Diagram

Explicitly, through this work the following contributions are made:

- An OP environment task is designed under the Assistive Gym package.
- The HRC policy learned during simulation is transferred to a real environment where a human is present.

The benefits of such research include:

- Policies can be learned faster through simulations rather than repetitive trial and error episodes in real life.
- Reduced costs are present if the environment to be studied can be simulated. For instance no human resources are required and no physical environment needs to be utilized for the learning process.
- Since learning happens through simulation, the rate of learning is controllable.
- Safety can be increased drastically since real humans would not be involved in the training process.

1.2 Problem Statement

After contextualizing the problem at hand, a concise problem statement has been developed.

Problem Statement

The recent growth of cobots and RL methods has brought the need for simulation environments that allow the possibility of simulating HRC scenarios. Current simulation environments are able to train RL policies but they often focus on a single agent rather than on multi-agent interactions and HRC settings. Furthermore, the implementation of existing HRC simulation environments into the real world has yet to be fully explored.

1.3 System Description

The task to be designed is a pick, reaching, and place task. For the sake of clarity and simplicity the task will be referred as a "reaching" task for the rest of this document. For this task, the robotic arm picks up an object and hands it into a human/operator's hand. The object to be picked up could be a tool or a package that is out of reach from the human operator. The utilized robotic arm is Sawyer from Rethink Robotics. This is a 7-DoF industrial collaborative robot designed to help out with manufacturing tasks and work alongside humans.

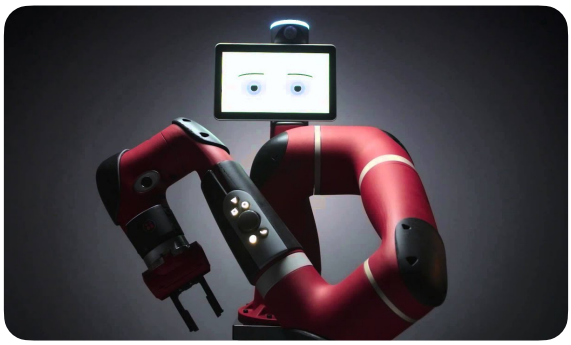


Figure 7: Sawyer Robot

The system to be studied involves a Sawyer robot, a human operator, and an object to be handled. A simple example of the described task is shown in Figure 8 below.

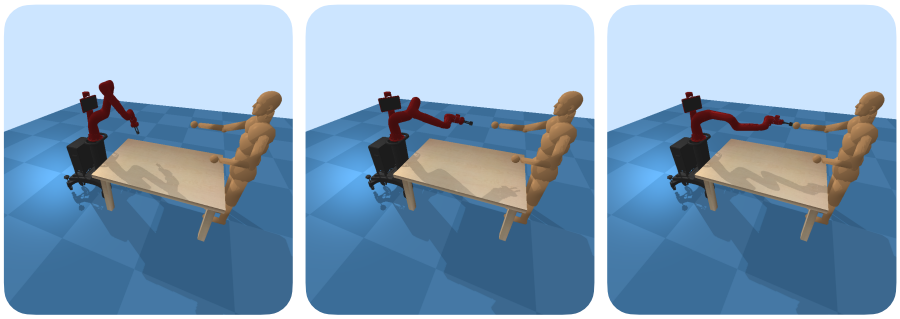


Figure 8: Reaching Task Example, Sawyer Robot Reaching Operators Hand

From the above example, it is clear that depending on the workspace layout, the task can become more or less complex. There are several important aspects to consider, including the position of the object to be picked up, the location of the operator, how the robot releases the object, and how the operator grabs the object. Intelligent control and more specifically RL is utilized to optimize the task at hand. In particular, proximal policy optimization (PPO) is used to train the designed environment. PPO is a class of reinforcement learning algorithms released by OpenAI Gym (Schulman et al. 2020). In addition to being simpler to implement and tune, these algorithms perform comparably well to state-of-the-art approaches. Robots can be taught complex tasks using this form of intelligent control in a relatively short time. Most importantly, reinforcement learning gives this system the ability to adapt to various scenarios.

2 Research Goal

2.1 Cycle Choice

This research project has a practice-oriented nature. Therefore, it was decided to adopt the intervention cycle. The intervention cycle is a predefined set of steps to reach a solution relating to operational problems (Hermans and Schoeman 2015).

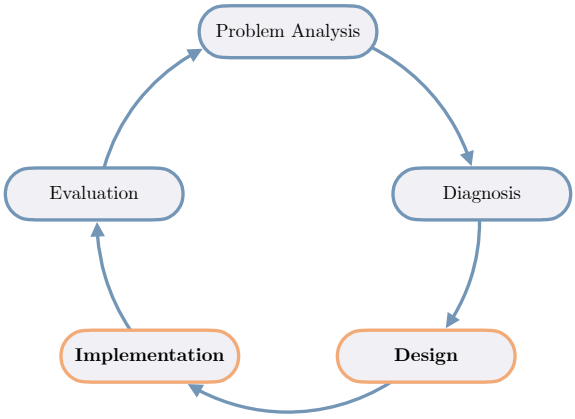


Figure 9: Intervention Cycle

The focus of this project lies in the design and implementation stage. On the one hand, the design phase is focused at the creation of a multi-agent/human-robot simulation environment for industrial tasks. On the other hand, the implementation stage focuses on transferring the simulated environment into a real scenario. In other words, the implementation stage can be seen as a validation of the multi-agent/human-robot simulation environment designed in the previous cycle stage.

2.2 Goal Statement

After having identified a problem statement, the research objective can be formulated.

Goal Statement

The goal of this project is to contribute to the library of simulation environments by ***designing*** *and* ***implementing*** *a multi-agent industrial environment where a cobot and a human collaborate to achieve a common goal.*

3 Research Questions

In order to acquire the data and knowledge necessary to achieve the design goal, a set of research questions has been formulated. Subsequently, secondary questions have been introduced to contribute to the answering of the main core questions. Two are the main research questions. On the one hand, the first question pertains the design of a HRC environment for an industrial setting. On the other hand, the second main research question focuses on implementing and validating the designed environment through sim-to-real. The following questions have been formulated:

Research Questions

1. *[Design Question]* How can the design of a human-robot collaboration environment be realized?
   (a) What are the characteristics of the human-robot environment that is to be designed?
   (b) What are the limitations and assumptions related to the design of the human-robot environment of study?
   (c) What are the characteristics of the Observation Space, Action Space, and Reward of the environment to be designed?
2. *[Implementation/Validation Question]* How can the simulated new environment be transferred to a real robot?
   (a) What steps need to be taken in order to synchronize simulation and real world parameters?
   (b) What state of the art sim-to-real techniques can be used to improve the transfer?

Important to note that the project is divided in different parts. Part II focuses on answering the first main research question (Design Question), while Part III relates to the second main research question (Implementation/Validation Question).

4 Research Methodology

The research is focused on designing and implementing a multi-agent reinforcement learning (MARL) environment where a cobot and human collaborate to achieve a common objective. In order to achieve said objective, the computer programming language python is used in combination with the physics-based simulation framework Assistive Gym. Research questions are answered with the help of literature, experimentation, and programming. Table 2 shows a clear overview of the research methods that are used to carry out the project.

Question		Deliverable	Strategy	Tool
1	(a)	Human-Robot Collaboration Simulation Environment for an Industrial Task	Experimentation Programming Literature	OpenAI Gym Assistive Gym Python
	(b)			
	(c)			
	(d)			
2	(a)	Implementation of the Designed Environment (i,e, collaborative assembly task)	Experimentation Programming Literature	Python Sawyer Robot Computer Vision (ZED 2)
	(b)			

Table 2: Overview of Research Methods

The following description provides a detailed overview of the main tools and methods utilized for the research.

Assistive Gym
Assistive Gym is a high level simulation framework that allows the creation and customization of robot simulation environments for assisting people and physically interacting with them (Erickson et al. 2020). At the core of Assistive Gym is the open source physics engine PyBullet (Coumans and Bai 2016). In addition to its ability to run multiple real-time simulations on both CPUs and GPUs, PyBullet is able to simulate cloth and soft bodies, and programmatically create human and robot models of varying shapes, sizes, weights, and joint limits. Moreover, OpenAI Gym integrates directly with Assistive Gym, enabling the use of control policy learning algorithms, including deep reinforcement learning.

