Master Thesis

Advancing Human-Robot Interaction via Digital Twins

A Benchmark Designed for Toyota Human Support Robot

Yi Wu

Advisor: Oya, Celiktutan - King's College London oya.celiktutan@kcl.ac.uk

August 27, 2023

Acknowledgments

Completing this master's thesis has been a transformative journey, and I am deeply grateful for the guidance and support I have received from numerous individuals who have played a pivotal role in shaping my academic and personal growth. Their wisdom, encouragement, and mentorship have been instrumental in reaching this significant milestone.

I extend my heartfelt appreciation to all the teachers who have imparted their knowledge and expertise during my time as a master's student. Your dedication to education and willingness to share your insights have greatly enriched my learning experience.

First and foremost, I would like to express my deepest gratitude to my esteemed supervisor, Dr. Oya Celiktutan from King's College London. Her unwavering commitment to my scholarly development, patient guidance, and insightful feedback have significantly enriched my understanding of research methodologies and critical thinking. Dr. Celiktutan's mentorship has been a beacon of inspiration, motivating me to strive for excellence in my work.

I would also like to extend my sincere gratitude to Mitesh Patel and Tomasz Bednarz from NVIDIA California, for their co-guidance and support in my work. Their insights have been invaluable in shaping the direction of my research.

I extend my sincere appreciation to PhD Edoardo Cetin and Ruiqi Zhu at King's College London for their exceptional guidance in refining my work and providing me with valuable insights into innovative research methods. Their generous contributions have had a profound impact on the quality and depth of my research endeavors.

Furthermore, I am profoundly indebted to Dr. Rui Liu, Assistant Professor in the College of Aeronautics & Engineering at Kent State University for introducing me to the captivating world of robotics. His unwavering commitment to nurturing young researchers, combined with his extensive expertise in robotics, has not only broadened my horizons but also ignited a profound passion within me for delving into the intricate intersection of machine learning and autonomous systems. Dr. Liu's mentorship has been an enlightening and transformative experience, and I am genuinely grateful for the invaluable knowledge and inspiration he has imparted.

I am grateful to my fellow classmates and colleagues for fostering a collaborative and intellectually stimulating environment. Our shared discussions and mutual support have made this academic journey both engaging and rewarding.

I would be remiss not to acknowledge the unwavering support of my family. Their belief in my potential and their encouragement have been a constant source of strength throughout this endeavor.

I extend my sincere gratitude to the BDMA program for affording me the opportunity to embark on this remarkable journey. The experiences gained and lessons learned have truly contributed to the richness of this academic exploration.

In conclusion, this thesis would not have been possible without the unwavering support and guidance of my mentors, colleagues, and family. As I move forward, I am grateful for the knowledge and experiences gained during this journey, and I am excited to apply them to new challenges and opportunities.

Contents

Abstract:
This work presents a digital twin of the Toyota Human Support Robot (HSR) using the Omniverse Isaac Sim simulator. A Universal Scene Description (USD) model of HSR is offered to enable photorealistic simulation. Benchmark scenarios focusing on human-robot interaction (HRI) are proposed, including tasks involving physical contact and collaboration. Reinforcement learning methods are tested on the benchmarks as a proof-of-concept. The potential of digital twin technology for robotic sim2real transfer is demonstrated through a real2sim implementation. This research aims to provide the community with an open-source HSR digital twin and HRI benchmarks to advance household robotics.

Keywords: Human-robot interaction, digital twin, simulation, benchmarking, reinforcement learning

Résumé:
Ce travail présente un jumeau numérique du robot d'assistance Toyota Human Support Robot (HSR) en utilisant le simulateur Omniverse Isaac Sim. Un modèle Universal Scene Description (USD) de HSR est proposé pour permettre une simulation photoréaliste. Des scénarios de référence axés sur l'interaction homme-robot (HRI) sont proposés, comprenant des tâches impliquant un contact physique et une collaboration. Des méthodes d'apprentissage par renforcement sont testées sur les références comme preuve de concept. Le potentiel de la technologie de jumeau numérique pour le transfert sim2réel robotique est démontré à travers une implémentation réel2sim. Cette recherche vise à fournir à la communauté un jumeau numérique HSR open-source et des références HRI pour faire progresser la robotique domestique.

Mots-clés: Interaction homme-robot, jumeau numérique, simulation, étalonnage, apprentissage par renforcement

Chapter 1

Introduction

1.1 Background

Human-Robot Interaction (HRI) as a discrete problem was envisioned by the 20th-century author Isaac Asimov in 1941, within the pages of his novel "I, Robot," where he introduced the now-famous Three Laws of Robotics. Over the years, HRI has evolved into a transformative research area with the potential to revolutionize numerous industries. It encompasses a wide range of applications, including both industrial and domestic settings.

Domestic robots, also known as household robots, are designed to improve the quality of human life in the aspect of routine domestic chores assistance, education, and entertainment. According to Polaris Market Research, the global household robots market Size & share was valued at USD 8.03 Billion in 2021 and is predicted to increase at a CAGR of 16.8%, to reach USD 31.99 Billion by 2030. It is said that we might exceed a one-to-one ratio of household tele-robots to humans in the future.

Some well-established domestic robot representatives include NAO, Pepper, Silbot, Sanbot Elf, Astro, and Toyota Human Support Robot (HSR), which serve for both commercial and research purposes. Moreover, emerging humanoid robots, such as Ameca from Engineered Art, Optimus from Tesla Bot, EVE from 1X invested by OpenAI, and Atlas from Boston Dynamics, etc., are inspiring examples of the latest innovations in this field.

The anticipated surge in household robots relies on the capabilities powered by deep learning, which enables the robot to "see", "hear" and "touch", comprehending its surroundings. For instance, Robotic Transformer 2 (RT-2), a vision-language-action (VLA) model by Deepmind, can translate human language instructions and surrounding environment visuals into robot control signals. However, training the deep learning model with real robots in the real world can be unsafe, costly, and inefficient. It's either the robot, or the interacted target, that is too precious to be experimented on for too many times.

Thus simulation plays a pivotal role by providing realistic virtual environments for extensive training. Simulators like PyBullet, MuJoCo, Gazebo, Issac Sim are based on different physics engines along with certain tradeoffs. But achieving proficiency in the simulator soly doesn't mean the performance can be migrated into the real world. For example, a robot good at grasping cubes doesn't guarantee a successful manipulation on complex objects in the real world. The gaps between the reality and simulation environments need to be filled, that is where the digital twin (DT) takes the place.

DT is a real-time virtual replica of a Physical Twin (PT) (i.e., physical entities of a robot and its surroundings)[47]. A photorealistic and high-fidelity simulator will allow DT mirrors PT physically. A useful use case would be remote surgery. Furthermore, owing to the simulator's advanced physics engine, precise kinematics can be ensured, thus demonstrating its significance in scenarios like virtual crash tests for automobiles. In the field of robotics, the DT enables a robot to undergo multiple grasping trials with unseen

objects within the virtual environment before it's ready to take a move in the real world. What's more, the photorealism of the simulator gives chances to generate countless realistic virtual scenes to train the robots' vision perception and promise seamless transitions into the real world as well.

1.2 Motivation

The Toyota Human Support Robot (HSR) stands out as a noteworthy and commonly used mobile domestic robot. Since its introduction to the public in 2012, the Toyota HSR has gained widespread adoption within research labs focused on Human-Robot Interaction (HRI) worldwide.

To enhance the performance of Toyota HSR using Digital Twin (DT) technology, a highly realistic simulator that accurately mirrors real-world conditions is necessary. However, traditional robot simulators, i.e. Mujoco, lack photorealism. By contrast, Nvidia has come up with its state-of-art robotics simulation toolkit, Omniverse™ Isaac Sim, which allows practitioners to create robust, physically accurate, photorealistic simulations and synthetic datasets. Isaac Sim supports navigation and manipulation applications through Robotic Operating System (ROS) [54]. It simulates sensor data from sensors such as RGB-D, Lidar, and IMU for various computer vision techniques such as domain randomization, ground-truth labeling, segmentation, and bounding boxes. Those features make Isaac Sim a fundamental simulator for Digital Twin applications.

Omniverse Isaac Sim uses the Universal Scene Description (USD) interchange file format to represent scenes. Currently, no USD format model exists for the Toyota HSR; only the Unified Robotics Description Format (URDF) model was provided by the Toyota Research Institute in July 2017 [3] [5]. Although Isaac Sim supports URDF file imports, further tuning the physics parameters is necessary, in order to enable the robot to be controlled in a realistic way in the simulator. This paper will provide a complete USD model of Toyota HSR.

Moreover, despite Toyota HSR's worldwide popularity, although some HSR simulators have been built for users to play with it virtually [4], there is no universal simulation benchmark that allows researchers to train and compare their algorithm performance with the Toyota HSR. Existing robotics benchmarks primarily serve for non-mobile robots, such as Franka Emika Panda, Baxter, Kinova Jaco, PR2, Sawyer, UR series, among others, rather than telerobots. In this work, we will build several benchmark scenarios which have clear reinforcement learning (RL) task settings and rewards functions to fill this gap.

Furthermore, most robotics benchmarks focus on robot–object interaction scenarios (i.e. navigation, and object manipulations). Undoubtedly, these scenarios are fundamental for training robotics performance. However, benchmarks specifically designed for Human-Robot Interaction (HRI) are in severe scarcity. It is due to the difficulties to model the humans in the simulator, and also their dynamic movements, and extremely their interactive responses to robots. Given the household-oriented nature of the Toyota HSR, our benchmarks will prioritize scenarios involving human presence, filling the Human-Robot Interaction (HRI) benchmarking research void.

In summary, in this work we will 1) offer a Universal Scene Description (USD) model of Toyota Human Support Robot (HSR); 2) use Omniverse Issac Sim as the photorealistic

simulator, and wrap HSR as a standard Isaac Sim Robot equipped with multiple controllers; 3) build several generalized Human-Robot Interaction (HRI) benchmark tracks that involve physical Human-Robot Interactions (pHRI), which serves as a basis for more complex benchmark design in the future work; 4) test the benchmark with popular reinforcement learning algorithms; 5) and make a proof of concept for real2sim implementation as a meaningful endeavor of Digital Twin (DT) technology.

1.3 Chapter List

Chapter Introduction provides background on human-robot interaction, motivation for using digital twin simulation, and an overview of contributions.

Chapter Related Work offers literature review of HRI, simulation, Digital Twins and highlights research gaps.

Chapter Background details on the Toyota HSR robot and Omniverse simulation platform.

Chapter Methodology explains the approach for constructing the HSR model and building benchmark environments.

Chapter Experiments and Results documents experimental configurations, training, and results on the proposed benchmarks using RL.

Chapter Conclusion and Future Work summarizes contributions and limitations and discusses potential extensions and open problems for future work.

Chapter 2

Related Work

2.1 Overview

Human-robot interaction (HRI), simulation platforms, digital twins, and reinforcement learning are key areas in current robotics research. Their intersection provides new perspectives for advancing household robotics. This work explores the potential of a Toyota HSR digital twin system for home robots by building on related work.

2.2 Human-Robot Interaction (HRI)

Human-Robot Interaction (HRI) is an extensively researched topic. It focuses on the interactions and communications between humans and robots, requiring robots to perceive, interpret, and respond to human states [55]. HRI initially derived from the concept Human-Robot Collaboration (HRC), which focuses more on how humans and robots work together simultaneously to achieve shared goals. HRC tasks are subdivided in four categories [57], namely, object handover (i.e. handovering objects bidirectionally between humans and robots), object handling (i.e. human and robot carrying the same object collaboratively), collaborative assembling (i.e. collaborative assembling desktop PCs [25], candy tins [40], and USB adapters [23]), and collaborative manufacturing. In a broader context, the landscape explored in this study intersects with both HRI and HRC.

Regarding the "how to learn" problem for robots in HRI, a novel area called Human-Interactive Robot Learning (HIRL) [49] emerges. In HIRL, robots engage in dynamic learning via human interaction, e.g. Learning from Demonstration to accomplish cooking procedures under human demonstrations, Learning by Imitation to mimic dance moves, Learning from Human Feedback to observe rewards from verbal instructions, posture corrections, or human facial expressions, Teacher-learner adaptation to adapt to different humans' habits, and Human-in-the-loop lifelong learning to for continuous learning. These dynamic modes of learning bring profound insights in designing the HRI benchmarks.

2.3 Simulators for HRI

2.3.1 A Comparison

Simulators create a replication of the real world, however, they are never perfect. The gap between these two worlds is called the sim-to-real gap. And it is crucial when training robots using Reinforcement Learning Algorithms to transfer onto physical robots. Based on the task settings, the requirements for the simulator could be slightly different. Suppose a robot relies on cameras as sensory input, a simulator's photorealism will be an essential

feature, while a simulator used for robotic grasping may place more importance on physical accuracy.

A multifacet comparison among robotics simulators has been provided by [51], see Figure 2.1. We could see that although traditional simulators like MuJoCo, employ high-performance physics engines, they fall short in delivering photorealistic renderings, impeding full functioning to build the digital twins. Some simulators, such as Gazebo and Pybullet can construct high-fidelity scenes, but their functionalities are not as complete as Nvidia's Omniverse Isaac Sim.

Isaac Sim integrates a comprehensive set of functionalities, such as Robotic Operating System (ROS) support [54], connectors to other 3D modeling software, inverse kinematics, soft-body contacts, etc., filling the gaps of real2sim and sim2real in both ways. Besides, Isaac Sim has proven to be a mature Digital Twin simulation application. In the business context, it has served Amazon in building Warehouse Digital Twin, and BMW for validating and testing the virtual factories [9].

Simulator	RGBD + LiDAR	Force sensor	Linear + Cable actuator	Multi-Body Import	Soft-Body Contacts	DEM Simulation	Fluid Mechanics	Headless Mode	ROS Support	HITL	Teleoperation	Realistic Rendering	Inverse Kinematics
Airsim	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓, Unreal	✗
CARLA	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✓, Unreal	✗
CoppeliaSim	✓	✓	Linear only	✓	✗	✓	✗	✓	✓	✓	✓	✗	✓
Gazebo	✓	✓	Linear only	✓	✗	Through Fluidix	Through Fluidix	✓	✓	✓	✓	✗	✓
MuJoCo	✓	✓	✓	✓	✓	✓	Limited	✓	✗	HAPTIX only	HAPTIX only	✗	✗
PyBullet	✓	✓	Linear only	✓	✓	✓	✗	✓	✗	✗	✓	✗	✓
SOFA	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓, Unity	✗
UWSIM	RGBD only	✓	✗	✓	✗	✗	✗	✓	✓	✓	✓	✓, custom	✗
Chrono	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗	✓	✓, offline	✓
Webots	✓	✓	Linear	✓	✗	✗	Limited	✓	✓	✗	✓	✗	✗
NVIDIA Isaac	✓[44]	✓[4]	✓[44]	✓[44]	✓[4]	Through Flex [45]	Through Flex [35]	✓[44]	✓[44]	✓[39]	✓[36]	✓, Omniverse	✓[4]

Figure 2.1: Comparison among popular robotics simulators.

2.3.2 Building on Top of Fundamental Simulators

Furthermore, on top of the general simulators, there are interactive high-fidelity indoor environments built for robotics, i.e. iGibson [43] a large-scale object-centered environment, Habitat [59] and RoboTHOR [29] for embodied AI agents, and VRKitchen [35] to generate indoor scene in Nvidia Omniverse.

Specially, for the purpose of HRI, there are human-involved simulators where users can deploy their own agents and algorithms. Underneath, they use the general simulator like Gazebo, Unity, and Pybullet, and on top of it, they tailor for specific HRI contexts. For instance, MengeROS [20], HuNavSim [52], and SocNavBench [21] deal with robot social navigation task. Pedsim_ROS [36], GRADE [22] provide crowd simulation. Besides, there are simulators that offer high-level human social behaviors, beyond basic movements like walking and idle. For example, in IMHuS [33], the human is able to respond to the robot's request to press the elevator button for it, and in VirtualHome [53] the human can set up a dinner table. However, the physical interactions between humans and robots are rarely considered.

In order to build a simple, easy-to-use, and generalized HRI benchmark, we will use Omniverse Isaac Sim as the simulator. Animations will be loaded to model human movements.

2.4 Reinforcement Learning (RL) Benchmarks for HRI

Reinforcement Learning benchmarks offer standardized and well-defined tasks along with associated reward functions, which allow users to evaluate and compare the performance of their RL algorithms virtually.

In the field of robotics, there have been benchmarks for manipulations, i.e. SURREAL [32], Meta-World [64], RLBench [38], SAPIEN [62], SoftGym [46], IKEA Furniture Assembly environment [42], etc.

Regarding Human-Robot Interaction (HRI) benchmarks which involve human presence, ALFRED [58] stands for Vision-language-action (VLA) benchmark, mapping from natural language instructions and egocentric vision to sequences of actions for household tasks. Assistive Gym [31] focuses on physical robotic assistance to a patient, including 6 tasks Itch Scratching, Bed Bathing, Drinking Water, Feeding, Dressing, and Arm Manipulation, supporting robots PR2, Jaco, Baxter, Sawyer, Stretch, and Panda. Based on the MuJoCo physics engine, it suffers from oversimplified human models. Robot-Assisted Dressing [28] is another assistive benchmark assisting humans dressing deformable clothes, based on the Nvidia PhysX engine. HabiCrowd [60] is the first standard benchmark for crowd-aware visual navigation.

In the context of physical human-robot interactions, HandoverSim [24] deserves special mention. This benchmark remarkably simulates the intricate process of handovers between robots and humans. This represents a meaningful step forward in benchmarking partial-body physical Human-Robot Interaction (pHRI) scenarios, capturing the nuances of close physical interactions between robots and humans.

2.5 Digital Twin (DT) Associated HRI

2.5.1 Communication Methods with the Physical Counterpart

As mentioned in Chapter 1.1, the Digital Twins (DT) is known as a computer-generated equivalent of its Physical Twin (PT). DT interacts with its PT through sensors and actuators. Sensors gather data about the PT and their surroundings, which are then transmitted to the DT in real time. AI-enabled calculations or decisions are made in the DT simulator, and control signals will be sent back from DT to PT for execution on the fly.

2.5.2 Classification of Use Cases

A Digital Twin (DT) seamlessly integrates both functional and physics data, each serving a distinct purpose.

The importance of incorporating functional data into a DT lies in its capability to enable holistic information integration. For instance, DTs are employed in the construction of Smart Cities, facilitating the amalgamation of diverse data sources to create a comprehensive overview of urban operations.

On the other hand, for physics data, the significance of utilizing DTs stems from their inherent capacity for physics-based reasoning. Physics engines within DTs function as state transition mechanisms, realistically simulating how systems evolve over time. To serve for this purpose, both high photorealism to model the shape, and an accurate physics engine to model the kinematics are indispensable. This empowers DTs to forecast the future by simulating potential outcomes before they actually occur. The realm of Human-Robot Interaction (HRI) particularly benefits from this capacity.

2.5.3 Approaches to Implement DT in HRI

Despite the numerous applications of DT for human-robot contexts in different application areas, only a few review articles analyzed this joint scientific landscape of DT-associated HRI [30] [41] [37].

In general, in order to build HRI digital twin, it needs to fulfill several key components, i.e. the integration with a robotic middleware like Robotic Operating System (ROS) [54] for the purpose of real-to-sim and sim-to-real transferring; a photorealistic simulator equipped with an accurate physics engine used to model the environment, including the robot, the human, and the sensors [39]; and an available API to train or employ decision-making module for generating control signals.

2.5.3.1 Reconstruct the Reality in the Simulator

The first way to bridge between the physical world and its digital counterpart is to transfer real-world objects into the simulator. For instance, an ABB IRB 120T robotic arm mounted to a two-dimensional Festo linear actuator is trained to play table tennis with a human [18]. Firstly, data of human serving balls is captured by stereo pair cameras, and ball trajectories are extracted and fed into the simulator for the robot to train. Then the physical robot is able to deploy the trained policy and keeps updating its policy during real-time playing with humans.

However, daily objects are not as simple as ball shapes. When transposing real-world physical objects into the digital realm, particularly those reliant on intricate 3D shapes, the utilization of 3D reconstruction technologies becomes imperative.

For object reconstruction, state-of-the-art algorithms, such as the Nerf series, i.e. Nerf [48], Instant Nerf [50], and Neuralangelo [45], empower robots to interact with objects that mirror complex realistic shapes. For human reconstruction, notably, it detects both body and the intricate articulation of hand gestures, such as HybrIK [44], or even facial expressions. Nevertheless, for 3D full-scene construction, it remains a challenge when segmenting the small target object from the entirety of the scene. Current 3D segmentation algorithms mostly deal with large pieces of objects, such as indoor furniture segmentation [26].

2.5.3.2 Generate Synthetic Scenes in the Simulator

In spite of the challenges posed by the intricacies of 3D reconstruction, Digital Twin can be implemented in another way around.

When faced with limited or non-existent data, Digital Twins can harness their simulation's photorealism to generate synthetic data closely resembling real-world scenarios for training purposes. In this case, the should be trained using these synthetic vision inputs, establishing a connection to reality by applying the learned policies to actual camera sensor data. For example, Amazon's Warehouse Digital Twin allows the sorting robot to train with virtual parcels based on visual inputs and achieve practical high sorting skills before the warehouse is established. This largely decreases the project cycling time.

In the context of Human-Robot Interaction (HRI), one example is human-to-robot handovers [27]. In this scenario, both the simulated and real-world environments leverage RGB-D frames as implicit input data for Reinforcement Learning (RL) algorithms, thus bridging any gaps between simulation and real-world sensing. To this end, our benchmark track, Mirroring Human Gesture (as shown in Chapter 4.4.5) which adopts RGB-D camera data to train the agent, can be viewed as a proof of concept of utilizing Digital Twin to enhance robot social skills.

2.6 Opportunities and Gaps

Concerning the two methods for implementing Digital Twin to enhance Human-Robot Interaction, the first approach 3D reconstruction of the environment is currently not fully mature.

The author's exploration has included successful tests using cutting-edge algorithms, such as HybirK [44], to achieve real-time human body and hand pose reconstruction within Isaac Sim, and then Instant Nerf [50] to reconstruct complex-shaped objects. It's crucial to note that the human reconstruction process requires hardware with substantial RAM capabilities, high refresh rates, and low latency, which can pose challenges in practical implementation. Additionally, the object reconstruction process faces difficulties related to mesh noise, potentially affecting the fidelity of the reconstructed models.