Sawyer Robot
Sawyer is a collaborative robotic arm with a maximum reach of 1,260mm, a payload of 4kg, and seven degrees of freedom. Its first three joints have a range of 350 degrees, with Joints 4 and 5 restricted to 340 degrees and Joint 6 to 540 degrees. Table 3 gives a detailed overview of the robot's specifications. The Sawyer robot can be controlled using python with the help of ROS and Intera. The Robot Operating System (ROS) is an open-source, meta-operating system for robotics. As an operating system, it provides a number of functions,

including hardware abstraction, device control at a low level, implementation of commonly used functionality, message-passing between processes, and package management. Intera is a robust software control platform already incorporated into Sawyer, which allows easy control of the robot.

Basic Specifications	
Max Reach	1260 mm
Typical Tool Speed	1.5 m/s
Degrees of Freedom	7
Operating Temperature	5° C - 40° C 80 % relative humidity
Joint Ranges	J0 - J3 = 350° J4 - J5 = 340° J6 = 540°
Payload	4 kg
Power Requirements	100-240 V AC, 47-63 Hz, 4A Max
I/O Ports (Controller)	8 digital in/ 8 digital out
Communication	Modbus TCP, TCP/IP
IP Class	54
Collaborative Standards	ISO 10218-1:2011
I/O End of Arm	4 digital in / 2 digital out / 2 analog in / 24 V DC 2A, ClickSmart plate required

Table 3: Sawyer Specifications Sheet

ZED Camera

The ZED 2 stereo camera is an essential tool for this project. Indeed, it enables Sawyer to "see". A stereo camera is a camera equipped with two or more lenses, with one image sensor for each lens. This technique enables the camera to simulate human binocular vision and, as a result, to record three-dimensional images. In particular, the concept of triangulation is at the base of stereo camera. Typically, in Computer Vision (CV), triangulation refers to the process of identifying a point in 3D space based on its projection onto either two or more images. Similar to Sawyer, ZED offers its own Software Development Kit (SDK) and Python API, which allows easy and flexible use of the camera.

The following illustration provides an overview of how all of the mentioned tools are controlled and interact with one another.

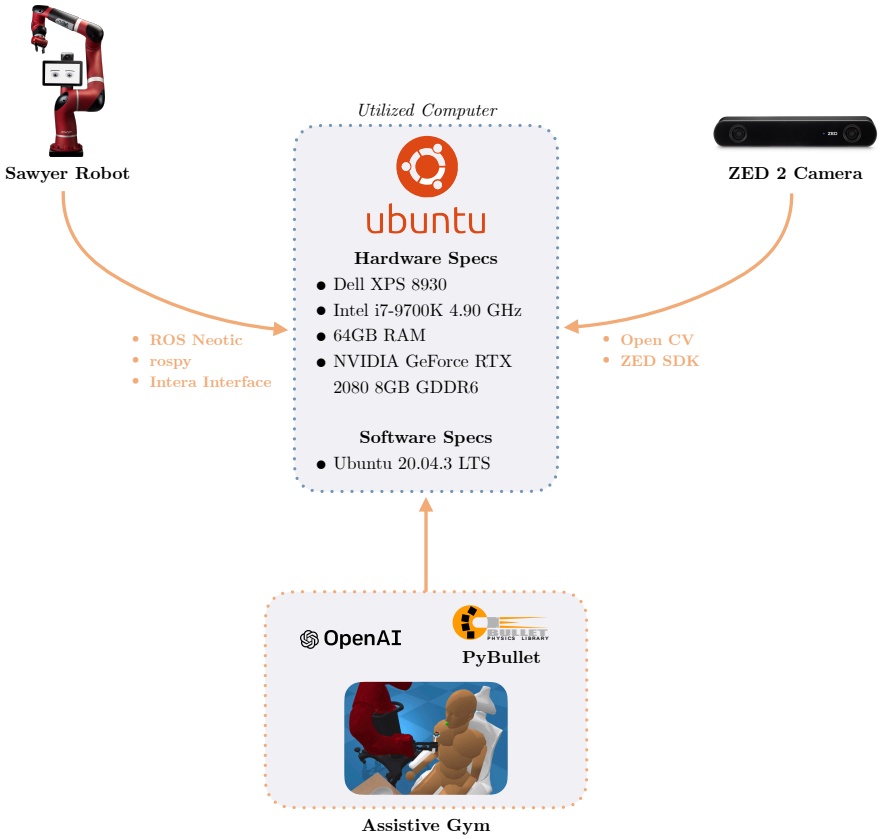


Figure 10: Schematic of Methods and Tools Utilized

4.1 Validation

Validation is performed to assess the outcome of the research. The internal validity of the research is achieved by ensuring that the predetermined goal is reached. Simply put, it is crucial to verify the functioning of the designed environment. For the purpose of this study, the simulated environment is transferred to a real scenario. Additionally, in order to ensure internal validity it is crucial to pay close attention to the various variables of the environment and what effects they have.

4.2 Risk Analysis

Risk analysis is essential in order to identify the various hazards that could arise during the research. Challenges could be faced in the process of programming and designing the environment as well as when transferring the learned policy onto the real robot. In other words, various assumptions would have to be made while programming, designing, and implementing. Assumptions are essential in order to simplify the model and achieve the research goal.

Part II

Design Stage

5 Simulation Setup

The purpose of this section is to describe the formulation and design of the "reaching" task using Assistive Gym in simulation (Erickson et al. 2020). A detailed description of the simulation environment is provided followed by the action space, observation features, and reward function definitions. In order to share the python code designed for the 'reaching' task, a GitHub repository was set up at the following URL: `https://github.com/gansaldo/reaching-task`. Further information regarding the coding process can be found in the repository.

5.1 Outline of Reinforcement Learning (RL)

Before diving into the details of the simulation a brief and concise explanation of a typical RL setup is introduced in the following diagram and further described.

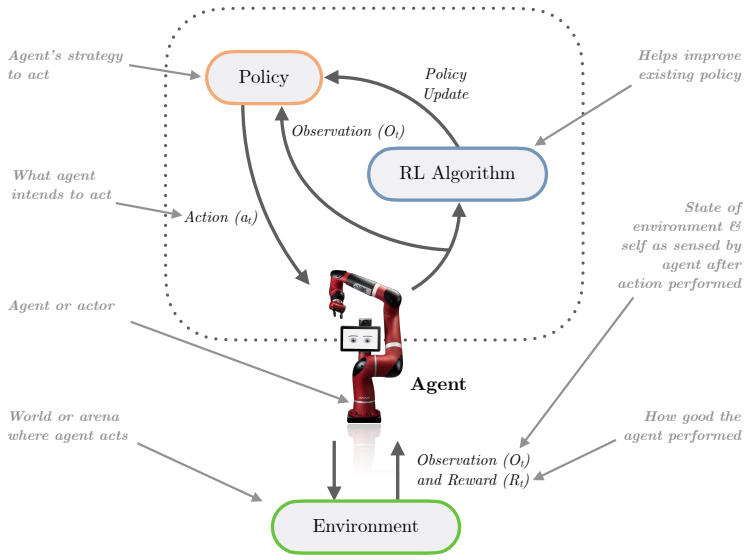


Figure 11: Typical RL Setup, Showing Essential Components and Terms

The goal of an agent (i.e. Sawyer) is to learn the most efficient possible way to perform a certain task in an Environment. This goal is accomplished by the agent taking actions (i.e. moving joints at different angles) in accordance with a strategy often referred to as policy. After the action is performed, the agent observes the state of the environment (Observation) and a goodness score (Reward) based on its last action. As a result of this feedback loop, the RL algorithm updates or improves the existing strategy (or policy) in order to increase future performance.

5.2 Simulation Environment

In order to train the model for the "reaching" task, an environment is designed in Assistive Gym. The HRC task consists of two sub-tasks, a *deterministic* and subsequently a *stochastic* one. The first sub-task is called *deterministic* since the robot's joint angles and movements are predetermined and hardcoded. The second sub-task is referred to as *stochastic* since the robot's joint positions are not predetermined but are the result of an RL policy learned in simulation and specifically using Assistive Gym. For the 'reaching' task, a blue and a red marker are placed on a shelf out of reach from a human operator. The position of the markers is constant for all simulations. Firstly, in the *deterministic* sub-task, the human points at the desired marker, and through computer vision (CV), the robot understands which of the markers was chosen. The chosen marker is then grabbed by the robot using predetermined joint positions. Secondly, the *stochastic* sub-task consists of the robot passing the marker to the human's right hand. This sub-task is solely based on the environment designed in Assistive Gym which allows the robot to learn to reach a human's hand.

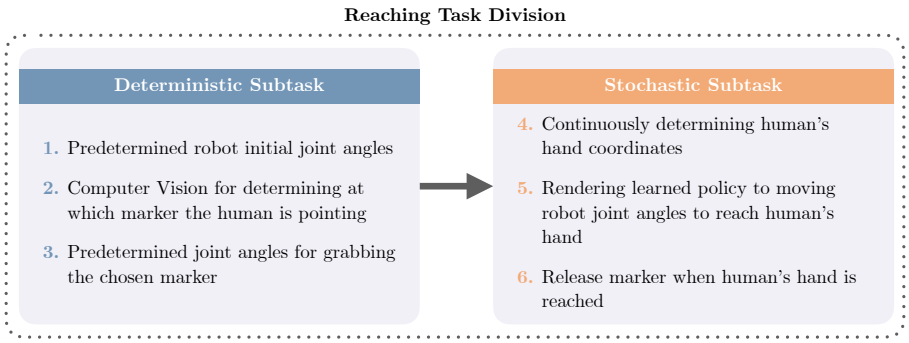
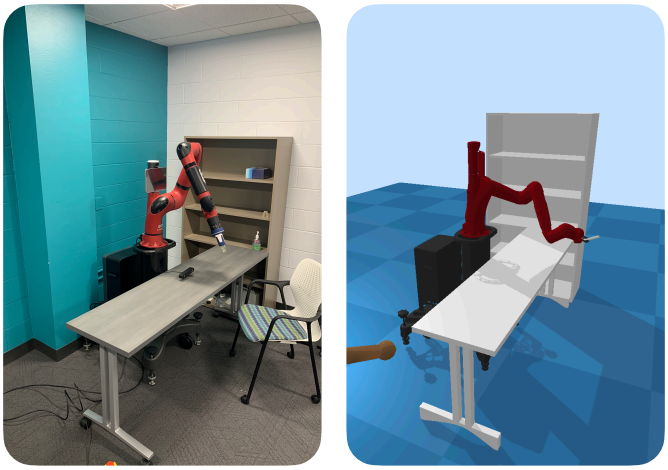


Figure 12: Task Diagram Division

Figure 13 below shows a side by side picture of the real environment and the designed environment.

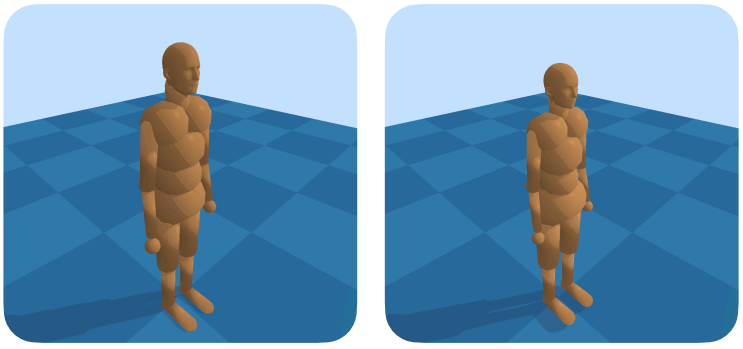


(a) (b)

Figure 13: (a) Real Environment and (b) Designed Environment

5.2.1 Environment Limitations & Assumptions

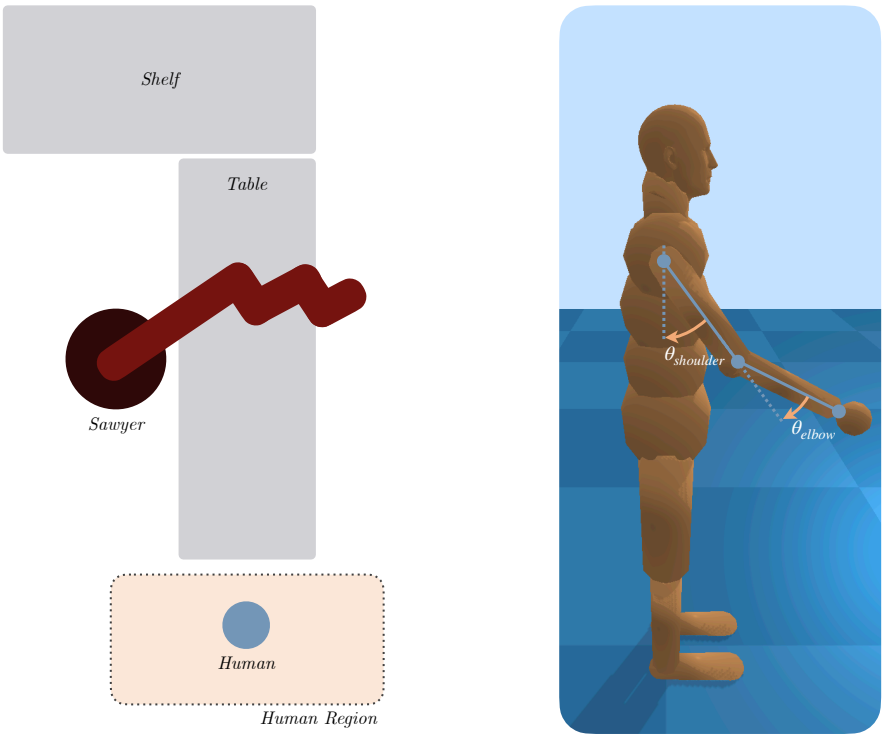
The human model utilized for the designed task is a configurable capsulized male or female model as shown in Figure 14 below. The body sizes, weights, and joint limits of the human model match published 50th percentile values as provided by Assistive Gym.



(a) (b)

Figure 14: Simulation Human Model: (a) Male Model, (b) Female Model

The human model is initially randomly positioned standing up within a predetermined region that can be reached by the robotic arm (refer to Figure 15 (a)). The right elbow angle θ_{elbow} and right shoulder angle $\theta_{shoulder}$ are uniformly sampled between -50 and -10 degrees and between -70 and -10 degrees respectively (refer to Figure 15 (b)). Additionally, for the designed environment, the simulator randomly selects between a male or female human model.



(a) (b)

Figure 15: (a) Schematic Top View of Simulation (b) Human's Arm Angles

When simulating physical human-robot interactions, it is essential to model realistic human joint limits. Nevertheless, the problem with modeling joint limits is they are pose-dependent, meaning the range of motion of one joint is determined by other joints' configurations. In a paper published by Akhter and Black, a method for modeling realistic limits of human joints using discrete operations is presented (Akhter and M. J. Black 2015). In recent research, a fully connected neural network model, trained on human motion capture data, was used to reduce the computational requirements of determining if a human pose is valid (Jiang and C. K. Liu 2018). Favorably, Assistive Gym already incorporates this neural network model to represent realistic joint limits in the human arm.

5.3 Actions and Observations

Actions

An action space A is defined for the 'reaching' task where the robot moves from an initial position to a release point for the object to be passed. Table 4 below gives an overview of the action space for the studied environment, in particular the Sawyer robot. The ability of Sawyer to perform actions at any given location is affected by the physical limits of the arm. Indeed, each joint of the robot is able to rotate till a certain limit. Actions for the robot are defined as $a = (\Delta J_0, \Delta J_1, \Delta J_2, \Delta J_3, \Delta J_4, \Delta J_5, \Delta J_6) \in A$, where $J_0, J_1, J_2, J_3, J_4, J_5, J_6$ are the robot's joint angles and $\Delta J_0, \Delta J_1, \Delta J_2, \Delta J_3, \Delta J_4, \Delta J_5, \Delta J_6$ are delta joint angles that are added to the robot's current joint angles in order to move the robot's arm closer to the desired target. Simply put, given a release point, the robot moves its joints to reach the release point after which the object to be passed is released.