Under such circumstances, the second approach, training the robot using visual inputs from synthetic scenes and migrating its visual-based control capacity to real camera sensors seems to be a more promising method at the current moment. To this end, HabiCrowd [60], the first standard benchmark for crowd-aware visual navigation, and HandoverSim [24], trained from RGB-D data inputs, are pioneering algorithms in this field, as mentioned in Chapter 2.4.

To effectively train robot skills using synthetic vision data, the simulated scenes should be as rich and varied as possible. High-fidelity simulators mentioned in Chapter 2.3.2 thrive to fill this gap.

For HRI purposes, realistically generating human presence in these virtual environments poses several challenges:

- Highly realistic human models are needed given the reliance on visual training data. However, increasing model fidelity often comes at the cost of physics accuracy. In Isaac Sim specifically, the animated humans lose collision features currently (see Chapter 4.3.1). Thus simulating body contact becomes a problem.

- The body's deformable geometry is hard to model for the simulated human.

- The animations require not just basic body movements, but also nuanced hand gestures and facial expressions to capture the intricacies of human motion and nonverbal communication.
- Truly interactive, dynamic responses to the robot's actions are difficult to synthesize in simulated humans. But this interactivity and feedback is critical for Human-Robot Interaction, as stated in the concept of Human-Interactive Robot Learning (HIRL) [49] (Chapter 2.2).

This work will explore scenarios incorporating animated humans to bridge the gap of HRI simulations, despite these limitations.

Chapter 3

Background

This chapter provides an in-depth overview of the Toyota Human Support Robot (HSR) and its technical specifics. Additionally, the chapter steps through the fundamentals of Omniverse Isaac Sim, and outlines how they enable robotics research and reinforcement learning. This background on HSR and Isaac Sim establishes the fundamental knowledge needed to understand the approach and methodology used in this project.

3.1 Toyota Human Support Robot (HSR)

In our study, the Toyota Human Support Robot (HSR) was explored as a robot for this use case. The HSR is a robotic assistant designed for home use. It can be controlled through voice commands or a tablet, and it has the ability to move autonomously by recognizing its surroundings. The robot is equipped with a Graphical User Interface (GUI)(Figure 3.1), and it runs on the Robotic Operating System (ROS), an open-source platform [54]. The HSR also features Mapping and Obstacle Avoidance capabilities, allowing it to create maps of its environment and navigate around obstacles.

Figure 3.1: HSR

3.1.1 HSR Technical Specs

HSR achieves both seated and standing heights, ideal for participating in household settings. The HSR weighs 37 kg with a max speed of 0.8 km/hour. It stands at a height of 100cm with an arm length of 60 cm. The neck can extend up to 34.5 cm, allowing for a maximum total height of 135 cm. For more details see Figure 3.2 and Table 3.1. Additional details about the HSR technical specs can also be found in papers [19] [63].

Figure 3.2: HSR Dimensions

Specification	Details
Dimensions	43cm (diameter) x 60cm (arm) x 135cm (height)
Weight	37 kg
Max. Payload	1.3 kg
Max. Speed	0.8 km/h
Avg. Run Time	2 – 3 hours
Terrain	5mm max step size
	50 max incline
Sensors/Components	Microphone Array
	RGB-D Camera
	Wide-Angle Camera
	Force Torque Sensor
	Stereo Camera
	IMU
	Laser Range Sensor

Table 3.1: HSR Technical Specs

3.1.2 HSR Degree of Freedoms (DoFs)

HSR possesses a 5-DoF arm fitted with a gripper, see Figure 3.3. It has totally 26 degrees of freedom (26-DoF). Each joint' index, name, type, and range are listed in Table 3.2.

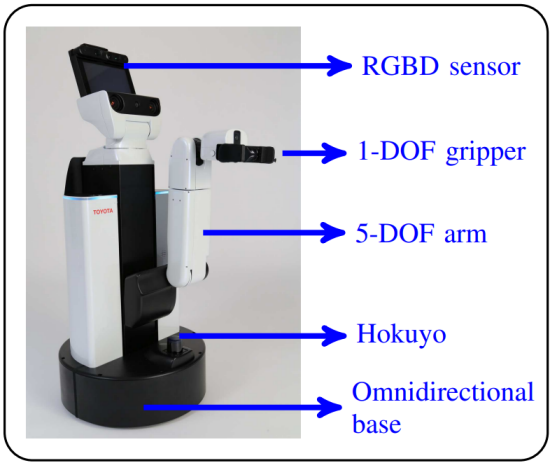

Figure 3.3: HSR DoFs

Joint Index	Joint Name	Joint Type	Joint Range
0	arm_roll_joint	Revolute	$[-119.75, 220.02]$
1	wrist_flex_joint	Revolute	$[-110.01, 69.90]$
2	arm_flex_joint	Revolute	$[-150.11, 0]$
3	wrist_roll_joint	Revolute	$[-110.01, 210.28]$
4	arm_lift_joint	Prismatic	$[0, 0.69]$
5	base_roll_joint	Revolute	$[-180, 180]$
6	torso_lift_joint	Prismatic	$[0, 0.345]$
7	hand_l_proximal_joint	Revolute	$[-45.72, 71.04]$
8	hand_motor_joint	Revolute	$[-45.72, 71.04]$
9	hand_r_proximal_joint	Revolute	$[-45.72, 71.04]$
10	base_l_drive_wheel_joint	Revolute	none
11	base_l_passive_wheel_x_frame_joint	Revolute	none
12	base_r_drive_wheel_joint	Revolute	none
13	base_r_passive_wheel_x_frame_joint	Revolute	none
14	head_pan_joint	Revolute	$[-220.02, 100.27]$
15	hand_l_spring_proximal_joint	Revolute	$[0, 40.00]$
16	hand_r_spring_proximal_joint	Revolute	$[0, 40.00]$
17	base_l_passive_wheel_y_frame_joint	Revolute	none
18	base_r_passive_wheel_y_frame_joint	Revolute	none
19	head_tilt_joint	Revolute	$[-89.95, 29.29]$
20	hand_l_mimic_distal_joint	Revolute	$[-39.99, 0]$
21	hand_r_mimic_distal_joint	Revolute	$[-39.99, 0]$
22	base_l_passive_wheel_z_frame_joint	Revolute	none
23	base_r_passive_wheel_z_frame_joint	Revolute	none
24	hand_l_distal_joint	Revolute	$[-45.72, 71.04]$
25	hand_r_distal_joint	Revolute	$[-45.72, 71.04]$

Table 3.2: HSR 26-DoF Joints Details. The units of Joint Range are in degrees for the revolute joints, and in meters for prismatic joints.

3.1.3 HSR Sensors and Equipments

The HSR provides an array of sensors of multiple modalities to capture inputs from the environment and other agents [63]. The head (Figure 3.4) of the HSR mounts a microphone, a depth-sensing Red Green Blue (RGB-D) camera, a wide-angle camera, and a higher-resolution stereo RGB camera, which captures sound, images, and the depth map of the scene. Although these sensors are all front-facing, the 320-degree pan and 120-degree tilt allow reading visual input from a large part of the three-dimensional environment. What's more, the hand of HSR mounts an additional camera and a force sensor to provide visual input and the force on the wrist while gripping an object [19].

The vision perception data used for vision-based RL is collected from the RGB-D camera (Head 3D sensor).

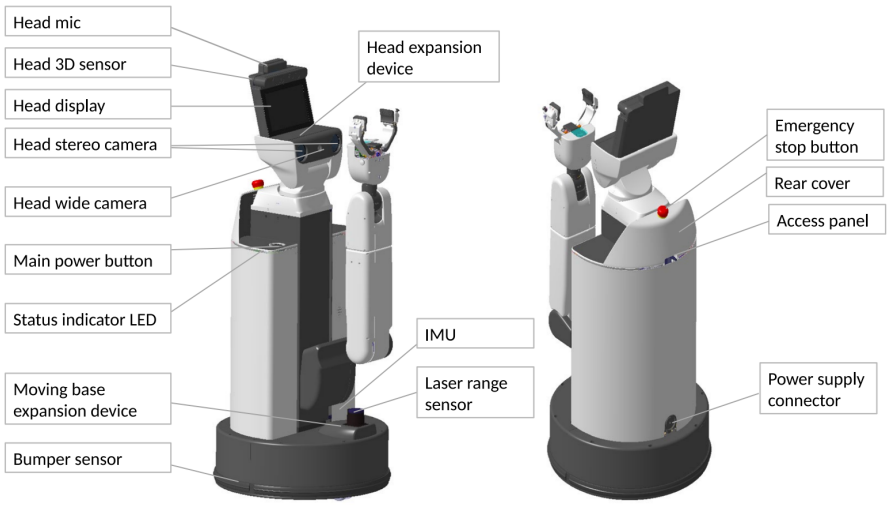

Figure 3.4: HSR Sensors and Equipments

3.2 Omniverse and Isaac Sim

3.2.1 NVIDIA Omniverse™

NVIDIA Omniverse™ [11] is an extensible platform for virtual collaboration and real-time, physically accurate simulation. It enables individuals and teams to develop Universal Scene Description (USD)-based 3D workflows and applications. It consists of five core components, see Figure 3.5. Out of which, Omniverse Kit is the SDK for building Omniverse applications like Issac Sim, see Figure 3.6.

In kit, the extensions can be viewed as versioned packages with a runtime enabled/disabled state. One extension can depend on another, which makes the kit application development very modular.

Component	Name	Description
	Omniverse Nucleus	The central database and collaboration engine of Omniverse. With Nucleus, users share and modify representations of virtual worlds.
	Omniverse Connect	The libraries that connect popular content creation tools to Omniverse. With Connectors, users continue to work in their favorite software applications, such as SketchUp, Maya, and Unreal Engine, while benefitting from other Omniverse tooling.
	Omniverse Kit	The toolkit for building native Omniverse Applications, Extensions, and Microservices.
	Omniverse RTX Renderer	An advanced, multi-GPU renderer based on NVIDIA RTX™ that powers both real-time ray tracing and referenced path tracing.
	Omniverse Simulation	A powerful suite of tools and SDKs that simulate a physically accurate world.

Figure 3.5: Omniverse Components

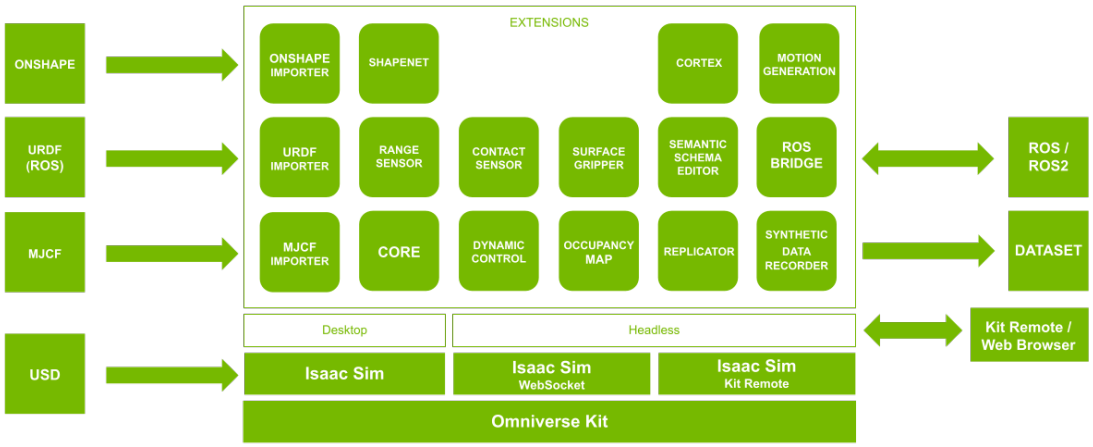

Figure 3.6: Omniverse Kit

3.2.2 NVIDIA Isaac Sim

NVIDIA Isaac Sim™ [10], is a robotics simulation application built by Omiversse Kit on top of Omniverse, encompassing multiple functionalities of Omniverse Kit.