Num	**Action**	**Range**	
		Min	Max
0	Delta Joint 0 Angle	-3.05 *rad*	3.05 *rad*
1	Delta Joint 1 Angle	-3.05 *rad*	3.05 *rad*
2	Delta Joint 2 Angle	-3.05 *rad*	3.05 *rad*
3	Delta Joint 3 Angle	-3.05 *rad*	3.05 *rad*
4	Delta Joint 4 Angle	-2.97 *rad*	2.97 *rad*
5	Delta Joint 5 Angle	-2.97 *rad*	2.97 *rad*
6	Delta Joint 6 Angle	-4.71 *rad*	4.71 *rad*

Table 4: Action Space

Observations

An observation is captured by the robot at the start of each trial. This includes the end effector position of the robot $E_p = (E_{p,x}, E_{p,y}, E_{p,z})$, the distance to the target $d_t = (d_{t,x}, d_{t,y}, d_{t,z})$, robot joint angles $J_o = (J_{o,0}, J_{o,1}, J_{o,2}, J_{o,3}, J_{o,4}, J_{o,5}, J_{o,6})$, and the hand position $H_p = (H_{p,x}, H_{p,y}, H_{p,z})$. Table 5 below gives an detailed overview of the observation space. In general, an observation for the robot is defined as $o = (E_p, d_t, J_o, H_p)$. This observation o is then provided to a policy $\pi(o)$, which outputs joint angles for reaching the human's hand.

Num	Observation		Range	
			Min	Max
0	End Effector Position	x	/	/
		y	/	/
		z	/	/
1	Distance to Target	x	/	/
		y	/	/
		z	/	/
2	Sawyer Joint Angles	*J0*	-3.05 rad	3.05 rad
		J1	-3.05 rad	3.05 rad
		J2	-3.05 rad	3.05 rad
		J3	-3.05 rad	3.05 rad
		J4	-2.97 rad	2.97 rad
		J5	-2.97 rad	2.97 rad
		J6	-4.71 rad	4.71 rad
3	Hand Position	x	/	/
		y	/	/
		z	/	/

Table 5: Observation Space

5.4 Reward Definition

To calculate the reward value used to train policies, a reward function $R(S)$ is defined for the 'reaching' task given the global state of the simulation environment S. The reward function is defined as follows:

$$R(S) = d_{weight} \times d_{target} + a_{weight} \times a_{reward} + p_{score}$$

where:

- d_{target}: is the the Euclidean distance between the gripper and hand's position.
- d_{weight}: is the weight related to the Euclidean distance. The higher the weight the higher the penalty for being far to the target.
- a_{reward}: is the norm of the actions taken for moving the joint angles of the robot. This makes sure that the robot takes the lower amount of actions to reach the human's hand.
- a_{weight}: is the weight related to the actions taken by the robot. The higher the weight the higher the penalty for taking actions and moving joints.
- p_{score}: is a penalty related to the gripper velocity and the total force applied to the human. The higher the force and velocity the higher the penalty. This makes sure that little force is applied to the human.

5.5 Policy Learning Method

In order to learn a policy that permits a robot to reach a human's hand, a reinforcement learning formulation is employed. In particular, proximal policy optimization (PPO) is

utilized (Schulman et al. 2020). This is a deep reinforcement learning technique using a fully connected neural network with two hidden layers of 50 nodes and tanh activations in order to learn policies. Nevertheless the choice of the reinforcement learning algorithm plays little role in this research. Theoretically, any algorithm that provides satisfactory results and can function in a continuous environment could be chosen.

The policy is trained with random variation in a human pose (as discussed earlier) in order to facilitate better transfer of the learned policies to the real world. The policy is trained using a NVIDIA GeForce RTX 2080 8GB GDDR6 GPU machine with an Intel i7-9700K CPU. To ensure the same effectiveness, the training algorithm's hyperparameters were identical to those chosen for the original Assistive Gym Paper (Erickson et al. 2020). To generate the policy, the algorithm was allowed to run for ten million timesteps. In practice, this number proved to be large enough for the robot to develop an effective policy while also being small enough to train within a reasonable period of time. Every episode was 200 timesteps long, which means the robot was able to observe 200 observations and take 200 actions in order to solve the task. At the end of each episode, the robot, human, and environment were reset to their original positions. Over the course of training episodes, the robot learns which actions are appropriate for completing the task and which aren't until an effective policy is learned. Figure 16 below shows the learning curve over the ten million timesteps used for generating the policy.

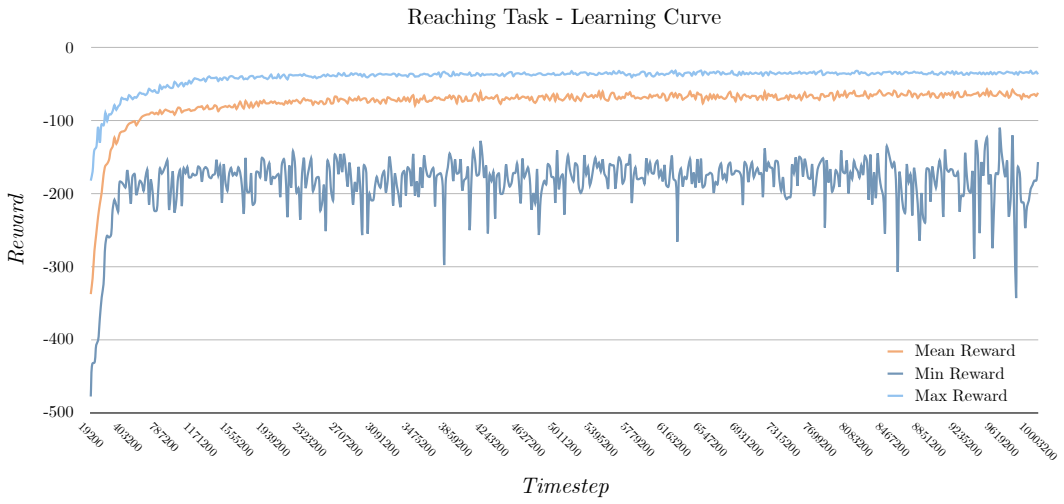


Figure 16: Learning Curve for the Reaching Task using PPO

It can be seen from the graph above that an effective policy is learned within the first 1500000 timesteps. Indeed, the learning curve follows a logarithmic growth pattern, in that improvements come quickly at the beginning but the gains decrease over time. It is interesting to note that while maximum reward values show to be consistent, minimum reward values exhibit great fluctuations. This may be due to the robot starting from initial configurations that are unfeasible for reaching the human's hand, or the human starting from a location and arm arrangement that the robot is physically unable to reach. While such scenarios may affect the learning process, this is not true for the current task as the overall learning curve still reaches a plateau.

Part III

Implementation/Validation Stage

6 Implementing Trained Model in the Real World

Several factors must be considered when implementing the simulation in the real world. A discussion of the integration of Computer Vision (CV), methods of synchronizing simulation with real-world parameters will be provided in the following sections.

6.1 Integration of Computer Vision

In the area of artificial intelligence (AI), computer vision refers to a technique in which computers and systems can derive meaningful information from digital images, video, and other visual inputs and take actions or make recommendations based upon that information. When implementing the 'reaching task' in the real-world, CV is required for both the *Deterministic* and *Stochastic* subtasks. For the purpose of this project, a ZED 2 camera was utilized (STEREOLABS 2021). The ZED 2 is one of the most powerful stereo camera currently on the market. It offers unmatched field-of-view, imaging quality, neural sensing, robustness, 3D object recognition, and cloud management.

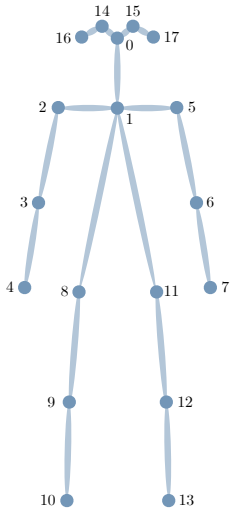
Figure 17: ZED 2 Stereo Camera

Essentially, the ZED is a camera that mimics the way in which human vision works. Using triangulation, the ZED provides a three-dimensional picture of what it observes. Moreover, the camera's API provides great flexibility for integrating the camera with robotic applications. In particular, for the 'reaching' task the 'body tracking' module offered by the ZED API is utilized to track and identify human's movements and gestures.

Figure 18: Output Example of Body Tracking Module

The objective of this module is to detect and track a person's bones. Bones are represented by their two ends, which are known as keypoints. The body tracking module utilizes neural network for keypoints detection and is able to provide 3D information about each detected keypoint.

Figure 19 below shows the body format used for the 'reaching' task. Each keypoint is indexed by an integer ranging from 0 to 17.



Keypoint Index	Keypoint Name
0	Nose
1	Neck
2	Right Shoulder
3	Right Elbow
4	Right Wrist
5	Left Shoulder
6	Left Elbow
7	Left Wrist
8	Right Hip
9	Right Knee
10	Right Ankle
11	Left Hip
12	Left Knee
13	Left Ankle
14	Right Eye
15	Left Eye
16	Right Ear
17	Left Ear

Figure 19: Body Format 18 Keypoints

Further explanation of how CV was utilized with the ZED camera in both the *Deterministic* and *Stochastic* subtasks is provided.

Deterministic Subtask

As mentioned in Part II, during the *Deterministic* Subtask the human can point at the desired marker. The two markers are positioned on the shelf and one is more on the right while the other more on the left. It is therefore crucial to understand whether the human is pointing at the right or left marker. It is possible to understand this by looking at the human body position. Figure 20 shows two standard scenario of the pose assumed by the human when pointing left and right respectively.

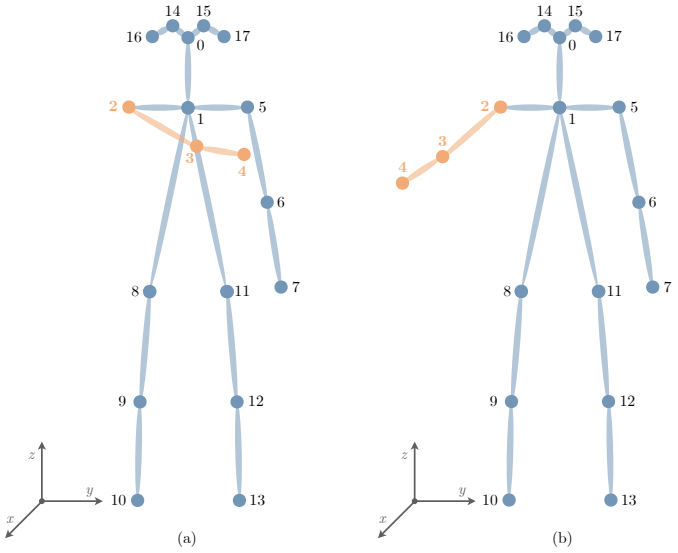


Figure 20: Pointing Human Poses, (a) Pointing Left, (b) Pointing Right

The ZED camera is able to detect 3D location of each joint, therefore it is possible to identify exactly which pose the human is assuming and thus whether it is pointing right or left. This can be done by using a simple decision tree as follows (note that the numbers refer to the keypoint index, i.e. 2_x is the x coordinate of keypoint 2, the right shoulder):

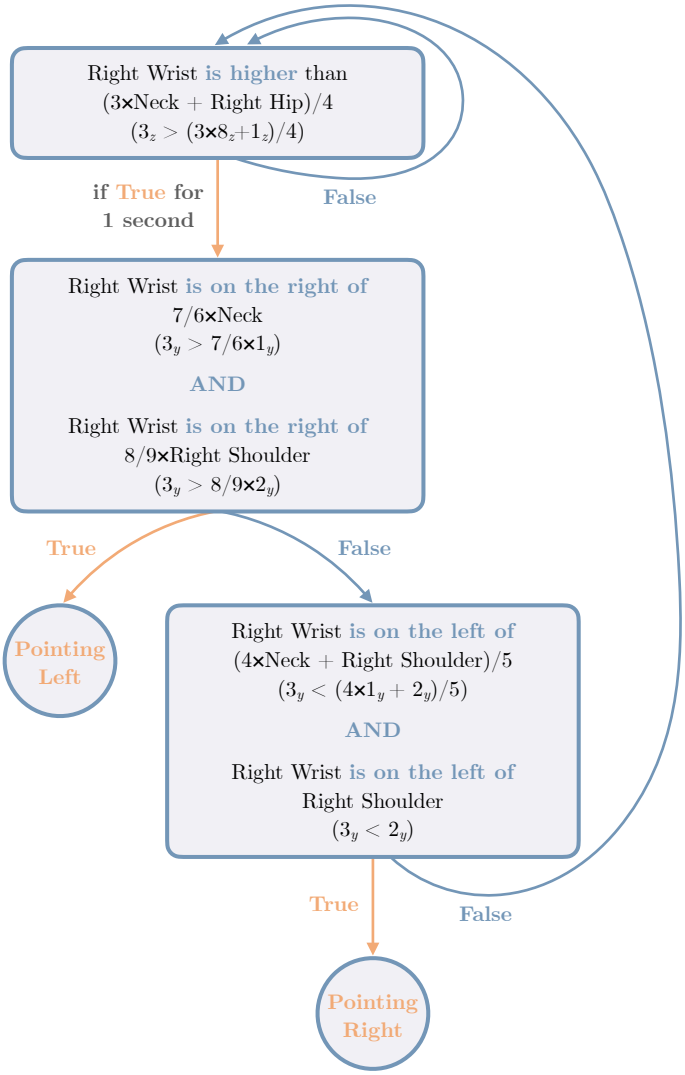


Figure 21: Decision Tree for Pointing

Using the decision tree, it is possible to determine whether the human is pointing left or right. A human can, however, move randomly while being observed by a camera, rather than remaining still. In addition, software errors could lead to incorrect detection of the human body. Thus, it is necessary for the human to maintain its position for roughly one second in order to verify that the pointing action is occurring.

Stochastic Subtask

The *Stochastic* sub-task, referred to in part II, consists of rendering the learned policy and passing the marker grabbed by the robot into the human's hand. For completing this sub-task it is imperative to continuously locate the human's right hand and feed this information into the rendered policy. As shown in Figure 22 below, keypoint 4, the right wrist, is the joint that needs to be detected by the ZED camera.

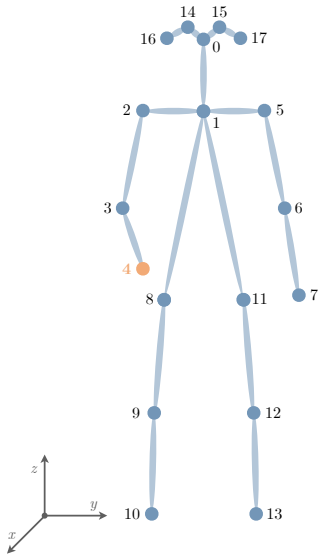


Figure 22: Possible Human Pose with Highlighted Wrist Joint

For the purpose of providing the policy with a smooth hand movement and reducing potential software or detection errors, the past 20 to 30 observations of the human's hand are averaged.

6.2 Synchronizing Simulation and Real World Parameters

There is a set of essential parameters that must be made consistent between the simulation and the physical environment in order to successfully transfer anything between the two environments. As a first step, it was necessary to match the reference system used by the robot in the real world to that used in simulation. Considering the physical robot uses a local coordinate system whose origin is its center, it is imperative to synchronize this system with the virtual robot's coordinate system. Aside from that, it was also necessary to match the ZED camera coordinate system with the simulation. This is essential, as accurate observations of the location of the human hand are crucial. For the sake of simplicity, it was decided to match all coordinate system to the one of the robot in the real world. Figure 23 below gives a schematic representation of the different coordinate systems present in the system.