3.2.2.1 Isaac Sim Features

Firstly, it leverages Omniverse Kit's powerful simulation technologies including advanced GPU-enabled physics simulation with PhysX 5, photorealism with real-time ray and path tracing, and MDL material definition support for physically-based rendering, see Figure 3.7.

Secondly, inherited from Omniverse Kit, Isaac Sim supports communications with physical robots through ROS, transmitting and simulating data from sensors, such as RGB-D, Lidar, and IMU for various computer vision techniques such as domain randomization, ground-truth labeling, segmentation, and bounding boxes.

Thirdly, assets from popular 3D computer graphics applications, like Maya, Unreal Engine, Unity, and Blender can be connected seamlessly by Omniverse Connectors with Isaac Sim. Those connectors convert 3D assets' format to USD that can be imported into Isaac Sim flawlessly. Nonetheless, Isaac Sim supports other formats' imports as well, i.e. MuJoCo Modeling XML File (MJCF) and Unified Robot Description Format (URDF) representations of robot models, it is benefited by Kit's extension modules, the MJCF importer and the URDF importer, see Figure 3.6.

Lastly, in developer mode, users can run arbitrary Python scripts as long as it's Kit-based apps.

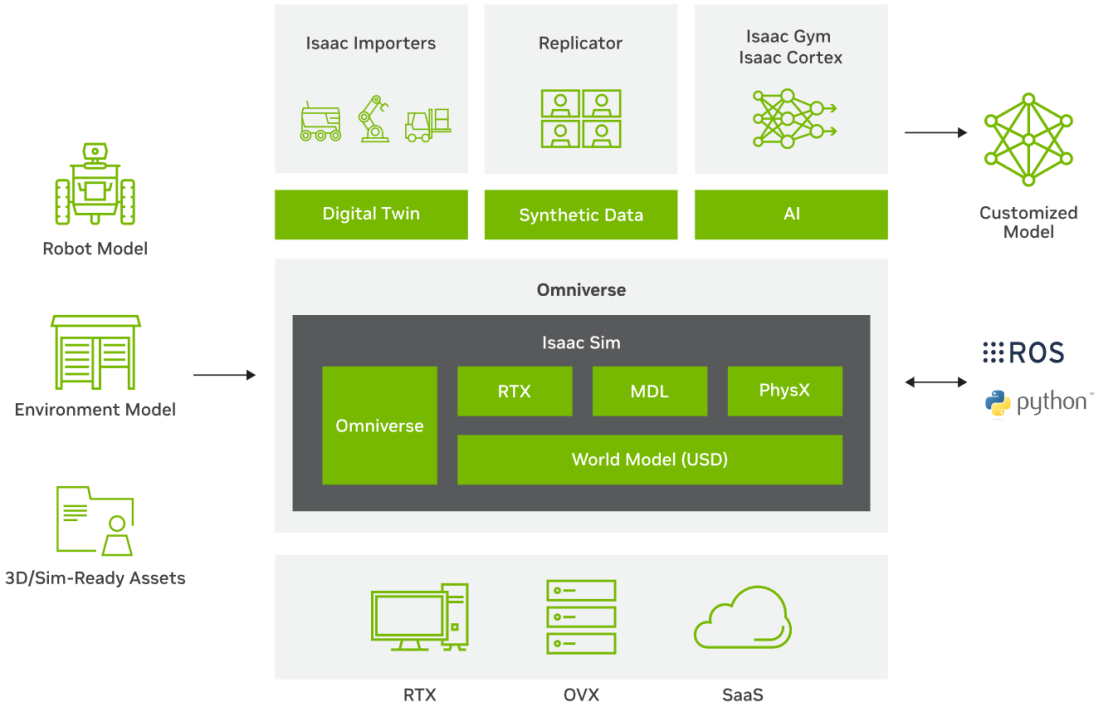

Figure 3.7: Isaac Sim Complete Solution Stack

3.2.2.2 Isaac Sim for Reinforcement Learning (RL) Training

There exist two toolkits of reinforcement learning (RL) simulation environments in Isaac Sim. One is Omniverse Isaac Gym extension, built as an inner module of Issac Sim [13], and the other is a standalone app Isaac Gym [6].

The standalone Isaac Gym is an initial release of tensor-based Gym APIs as part of the NVIDIA Omniverse Isaac Sim 2022.1. Subsequently, new versions of Isaac Sim have been released, including 2022.2.0 and 2022.2.1, and soon will be 2023.1.0. As per the official website, the standalone Isaac Gym will remain available until the Omniverse Isaac Gym functionality is feature complete. Thus the standalone Isaac Gym is excluded from long-term considerations.

In this paper, we will adopt the Omniverse Isaac Gym extension's approach to build our customized RL environment, similarly as demonstrated in the official Jetbot reinforcement

learning example [7]. It will ensure effective communication with other backend modules and extensions in Isaac Sim, such as the main extension omni.isaac.core and pxr APIs in Omniverse Kit.

3.2.3 Human Animations in Isaac Sim

3.2.3.1 USD Skeleton adn USD Animations

Omniverse Isaac Sim uses the Universal Scene Description (USD) interchange file format to represent scenes, offering support for the USD Skeletons format.

USD Skeletons serve as a means to represent bones within a hierarchical arrangement, allowing binding skin and animations onto it. USD Animations are essentially sequences of time-dependent data capturing the positions and orientations of each joint within the body. In detail, these animations encompass four primary attributes that can be retrieved in the API: time samples, joint names, joint positions, and joint orientations. However, owing to the intricate interdependencies among various joints, the manual definition of these four attributes falls short of producing realistic and ready-to-use animations.

The USD skeletons employed within Isaac Sim permit diverse bone structures to be imported as unified USD Skeletons format but keep their own original hierarchies. Well-known bone structures like SMPL [15] and SMPL-X [16] are widely used in human 3D Pose Estimation algorithms, designed to capture either body-only or body-and-hand poses. Distinct platforms often define their own unique bone structures. Different species and genders of course have different bone structures as well. Consequently, applying the animation from one bone structure type to another is not a common thing.

3.2.3.2 Human Animations Resources

Since our task is related to Human-Robot Interactions, it's important to explore the accessible animation resources in the context of Isaac Sim. Based on granularity, the animations can be categorized into three levels as body-only (i.e. walk, idle, sit), hand-involved (i.e. handovers, pointing, grasping), and facial expressions included (i.e. laugh, amazed). Several animation resources can be found below in order to build human-invloved benchmarks.

Omniverse Nucleus. As mentioned in Figure 3.5, Omniverse Nucleus [12] is a central database containing a wide range of asset samples. Notably, a collection of 24 animations is available in the root path /Isaac/2022.2.0/Isaac/People/Animations. These animations encompass various actions such as idle, walking, sitting down, typing on a keyboard, and pressing buttons. Among these, there are 12 animations dedicated to walking movements.

Omniverse Machinima App. Animations can also be generated endlessly from Omniverse Machinima App [8]. It has a few excited features. It allows to capture human motions using live cameras and make a 3D-animated character in real-time, see Figure 3.8. And thus numerous animations can be generated by the user and import them into Isaac Sim seamlessly. Regrettably, it omits the inclusion of intricate hand gestures, thus diluting its efficacy within scenarios involving physical Human-Robot Interactions. Moreover, its Audio2Face and Audio2Gesture technologies can generate facial or body animations based on the language content and tones' emotions.

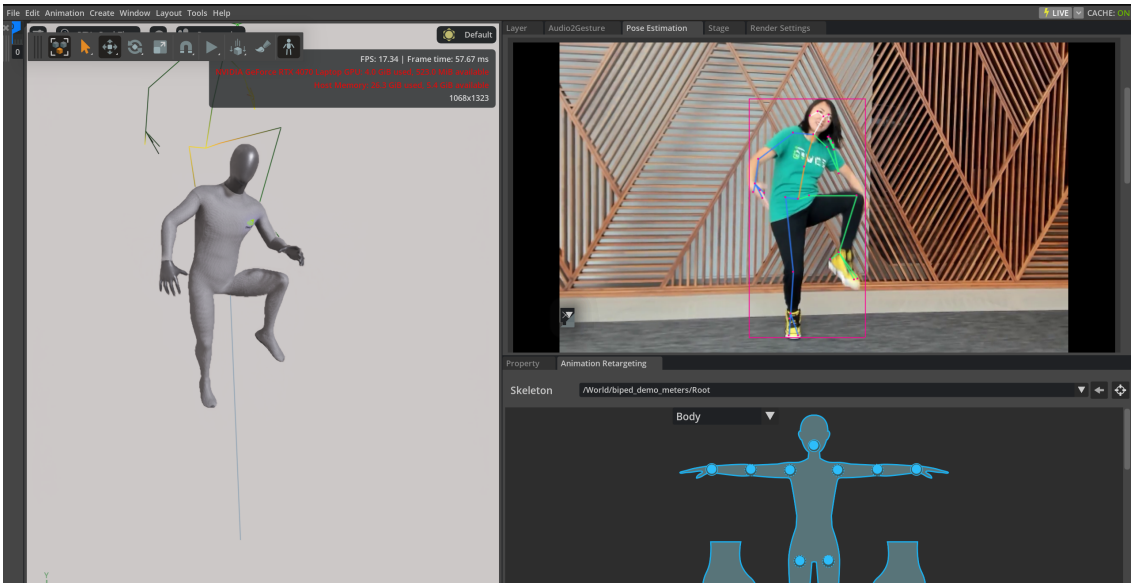

Figure 3.8: Real-time Motion Capturing in Omniverse Machinima App

External Apps through Omniverse Connectors. The connectors provided in Omniverse to external Apps allow the animation compatibly imported from external sources. The iClone and Actorcore by Reallusion are both excellent commercial character generators, supporting the animation exported in Omniverse USD format. As of August 2023, Actorcore provides more than 2300 motions, out of which 15 are free.

External Animation Datasets. Mixamo offers more than 2000 animations, that can be downloaded as Fbx for Unity format, and imported into Isaac Sim. As mentioned before, the intricacies of migrating these animations across different skeletal frameworks limit the flexibility of modifying actors' appearances.

Chapter 4

Our Methodology and Approach

This chapter illustrates the way to construct the Toyota HSR model in Isaac Sim and the way to control it, then the reinforcement learning benchmarks used to train it.

4.1 Toyota HSR Model Construction in Isaac Sim

4.1.1 URDF Importer

The official Toyota HSR model was provided in 2017 in Unified Robotics Description Format (URDF) for Robotic Operating System (ROS), as stated in Chapter 1.2. The following procedures are implemented to convert it to Universal Scene Description (USD) format:

Two official repositories [3] [5] which separately contain HSR mesh and URDF model are placed into one folder. A GitHub repository is provided for this step [61]. Then use the URDF Importer GUI (Figure 4.1) to import and visualize the HSR model. This could also be done by Isaac Sim Python API.