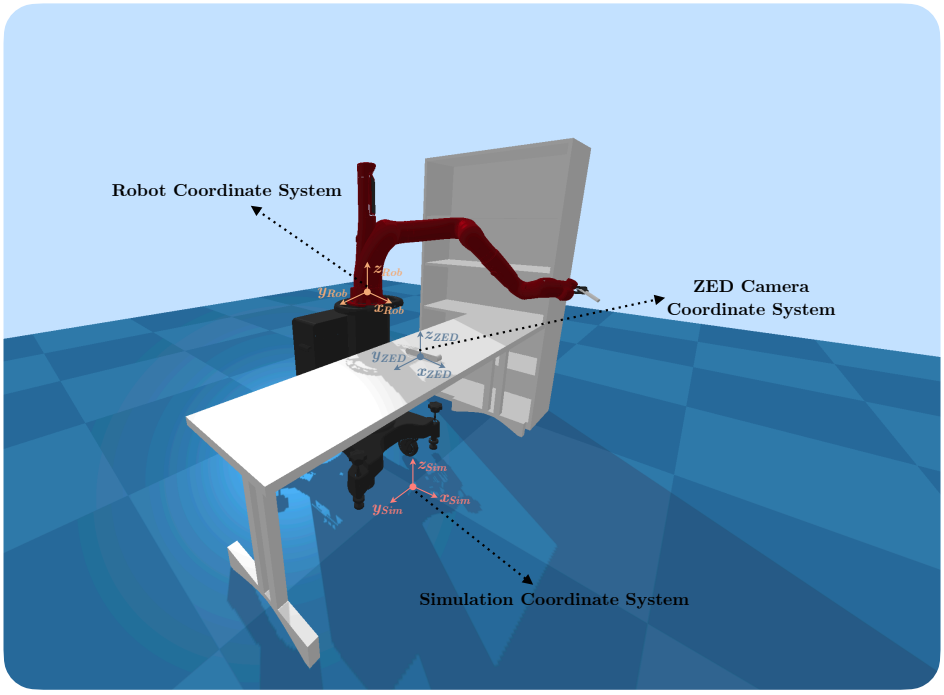


Figure 23: Various Coordinate Systems

Furthermore, in simulation, the observations with regards to joint angles returned by the robot are normalized between -180° and +180° ($-\pi$ and $+\pi$) by the model before training. Therefore, before incorporating the physical robot's observations into the model, the same normalization formula must be applied to ensure consistency. As a result of this normalization, the following formula is used:

$$J_{n,sim} = (J_{n,real} + \pi)\ (mod\ 2\pi) - \pi \quad \forall\ n \in N$$

where $J_{n,sim}$ is the sum of $-\pi$ and the remainder of $\frac{J_{n,real}+\pi}{2\pi}$. Specifically, $J_{n,real}$ is the real observation in radians of joint $n \in N = \{0, 1, 2, 3, 4, 5, 6\}$ and $J_{n,sim}$ is the normalized observation used by the model.

6.3 Sim-To-Real Approaches

Upon executing the learned policy on the real robot significant inconsistencies between the simulation and real-world environment were present. Particularly, these concerned the implementation of the action returned by the trained model. To illustrate, a robot in simulation and a robot in real life would behave differently when given the same action. For instance, consider the case where, in simulation, the robot generated a sequence of actions to move its end-effector 20 degrees clockwise and three centimeters down. When this same sequence of actions was applied to the physical robot, it was noted that the result was something slightly different and not entirely comparable to the movement of the robot in simulation. In order to resolve this issue, three strategies were employed and tested, including:

- **Calibrating Coordinate Systems**: it was found that an accurate calibration of the coordinate system is crucial for precise robot movement. A slight misalignment could often lead to great misestimation of human's joint locations.
- **Action Adjustments**: it was discovered that tweaking delta joint angles in the action space was advantageous in order to achieve smooth, real-world robot movement.
- **Domain Randomization**: as mentioned in previous sections, domain randomization is a well-proven method for sim-to-real scenarios. This was accomplished by adding random noise values equal to a magnitude of approximately 0.02 to the observation space.

Results & Discussion

The three strategies selected for sim-to-real transfer are discussed in more detail and their results are described.

Calibrating Coordinate Systems

Early in the experimentation process, it became evident that standardizing the coordinate systems between the simulation and real-world environment would be an essential component to ensure consistency and efficiency. Unsynchronized coordinate frames can cause problem in robot joint angles which become more pronounced and hazardous in environments involving humans. A robot, for example, may be learning how to pick up an object, in which case a slight difference in reference frames will only result in the robot picking up the object in an unsatisfactory manner. However, if a human is directly interacting with a robot, unwanted robot movements could harm the human rather than assisting it. Therefore, it is of paramount importance to ensure that coordinates frames are properly synchronized. As a result, adequate time was spent on ensuring that appropriate synchronization was achieved. Specifically, Euclidean Transformations were used to ensure that the calibration was accurate. Figure 24 below illustrates the schematic view of the calibration problem to be solved in order to obtain an appropriate calibration.

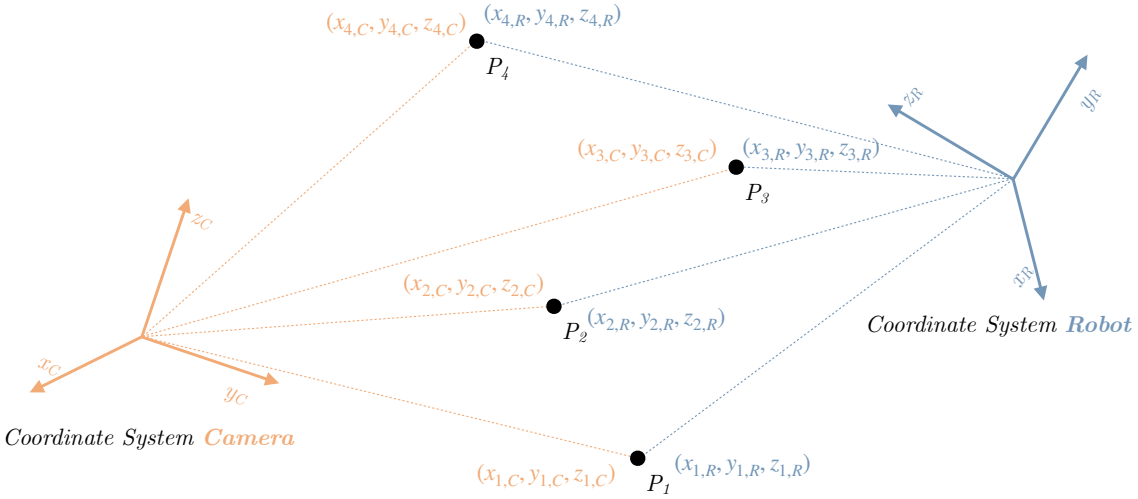


Figure 24: Schematic of Method for Calibrating Coordinate Systems

To convert any point n from the camera coordinate system to the robot reference frame, a rotation and translation transformation are required:

$$\begin{bmatrix} x_{n,R} \\ y_{n,R} \\ z_{n,R} \end{bmatrix} = R \begin{bmatrix} x_{n,C} \\ y_{n,C} \\ z_{n,C} \end{bmatrix} + T \quad \text{where} \quad R = \begin{bmatrix} R_1 & R_2 & R_3 \\ R_4 & R_5 & R_6 \\ R_7 & R_8 & R_9 \end{bmatrix}, T = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix}$$

It is therefore crucial to calculate both the rotation matrix R and the translation matrix

T. For this purpose, a system of equations can be set up in which 4 known points are sampled from both the camera and the robot (refer to Figure 24):

$$\begin{cases} \begin{bmatrix} x_{1,R} \\ y_{1,R} \\ z_{1,R} \end{bmatrix} = R \begin{bmatrix} x_{1,C} \\ y_{1,C} \\ z_{1,C} \end{bmatrix} + T \\ \begin{bmatrix} x_{2,R} \\ y_{2,R} \\ z_{2,R} \end{bmatrix} = R \begin{bmatrix} x_{2,C} \\ y_{2,C} \\ z_{2,C} \end{bmatrix} + T \\ \begin{bmatrix} x_{3,R} \\ y_{3,R} \\ z_{3,R} \end{bmatrix} = R \begin{bmatrix} x_{3,C} \\ y_{3,C} \\ z_{3,C} \end{bmatrix} + T \\ \begin{bmatrix} x_{4,R} \\ y_{4,R} \\ z_{4,R} \end{bmatrix} = R \begin{bmatrix} x_{4,C} \\ y_{4,C} \\ z_{4,C} \end{bmatrix} + T \end{cases}$$

Therefore, using the above equations and obtaining the rotation matrix in addition to the translation matrix, any new point detected by the camera may be easily mapped into the robot's coordinate system. A python code implementing this approach has been developed in which 4 end-effector positions are sampled from both the camera and robot reference frames. As a result, the system of equations is solved and the rotation matrix R and translation matrix T are derived. An example of how the calibration method is conducted in practice can be found at the following URL: `https://github.com/gansaldo/reaching-task#coordinate_system_calibrationpy`

Action Adjustments

Based on the received observations, the learned policy outputs a sequence of actions that the robot takes to get closer to the target. As mentioned in Section 5.3 these actions are delta joint angles. This entails that the changes in joint angles in the action can be added to the robot's current joint angles in order to move the robot's arm. This method enables the adjustment and fine-tuning of the delta joint angles to ensure that during a given time step, both the robot in simulation and the robot in physical form can reach the same position. Furthermore, this approach allowed fine-tuning to prevent the imposition of changes that were too small or too large. It is possible to cause jerky, unpleasant motion if the changes in joint angles are too small, as the robot will be able to reach its target location too rapidly, and then come to a complete stop. Indeed, jagged movements are not ideal in a setting where humans are involved as it could cause human discomfort. To mitigate this problem, an action is not taken for any delta joint angle smaller than a given threshold. Alternatively, if the delta of a particular joint is greater than a given threshold, then the joint will move as indicated by the delta.

$$\Delta J_{n,\text{ adjusted}} = \begin{cases} 0 & if \;\; \Delta J_n \leq \text{threshold} \\ \Delta J_n & if \;\; \Delta J_n > \text{threshold} \end{cases} \qquad \forall \; n \in N$$

After iteratively experimenting, a threshold of 0.2 *rad* for the delta joints was found to be a reasonable value to prevent the robot's movements from being jerky. It is important to note that after this new method was implemented, a new policy was learned over ten million timesteps. Figure 25 below shows the learning curve obtained during training.

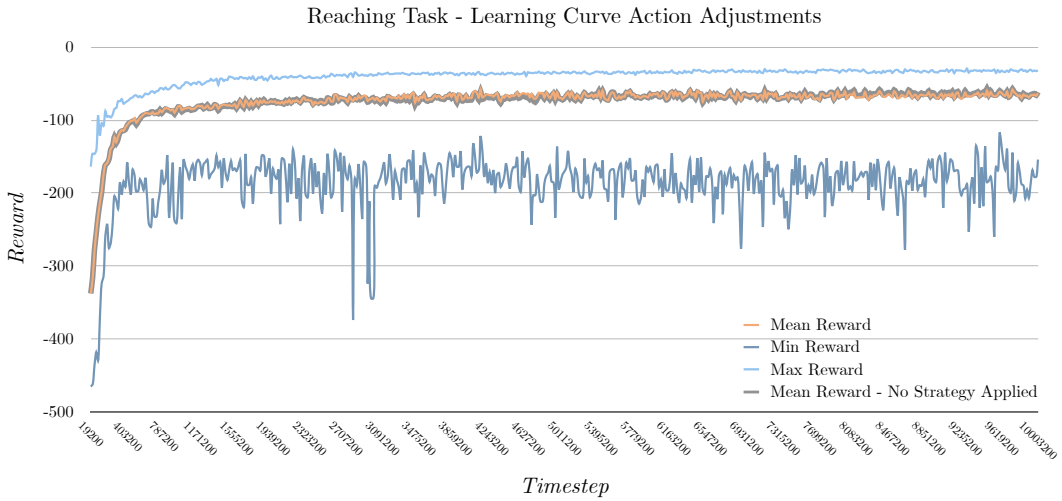


Figure 25: Learning Curve for the Reaching Task using PPO with Action Adjustments

It can be noted that the strategy of avoiding small delta joint angles does not have an adverse effect on learning. As can be seen from the graph, the mean reward when a strategy is applied follows an almost identical pattern as the mean reward when no strategy is applied. In spite of the fact that such an adjustment does not have a direct impact on the learning process and simulation, it has a dramatic impact on the transfer of simulation to the real robot since jerky movements are removed and smooth robot movement is achieved.

Domain Randomization

According to earlier sections, domain randomization has been shown to be effective in sim-to-real transfer. As a result, it was decided to include in the observations of the simulated environment uniformly distributed noise values between ± 0.02. In spite of the fact that the randomization did not appear to affect the performance of the robot either positively or negatively, it was still decided to maintain it. As a result of implementing this procedure, a new policy was learned over ten million time steps. Figure 26 illustrates the learning curve obtained during training.

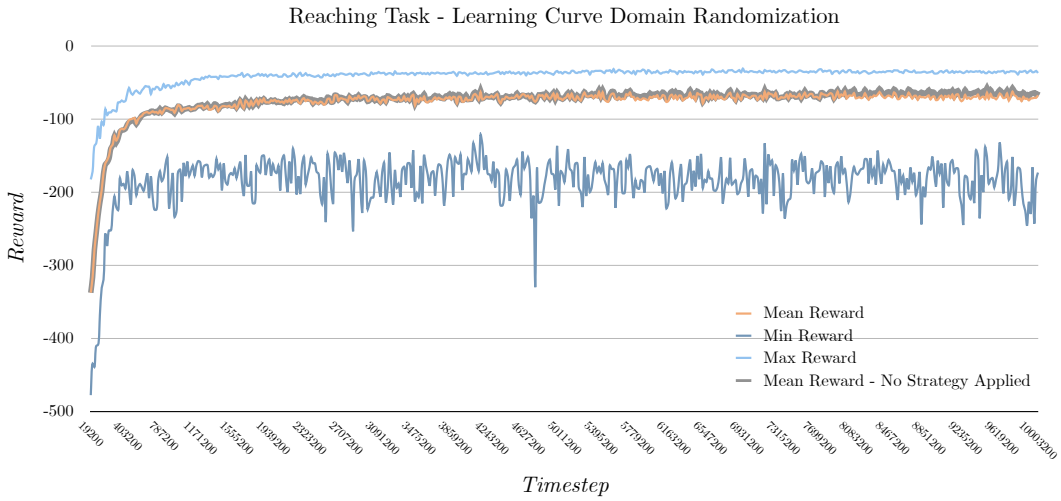


Figure 26: Learning Curve for the Reaching Task using PPO with Domain Randomization

As for the Action Adjustments, Domain Randomization does not have a negative effect on learning. Indeed, the graph indicates that the mean reward when a strategy is applied follows a pattern that is almost identical to that of the mean reward when no strategy is applied. The purpose of making such an adjustment is not to directly affect the simulation environment, but rather to make the policy resilient to minor random and unmodeled changes that could occur when transferring the policy to a real robot.

It came to light after implementing the above combination of changes to the simulated learning environment that the trained policies were robust and consistent to the extent that the physical robot was now able to consistently execute any simplistic policy learned in simulation. For greater clarity, the physical robot could accurately reproduce the motions generated by the designed environment with only the information required for the task being the joint angles of the robot and the location of the human's right hand. The physical robot was able to grab a chosen marker and pass it to the real person in virtually the same manner as the simulation robot. Based on the criteria outlined in Part II of this report, the goal of designing and implementing an HRC task was accomplished. Having successfully synchronized its physical and simulated environments, a physical robot could now follow a policy that was trained in simulation to achieve the same goal as a simulated robot.

Conclusion & Future Research

This research aimed at developing and implementing a multi-agent industrial simulation environment where a cobot and a human collaborate to achieve a common goal. Simply put, the proposed collaboration task consists of passing an object from a shelf to an operator's hand. Applications for such a system could be many. As an example, during manual assembly, an operator may need various tools that are out of reach, and as a result, a robot could grab and pass them. In addition, in case of logistical processes, if a package is located on a shelf out of reach from an operator, the operator could easily point to the desired package and the robot could pass it to the operator's hand. However, an interesting aspect of this research lies in how reinforcement learning is applied to simulation and how such an approach can then be applied to a real-world situation. Indeed, as cobots and RL methods have grown in popularity, there has been an increased demand for simulation environments that allow HRC scenarios to be simulated. This project aimed at addressing this issue and making an effort at merging the fields of HRC, RL, Simulation, and Sim-To-Real Transfer.