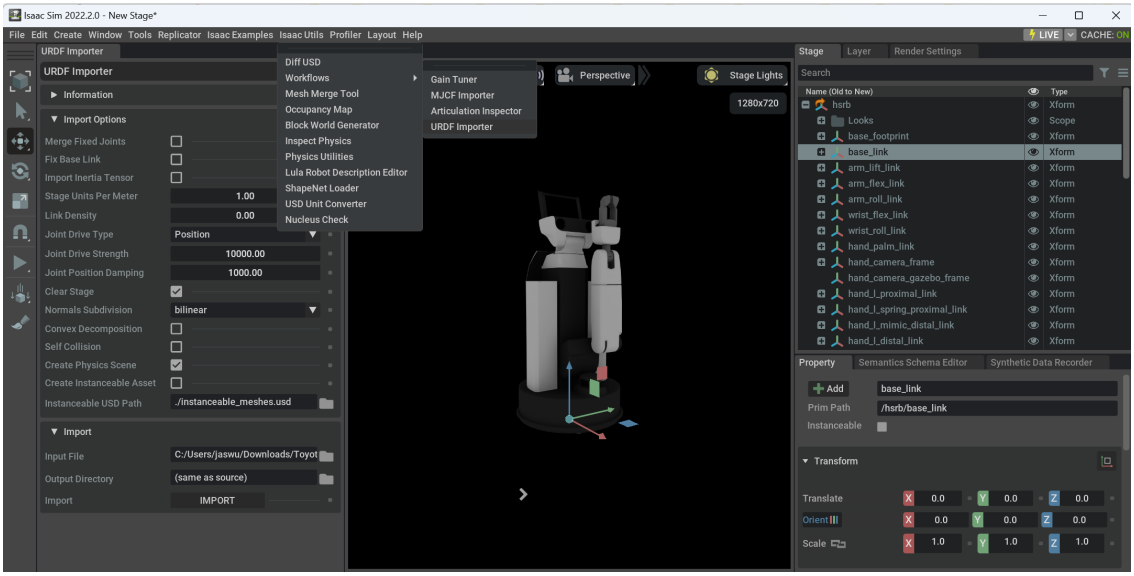

Figure 4.1: URDF Importer GUI

4.1.2 Tuning the Physics Parameters of USD

Add Camera. As mentionded in Chapter 3.1.3, HSR is equipped with four cameras, they are in the USD model. The vision perception data used for vision-based RL is collected from the RGB-D camera (Head 3D sensor), see Figure 4.2.

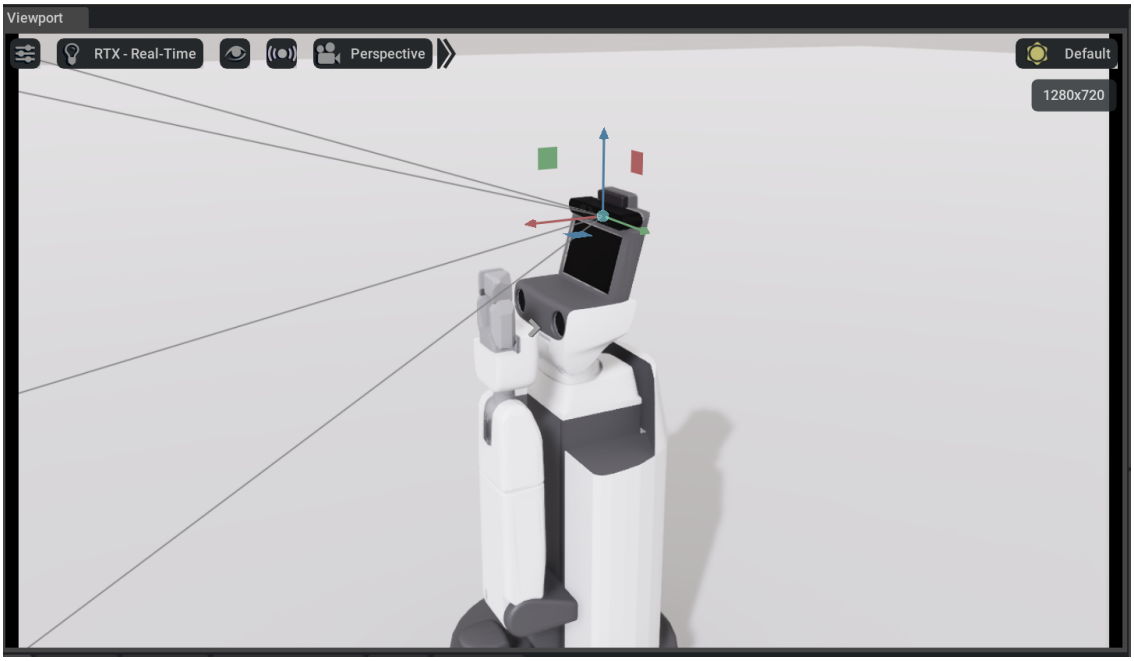

Figure 4.2: The RGB-D Camera

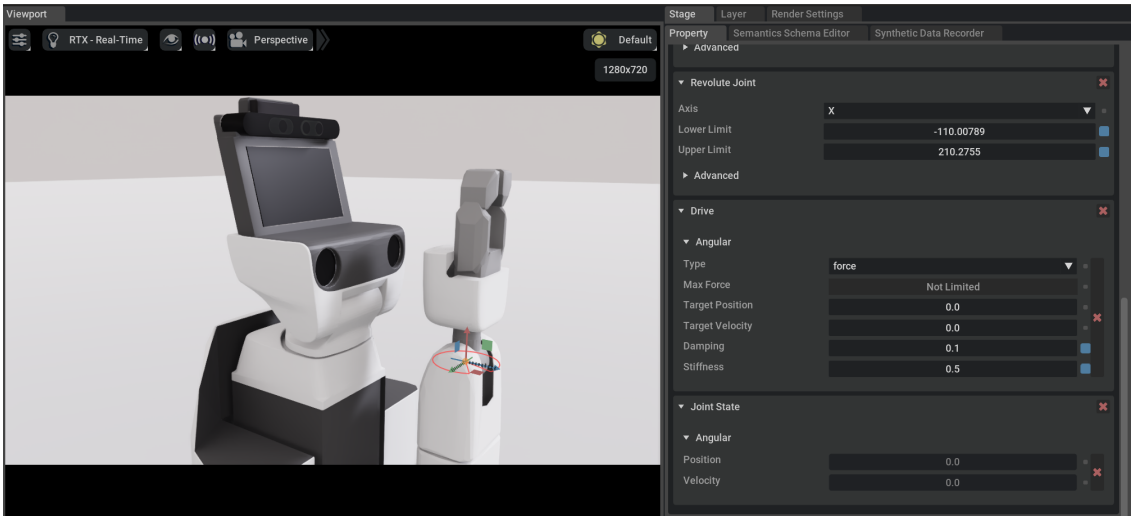

Figure 4.3: The Joint Drive

Add Joint Drives. Excluding the passive wheels and fixed joint, the drives are added onto each prismatic and revolute joint. The Drive type is set as 'force' with no maximum limit. The damping and stiffness will largely affect the stability and the responding speed. For the moment, the detailed articulator specs are not available for the author. According to the industry normal, the damping factor of robot manipulator joints is set at about $0.1 - 1\text{kg} \cdot \text{m}^2/s/\text{rad}$, and the stiffness is around $5 - 100\text{N/m}$.

In this paper, the damping factor and stiffness are set according to the control effect in the simulator. Two control modes are provided, the target position, or the target velocity. Target position control requires a tiny damping-to-stiffness ratio so that the

robot converges to the target angles/distance (for revolute joints and prismatic joints respectively) quickly. The absolute value of stiffness also matters, for example, a large stiffness will cause the robot to spin and fall when moving the arms.

Reasonably, the wheels are controlled by target velocity, and the arm is controlled by target positions. Because the wheels are moving at a certain speed, and the arm moves to a target location. Thus the default damping and stiffness in the USD model are tailored to serve this purpose.

Set frictions. The gripper's material is assigned a friction coefficient rate, both static and dynamic, at 0.5.

4.2 HSR Joint Control

4.2.1 Joint Control Modules in Isaac Sim

Before diving into the reinforcement learning benchmarking environment, it's essential to achieve robust joint control. Three practical ways to implement it. Either way, users are allowed to apply joint velocity control or joint position control. In the context of joints, the concept of position is slightly different from common sense. For the revolute joint, the position is a certain rotation angle; for the prismatic joint, it is a sliding distance.

4.2.1.1 UsdPhysics.DriveAPI Module

UsdPhysics.DriveAPI is the most fundamental control method, since Isaac Sim is a USD-based application, and underneath it is written in USD. Thus the USD module usage in Isaac Sim is consistent with the official OpenUSD Python Libraries [14]. The HSR can be controlled as follows.

```
from pxr import UsdPhysics
joint_prim=stage.get_prim_at_path("/hsr/arm_lift_link/arm_flex_joint")
drive = UsdPhysics.DriveAPI.Apply(joint_prim, 'angular') # revolute joint
drive.CreateStiffnessAttr(0.5)
drive.CreateDampingAttr(0.1)
drive.CreateTargetPositionAttr(90)
drive.CreateTargetVelocityAttr(10)
```

4.2.1.2 Dynamic Control Module

The omni.isaac.dynamic_control extension [2] provides a set of utilities and functions to explicitly control physics joints.

```
from omni.isaac.dynamic_control import _dynamic_control
dc = _dynamic_control.acquire_dynamic_control_interface()
robot_dc = dc.get_articulation("/hsr")
dc.wake_up_articulation(robot_dc)
dof_ptr = dc.find_articulation_dof(robot_dc, "base_l_drive_wheel_joint")
dc.set_dof_velocity_target(dof_ptr, 10*random.random())
dc.set_dof_position_target(dof_ptr, 10*random.random())
```

4.2.1.3 Articulations module

The Robot class inherits the Articulations [1], while HSR inherits the Robot. Wrapping HSR as a standard robot has several benefits. Similar to Dynamic Control, Articulations from omni.isaac.core extension offer explicit functions [1] to control the robot joints.

What's more, the ready controllers designed for other robots can be reused. One example is the DifferentialController module borrowed from omni.isaac.wheeled_robots extension. It asks for the input of the robot's linear and angular speed and turns it into the wheel rolling velocity in rad/s. In HSR, the wheel controller is implemented in the same way.

```
from omni.isaac.hsr.robots import HSR
from omni.isaac.hsr.controllers.differential_controller import
                                        DifferentialController

wheel_controller = \
  DifferentialController(name="wheel_controller",\
    wheel_radius=0.04, wheel_base=0.133*2)

# Actions are of linear speed and angular speed of the robot body
# The wheel_controller will convert them to wheel rolling speed
HSR.apply_wheel_actions(wheel_controller.forward(actions=[0.1,math.pi]))
```

In HSR's navigation tasks, the last approach will be adopted for control. In HSR's grasping task, the USD Drive method will be applied directly.

4.2.2 HSR Controllers

Three controllers are built for Toyota HSR. Underneath, each of them is programmed by control modules listed in Chapter 4.2.1, and by means of either velocity or position joint control.