The design stage of the project included the creation of the environment using the simulation framework Assistive Gym. Especially with the physics engine PyBullet included in Assistive Gym, realistic movements could be achieved. It is important to note that several assumptions and limitations regarding the human body's movements and positions were made. The human is specifically restricted to staying within a predetermined area and to moving its arm in a predetermined random manner. This limitation and assumption had no apparent impact on the effectiveness of the learned policy in the real world.

The implementation stage was focused on effectively transferring the learned policy to the real robot and use CV to allow for human tracking. On a general note, this stage served to identify a few essential steps needed to transfer the designed 'reaching task' in Assitive Gym from the simulation environment to a physical Sawyer robot with a human involved. Namely, these steps are the standardization of world-space, integration of CV, action generation, and action deployment (movement) from the simulated robot to the physical robot.

In conclusion, this work allows for a variety of further extensions, such as exploring alternative HRC scenarios as well as characteristics of simulated environments which may improve the transfer from simulation to reality. In particular, it would be interesting to design and implement a scenario in which multiple humans are involved. Moreover, aspects related to human discomfort as well as how the interaction with cobots might reduce human stress and workload could also be studied. With regards to sim-to-real aspects, a study of different physics engines could be conducted to determine their suitability for sim-to-real transfer experiments. In spite of PyBullet's usefulness, it would be fascinating to determine whether other physics engines make transferring tasks easier or more difficult. A second step, would be to examine which alternative reinforcement learning algorithms lend themselves most appropriately to simulation-to-reality applications and HRC environments.

Bibliography

ABI Research (2021). "Cobots for Flexible Automation". In: *ABI Research: The Tech Intelligence Experts*. URL: https://www.abiresearch.com/market-research/product/1027368-cobots-for-flexible-automation/.

Akhter, Ijaz and Michael J Black (2015). "Pose-conditioned joint angle limits for 3D human pose reconstruction". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1446–1455.

Arndt, Karol et al. (2020). "Meta reinforcement learning for sim-to-real domain adaptation". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2725–2731.

Bauer, Andrea, Dirk Wollherr, and Martin Buss (2008). "Human–robot collaboration: a survey". In: *International Journal of Humanoid Robotics* 5.01, pp. 47–66.

Breyer, Michel et al. (2018). "Flexible robotic grasping with sim-to-real transfer based reinforcement learning". In: *arXiv preprint arXiv:1803.04996*.

Brockman, Greg et al. (2016). "Openai gym". In: *arXiv preprint arXiv:1606.01540*.

Coelho, Fábio, Susana Relvas, and Ana Paula FD Barbosa-Póvoa (2018). "Simulation Of An Order Picking System In A Manufacturing Supermarket Using Collaborative Robots." In: *ECMS*, pp. 83–88.

Coumans, Erwin and Yunfei Bai (2016). "Pybullet, a python module for physics simulation for games, robotics and machine learning". In:

Ding et al. (2020). "Sim-to-real transfer for optical tactile sensing". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1639–1645.

Erickson, Zackory et al. (2020). "Assistive gym: A physics simulation framework for assistive robotics". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 10169–10176.

Fager, Patrik, Fabio Sgarbossa, and Martina Calzavara (2021). "Cost modelling of onboard cobot-supported item sorting in a picking system". In: *International Journal of Production Research* 59.11, pp. 3269–3284.

Fan, Linxi et al. (2018). "Surreal: Open-source reinforcement learning framework and robot manipulation benchmark". In: *Conference on Robot Learning*. PMLR, pp. 767–782.

Grosse, Eric H, Christoph H Glock, and W Patrick Neumann (2017). "Human factors in order picking: a content analysis of the literature". In: *International Journal of Production Research* 55.5, pp. 1260–1276.

Gualtieri, Luca, Erwin Rauch, and Renato Vidoni (2021). "Emerging research fields in safety and ergonomics in industrial collaborative robotics: A systematic literature review". In: *Robotics and Computer-Integrated Manufacturing* 67, p. 101998.

Hermans, C and WJ Schoeman (2015). "Practice-oriented research in service of designing interventions". In: *Acta Theologica*, pp. 26–44.

Hu, Michael et al. (2018). "The state of human factory analytics". In: *AT Kearney and D. Report*.

Jaghbeer, Yasmeen, Robin Hanson, and Mats Ingemar Johansson (2020). "Automated order picking systems and the links between design and performance: a systematic

literature review". In: *International Journal of Production Research* 58.15, pp. 4489–4505.

Jeong, Rae et al. (2019). "Modelling generalized forces with reinforcement learning for sim-to-real transfer". In: *arXiv preprint arXiv:1910.09471*.

Jiang, Yifeng and C Karen Liu (2018). "Data-driven approach to simulating realistic human joint constraints". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1098–1103.

Kaspar et al. (2014). "Reinforcement Learning with Cartesian Commands and Sim to Real Transfer for Peg in Hole Tasks". In:

Kolve, Eric et al. (2017). "Ai2-thor: An interactive 3d environment for visual ai". In: *arXiv preprint arXiv:1712.05474*.

Matas et al. (2018). "Sim-to-real reinforcement learning for deformable object manipulation". In: *Conference on Robot Learning*. PMLR, pp. 734–743.

Pasparakis, Alexandros, Jelle de Vries, and MBM de Koster (2021). "In Control or under Control? Human-Robot Collaboration in Warehouse Order Picking". In: *Human-Robot Collaboration in Warehouse Order Picking (March 31, 2021)*.

Peng, Xue Bin et al. (2018). "Sim-to-real transfer of robotic control with dynamics randomization". In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, pp. 3803–3810.

Puthuveetil et al. (2021). "Bodies Uncovered: Learning to Manipulate Real Blankets Around People via Physics Simulations". In: *arXiv preprint arXiv:2109.04930*.

Savva, Manolis et al. (2019). "Habitat: A platform for embodied ai research". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9339–9347.

Schaffer, Holden C (2021). "Investigating Sim-to-Real Transfer and Multi-Agent Learning in Assistive Gym". In:

Schmitz, Norbert, Jochen Hirth, and Karsten Berns (2010). "A simulation framework for human-robot interaction". In: *2010 Third International Conference on Advances in Computer-Human Interactions*. IEEE, pp. 79–84.

Schulman, John et al. (Sept. 2020). *Proximal Policy Optimization*. URL: `https://openai.com/blog/openai-baselines-ppo/`.

STEREOLABS (2021). "Zed 2 - AI stereo camera". In: *STEREOLABS*. URL: `https://www.stereolabs.com/zed-2/`.

Terry, Justin K et al. (2020). "Pettingzoo: Gym for multi-agent reinforcement learning". In: *arXiv preprint arXiv:2009.14471*.

Tobin, Joshua P (2019). *Real-World Robotic Perception and Control Using Synthetic Data*. University of California, Berkeley.

Vacaro, Juliano et al. (2019). "Sim-to-real in reinforcement learning for everyone". In: *2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)*. IEEE, pp. 305–310.

Vysocky, ALES and Petr Novak (2016). "Human-Robot collaboration in industry". In: *MM Science Journal* 9.2, pp. 903–906.

Wang, Qing et al. (2019). "Arena: a toolkit for multi-agent reinforcement learning". In: *arXiv preprint arXiv:1907.09467*.

Yamazaki, Kimitoshi et al. (2014). "Bottom dressing by a life-sized humanoid robot provided failure detection and recovery functions". In: *2014 IEEE/SICE international symposium on system integration*. IEEE, pp. 564–570.

Zamora, Iker et al. (2016). "Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo". In: *arXiv preprint arXiv:1608.05742*.

Zhao, Wenshuai, Jorge Peña Queralta, and Tomi Westerlund (2020). "Sim-to-real transfer in deep reinforcement learning for robotics: a survey". In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, pp. 737–744.