4.2.2.1 Wheel Controller

For navigation tasks, only wheels are allowed to move. There are 4 wheels at the base of HSR, two front wheels and two back wheels. Back wheels are drive wheels, indexed [10,12] at Table 3.2. Front wheels are omnidirectional, moving along with drive wheels, and each of them has 3-DoF rotatation along all x, y, and z axes, see index [11, 13, 17, 18, 22, 23] at Table 3.2.

The back driving wheels are controlled by DifferentialController, as stated in the Articulation control method in Chapter 4.2.1.3.

4.2.2.2 Gripper Controller

For the grasping task, the gripper is related to the joint indexes [15, 16, 24, 25] (Table 3.2) and only has 1-DoF, open or close. Underneath, it is controlled by 4 gripper joints velocities.

4.2.2.3 Arm Controller

The arm-related joints are indexed [0,1,2,3,4] (Table 3.2), however, the base_roll_joint indexed at [5] allows the robot's base to rotate clockwise or reversely, thus it is a critical joint and will affect the arm poses. We add it to the arm controller as well.

The 6 joints are controlled by USD Drive API and are to be set at a target position or a target velocity.

4.3 Human Animations

The animations used in this work is mainly from Omniverse Nucleus sample animations, see the first animation source in Chapter 3.2.3.

4.3.1 Troubleshooting Human Animation Issues

This paper aims to offer a physical Human-Robot Interaction (pHRI) benchmark, and Isaac Sim offers API to set an object as a collider, rigid body, or both, so that whenever a physics collision happens between two bodies, the Collision Reporter will notice that. However, the collision report of animations in Isaac Sim is currently a bug. As shown in Figure 4.4, the human sits down, but its collision surface doesn't follow and stays at its original place, thus the Collision Reporter in Isaac Sim can not be successfully detect the dymaic animation collisions. This bug is said to be fixed in the next version release of the Isaac Sim platform.

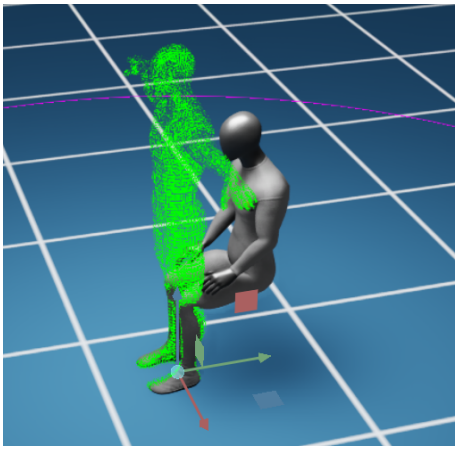

Figure 4.4: The Bug of Immovable Collision in Issac Sim

An alternative approach to address this issue is to utilize the hierarchical joints present in the USD skeleton (refer to Figure 4.5), as discussed in Chapter 3.2.3.1. The joint positions can be extracted in the global/world frame. Whenever the coordinates of these joints overlap with those of the Toyota HSR, a collision event is logged. Consequently, in this study, the human model is designated as a non-colliding entity. It can penetrate with the robot without any automated collision detection, but collision instances will be identified manually.

Figure 4.5: Use Joint Position Data to Check Animation's Collisions

A code snippet to retrieve the skeleton joints' position data under the world frame is shown below.

```
from pxr import UsdSkel
skeleton_prim = stage.GetPrimAtPath(skeleton_path)
skel_cache = UsdSkel.Cache()
skel_query = skel_cache.GetSkelQuery(UsdSkel.Skeleton(skeleton_prim))
xfCache = UsdGeom.XformCache()
xfCache.Update()
# Transformation matrix in world frame
world_transforms=skel_query.ComputeJointWorldTransforms(xfCache)
```

This approach also opens up new possibilities. For instance, in a scenario where a human model hands over a book to a robot, the book can be positioned at the same location as the fingertip joint until it is transferred to the robot. By leveraging the positional data of the human skeleton's joints, intricate physical interactions can be devised and realized.

4.4 Benchmark Design

The benchmark covers human-robot interaction modes: three navigation tasks, one object manipulation, and one mirroring human gestures. The navigation tracks are designed from easy to difficult levels. The simplest is to get near to people, then follow people, then circle around obstacles to reach people.

Each benchmark scenario can be trained using either transformation data (comprising positions and orientations of skeleton joints and objects) or RGB-D vision sensor data as inputs. In this paper, we use the former for navigation and object manipulation benchmarks, and the latter for mirroring human gesturesbbenchmark.

Each benchmark's action space can be transferred semantically to joint target position control or target velocity control. In this work, as illustrated in Chapter 4.2.2, the constructed wheel controller, gripper controller, and arm controller are subject to velocity, velocity and position control for now.

4.4.1 Navigation - Approaching a Randomized Human

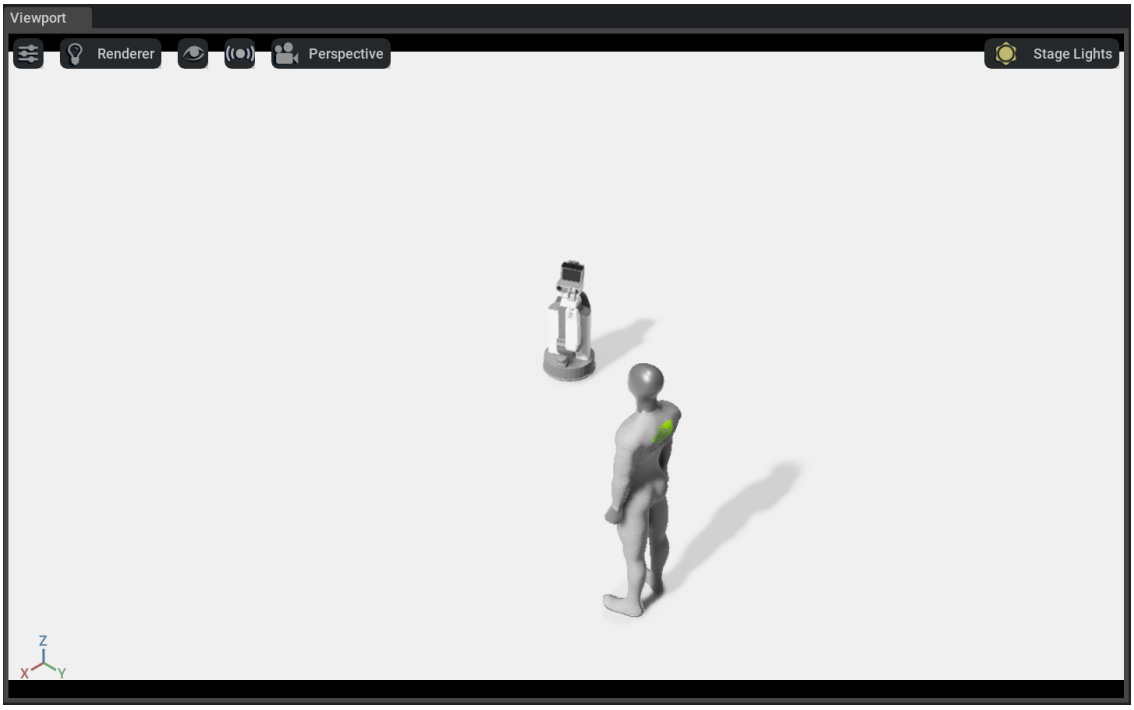

Figure 4.6: Navigation Scene - Approaching a Randomized Human

4.4.1.1 Human Initialization

At the beginning of each episode, a human is initialized at a random point whose x-axis and y-axis are within the range $[-5, 5]$.

4.4.1.2 Robot Initialization

The Toyota HSR is initialzed at position [0,0,0].

4.4.1.3 Human Behavior

The human is in an idle standing state at the initialized position, under animation stand_idle_
loop_skelanim offered by Omniverse Nucleus [12] (Chapter 3.2.3.2).

4.4.1.4 Robot Behavior

The HSR is set to try to get near where the human stands. If their distance is less than 0.3 meters, the episode is done.

4.4.1.5 Robot Controller

Wheel controller, robot body linear and angular velocity control.

4.4.1.6 Reward Functions

$$R_{total} = R_{distance} + R_{collision\_human}$$

$$R_{distance} = R_{prev\_dist\_robot\_human} - R_{curr\_dist\_robot\_human}$$

$$R_{collision\_human} = -20 \text{ if } R_{curr\_dist\_robot\_human} < 0.1, \text{ else } 0$$

The total reward encompasses the distance reward and collision reward. At each step, the distance is rewarded if the robot gets nearer to the human than at the last step. It is calculated by the previous step's human-robot distance minus the current step's human-robot distance. This reward design has an incremental feature, which is proven in practice to be more effective than merely using distance. The collision reward is set as a discrete large number -20 if the skeleton's Root joint overlaps with the robot base joint, in a manner that their distance is less than 0.1 meter. The built-in Collision Reporter in Isaac Sim is not used here, based on the reasons stated in Chapter 4.3.1.

4.4.2 Navigation - Following a Human

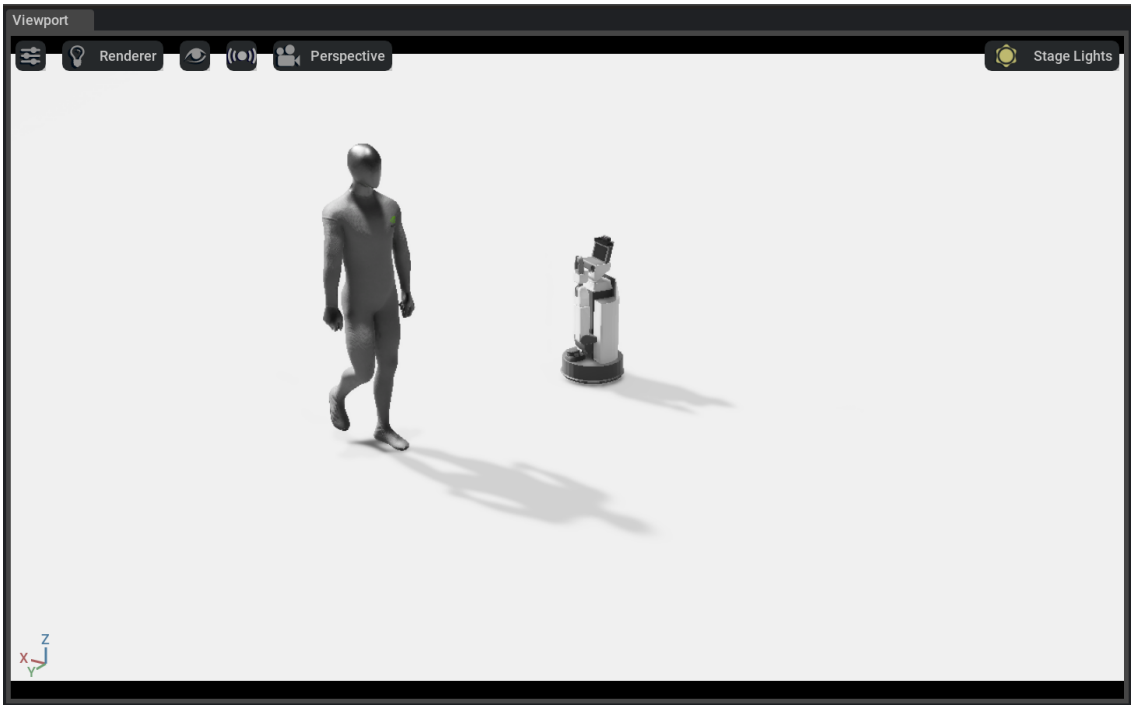

Figure 4.7: Navigation Scene - Following a Human

4.4.2.1 Human Initialization

At the beginning of each episode, a human is initialized at position [0,0,0] with a random choice of all walking mode animations offered by Omniverse Nucleus [12] (Chapter 3.2.3.2).

4.4.2.2 Robot Initialization

The Toyota HSR is initialzed at position [0,0,0].

4.4.2.3 Human Behavior

The human walks under a randomized walking mode animation.

4.4.2.4 Robot Behavior

The HSR is set to try to get near where the human locates in real-time. Safety concerns are solved in the collision punishment. There is no done until the end of the episode.

4.4.2.5 Robot Controller

Wheel controller, velocity control.

4.4.2.6 Reward Functions

$$R_{total} = R_{distance} + R_{collision\_human}$$

$$R_{distance} = R_{prev\_dist\_robot\_human} - R_{curr\_dist\_robot\_human}$$

$$R_{collision\_human} = -20 \text{ if } R_{curr\_dist\_robot\_human} < 0.1, \text{ else } 0$$

The reward function setting is same as the navigation task to getting near a randomized human, see Chapter 4.4.1.

4.4.3 Navigation - Approaching a Randomized Human with Obstacle Avoidance

Figure 4.8: Navigation Scene - Approaching a Randomized Human with Obstacle Avoidance

Human Initialization. At the beginning of each episode, a human is initialized at a random vacant position in an apartment room, and the sceen is downloaded from the TurboSquid website as a free asset [17]. It's first imported into Blender and then transmitted to Omniverse Isaac Sim via the Blender connector.

4.4.3.1 Robot Initialization

The Toyota HSR is initialzed at position [0,0,0].

4.4.3.2 Human Behavior

The human is in an idle standing state at the initialzed position, under animation stand_idle

_loop_skelanim offered by Omniverse Nucleus [12] (Chapter 3.2.3.2).

4.4.3.3 Robot Behavior

The HSR is set to try to get near where the human locates. If their distance is less than 0.3 meters, the episode is marked as done.

4.4.3.4 Robot Controller

Wheel controller, velocity control.

4.4.3.5 Reward Functions

$$R_{total} = R_{distance} + R_{collision\_human} + R_{collision\_obstacle}$$

$$R_{distance} = R_{prev\_dist\_robot\_human} - R_{curr\_dist\_robot\_human}$$

$$R_{collision\_human} = -20 \text{ if } R_{curr\_dist\_robot\_human} < 0.1, \text{ else } 0$$

$$R_{collision\_obstacle} = -0.5$$

The human-robot distance reward and collision reward is the same as the navigation task to getting near a randomized human, see Chapter 4.4.1. An extra reward robot-object reward is added. It is based on the Collision Reporter offered by Isaac Sim on 18 indoor objects (Table 4.1), which are set as rigid bodies with collisions preset. Each collision with each object will decrease the total reward by 0.5.

/World/scene/root/Coffee_table
/World/scene/root/Armchair__02
/World/scene/root/Armchair__01
/World/scene/root/Sofa
/World/scene/root/__chair__04
/World/scene/root/Armchair
/World/scene/root/Lamp_03
/World/scene/root/Stove
/World/scene/root/Plant_02
/World/scene/root/Drawer
/World/scene/root/Table
/World/scene/root/__chair__03
/World/scene/root/__chair01__02
/World/scene/root/__chair01__03
/World/scene/root/__chair__02
/World/scene/root/__chair01__01
/World/scene/root/__chair__01
/World/scene/root/Stove

Table 4.1: Obstacle Prim List

4.4.4 Object Manipulation - Pick and Place

Figure 4.9: Pick and Place Scene

Although vanilla object manipulation doesn't involve human presence, mastering grasping remains a fundamental skill for any household robot. As a result, the Toyota HSR undergoes training in this benchmark track to enhance its proficiency in object manipulation tasks.

4.4.4.1 Robot Initialization

The Toyota HSR is initialzed at position [0,0,0].

4.4.4.2 Robot Behavior

The HSR is set to try to grasp a cup from one table, and put it at a target place, the center of another table's surface.

4.4.4.3 Robot Controller

Arm controller, position control. Gripper controller, 1-DoF open/close control.

4.4.4.4 Reward Functions

$$R_{total} = R_{distance\_gripper\_object} + R_{distance\_object\_target} * 2 + R_{lifting} + R_{collision\_obstacle}$$

$$R_{distance\_gripper\_object} = 1 - min(Distance_{gripper\_object}/0.5, 1)$$

$$R_{distance\_object\_target} = 1 - min(Distance_{object\_target}/0.5, 1)$$

$$R_{lifting} = Height_{object\_curr} - Height_{object\_init}$$

$$R_{collision\_obstacle} = -0.5$$

The overall reward is composed of four parts: gripper-to-object distance, object-to-target distance for placing, lifting success, and collision avoidance. The distance reward works like this: if the gripper and object are within 0.5 meters, a reward is given; the closer they are, the higher the reward (up to 1). The lifting reward measures how much the object is raised above its initial table height, with a maximum reward of 1.2 for reaching the highest point. To encourage object placement, the distance reward to the target is scaled by 2. Lastly, collisions with 14 specified objects (listed in Table 4.2) detected by Collision Reporter in Isaac Sim lead to a penalty of 0.5 for each collision from the total reward.

/World/scene/food_on_table/cheeze_plate/cheese01
/World/scene/food_on_table/cheeze_plate/cheese02
/World/scene/food_on_table/cheeze_plate/cheese03
/World/scene/food_on_table/cheeze_plate/cheese04
/World/scene/food_on_table/cheeze_plate/cheese05
/World/scene/food_on_table/cheeze_plate/cheese06
/World/scene/food_on_table/cheeze_plate/cheese07
/World/scene/food_on_table/cheeze_plate/cheese08
/World/scene/food_on_table/cheeze_plate/fruit01
/World/scene/food_on_table/cheeze_plate/fruit02
/World/scene/food_on_table/cheeze_plate/fruit03
/World/scene/food_on_table/cheeze_plate/cuttingboard
/World/scene/food_on_table/glasses/glass_01
/World/scene/food_on_table/glasses/glass_03
/World/scene/food_on_table/table
/World/scene/food_on_table/table_01

Table 4.2: Collision Object Prim List

4.4.5 Human Gesture Mirroring

Human Initialization. At the beginning of each episode, a human is initialized at a random gesture, either pointing to the left or pointing to the right. The animation is extracted from a ballet hand gesture video in Omniverse Machinima App [8], see Figure 4.10.

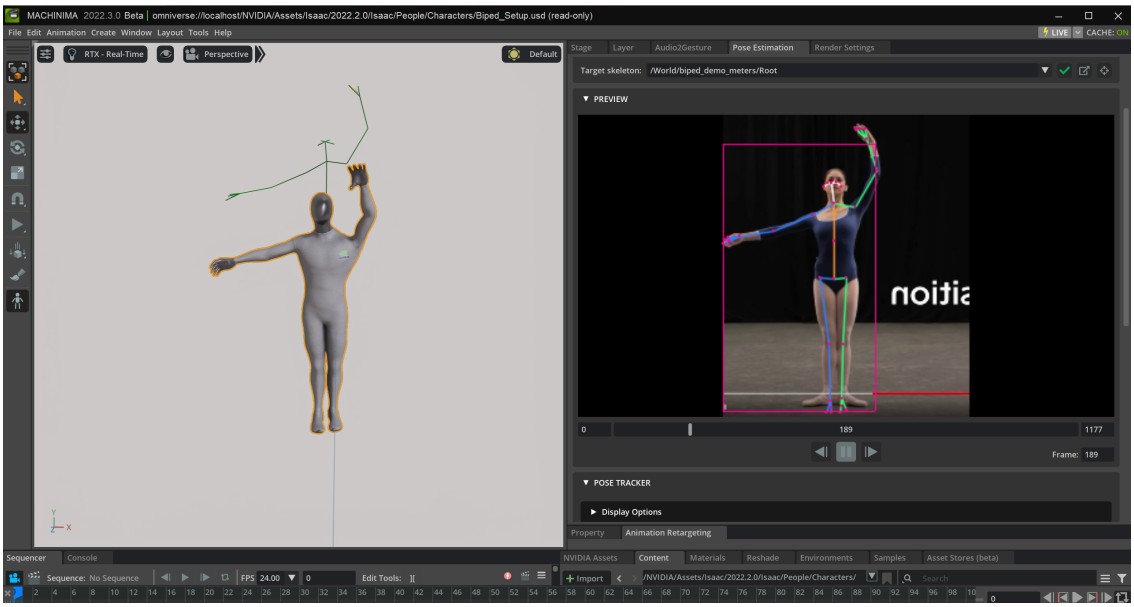

Figure 4.10: Synthetic Human Animation by Omniverse Machinima App

4.4.5.1 Robot Initialization

The Toyota HSR is initialzed at position [0,0,0], with a mounted RGB-D camera (see Figure 4.11 for its view). Both RGB-A data (in shape $256 * 256 * 4$) and Depth data (in shape $256 * 256$) is fed as the observation of of RL environment, a vector of length 327680.

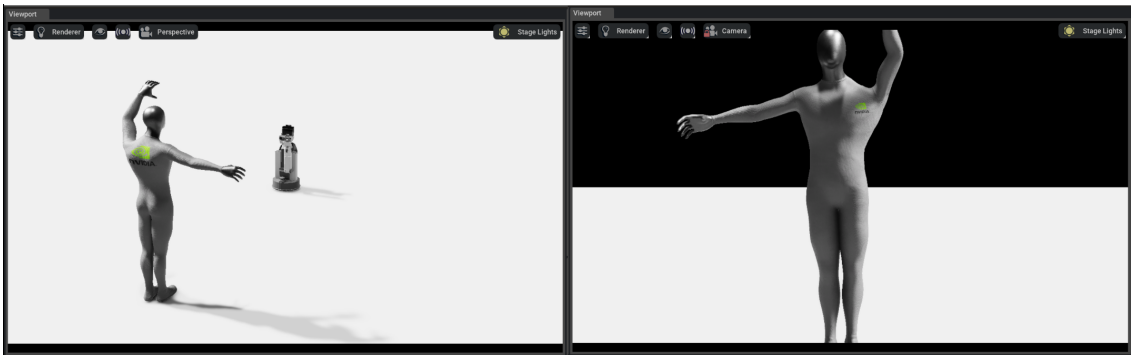

Figure 4.11: Perspective View and Robot Camera View

A code snipped of retrieving the RGBD data from Isaac Sim camera sensor.

```
from omni.isaac.sensor import Camera
import omni.replicator.core as rep
# Set the camera
self.head_camera = Camera(
prim_path="/World/hsr/head_rgbd_sensor_link/Camera")
self.head_camera.initialize()

# Attach the camera to replicator extension
render_product = rep.create.render_product(\
  "/World/hsr/head_rgbd_sensor_link/Camera", resolution=(256, 256))
self.rgb_wrist = rep.AnnotatorRegistry.get_annotator("rgb")
self.depth_wrist=\
  rep.AnnotatorRegistry.get_annotator("distance_to_camera")
self.rgb_wrist.attach([render_product])
self.depth_wrist.attach([render_product])

# Update the frame and get the data
rep.orchestrator.step()
self.rgba = self.rgb_wrist.get_data()
self.depth = self.depth_wrist.get_data()
```

4.4.5.2 Human Behavior

The human either points to the left or to the right. During the episode, the pointed direction won't change.

4.4.5.3 Robot Behavior

The HSR is set to use its wrist to point to either the left or the right, mimicking human gestures.

4.4.5.4 Robot Controller

Arm controller, position control.

4.4.5.5 Reward Functions

$$R_{total} = 1 \text{ if robot and human point to same direction else } 0$$

For each episode, if the robot points in the same direction as the human animation, then the reward will be 1, else 0.

Chapter 5

Experiments and Evaluation

5.1 Baseline Algorithms and Reference Metrics

Each benchmark is provided baseline algorithms evaluation result of algorithms PPO [56] and TD3 [34], provided by the stable-baselines3 library. These results serve as a reference for comparing the performance of users' own algorithms.

The metric used to gauge the success of the training process is the average reward achieved per episode, denoted as "rollout/ep_rew_mean". This metric reflects the mean cumulative reward obtained by an agent over the course of an episode during the training phase.

5.2 Hardware Configurations

The hardware configurations utilized for conducting the evaluations are presented in Table 5.1. It is noteworthy that all algorithms were trained on GPUs to leverage their parallel processing capabilities for efficient training.

```
CUDA Toolkit: 11.5, Driver: 12.0
Devices:
 "cpu"    | Intel64 Family 6 Model 186 Stepping 2, GenuineIntel
 "cuda:0" | NVIDIA GeForce RTX 4070 Laptop GPU (sm_89)

+-----------------------------------------------------------------------------+
| NVIDIA-SMI 528.76 Driver: 528.76 CUDA Version: 12.0                         |
|-------------------------------+----------------------+----------------------|
| GPU  Name  TCC/WDDM | Bus-Id        Disp.A | Volatile ECC    |
| Fan  Temp  Perf  Pwr | Memory-Usage   GPU-Util  Compute      |
|                      |                  |      MIG            |
|==============================+==================+======================|
|   0  NVIDIA GeForce | 00000000:01:00.0  On |          N/A    |
| N/A   60C    P0 48W | 7708MiB / 8188MiB | 76%       Default  |
|                      |                  |                N/A |
+-------------------------------+----------------------+----------------------+
```

Table 5.1: Algorithm Configuration

5.3 Algorithm Configurations and Performance Results

5.3.1 Navigation - Approaching a Randomized Human

With a stage frame rate of 24 fps, and considering Toyota HSR's maximum linear speed of 0.8 km/h and the initial positions of humans, each episode has been configured to last approximately 10 seconds, that is the maxinum espisode length is set at 1536, see Table 5.2. The PPO and TD3 configurations adhere to the standard settings provided by stabl-baselines3.

TD3 converges faster and achieves a better result.

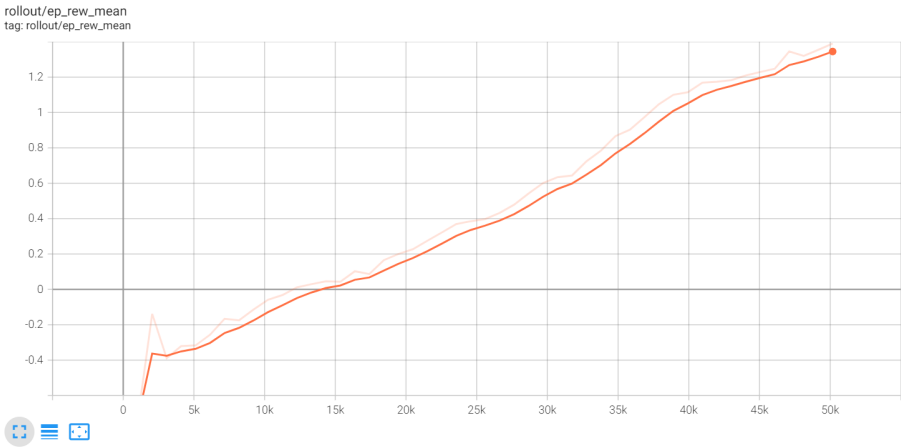

Figure 5.1: Navigation - Approaching a Randomized Human - PPO

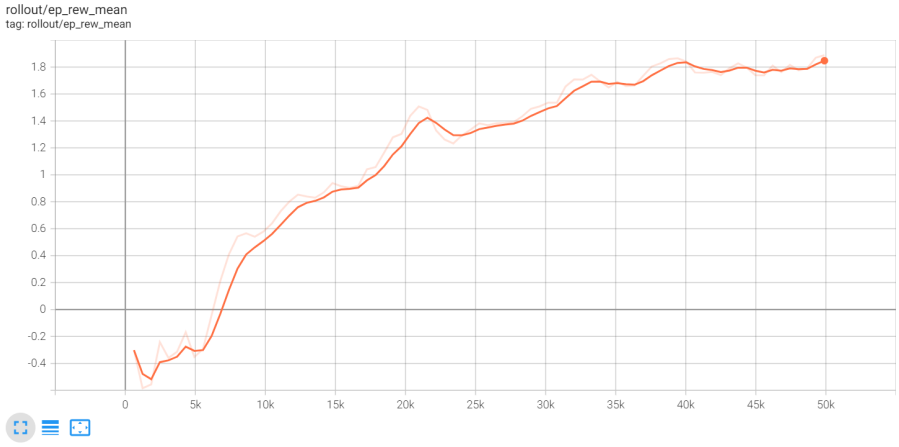

Figure 5.2: Navigation - Approaching a Randomized Human - TD3

```
total_timesteps = 50000
skip_frame=10,
physics_dt=1.0 / 60.0,
rendering_dt=1.0 / 60.0,
max_episode_length=256*6,

policy_kwargs_PPO = dict(activation_fn=th.nn.ReLU,
                         net_arch=[dict(vf=[128, 128, 128],
                         pi=[128, 128, 128])])
policy_kwargs_TD3 = dict(activation_fn=th.nn.ReLU,
                         net_arch=[128, 128, 128])


model = PPO(policy,
            my_env,
            policy_kwargs=policy_kwargs,
            verbose=1,
            n_steps=1024,
            batch_size=64,
            learning_rate=0.000125,
            gamma=0.9,
            ent_coef=7.5e-08,
            clip_range=0.3,
            n_epochs=5,
            gae_lambda=1.0,
            max_grad_norm=0.9,
            vf_coef=0.95,
            device="cuda:0",
            tensorboard_log=log_dir,
            )

model = TD3(MlpPolicy,
            my_env,
            policy_kwargs=policy_kwargs,
            learning_rate=0.000125,
            batch_size=64,
            buffer_size=1000000,
            learning_starts=5000,
            gradient_steps=5000,
            train_freq=1024,
            tensorboard_log=log_dir
            )
```

Table 5.2: Algorithm Configuration

5.3.2 Navigation - Following a Human

Algorithm configuration is exactly same as that of Chapter 5.3.1.

While PPO exhibits slightly faster early learning, TD3 catches up and matches its final performance.

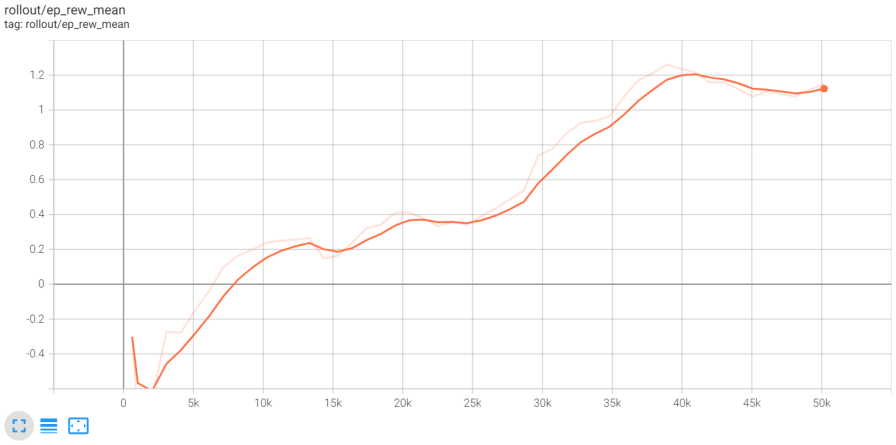

Figure 5.3: Navigation - Following a Human - PPO

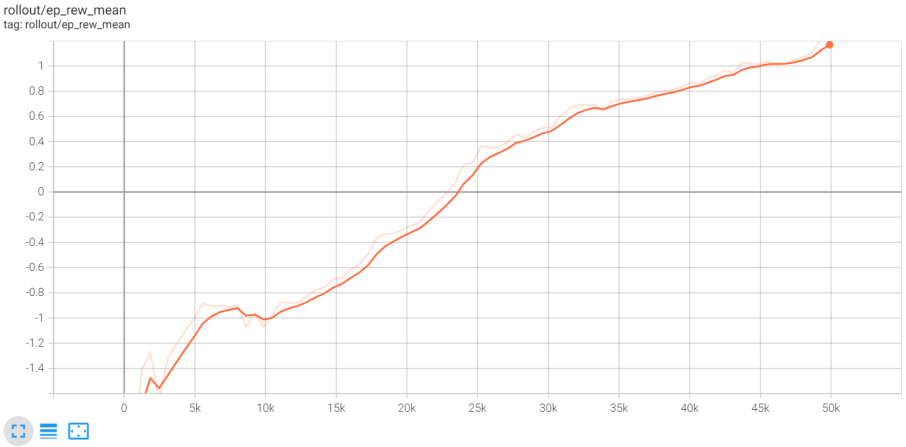

Figure 5.4: Navigation - Following a Human - TD3

5.3.3 Navigation - Approaching a Randomized Human with Obstacle Avoidance

Algorithm configuration is exactly same as that of Chapter 5.3.1.

PPO is providing better average rewards and greater robustness compared to TD3.

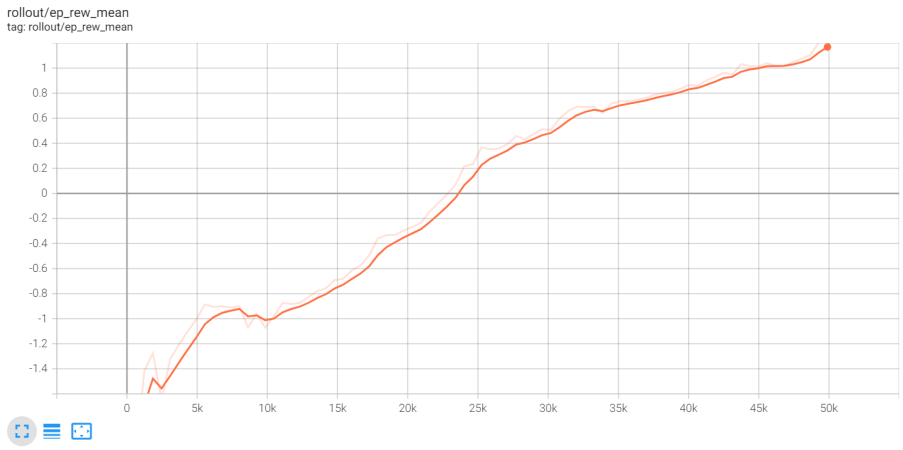

Figure 5.5: Navigation - Approaching a Randomized Human with Obstacle Avoidance - PPO

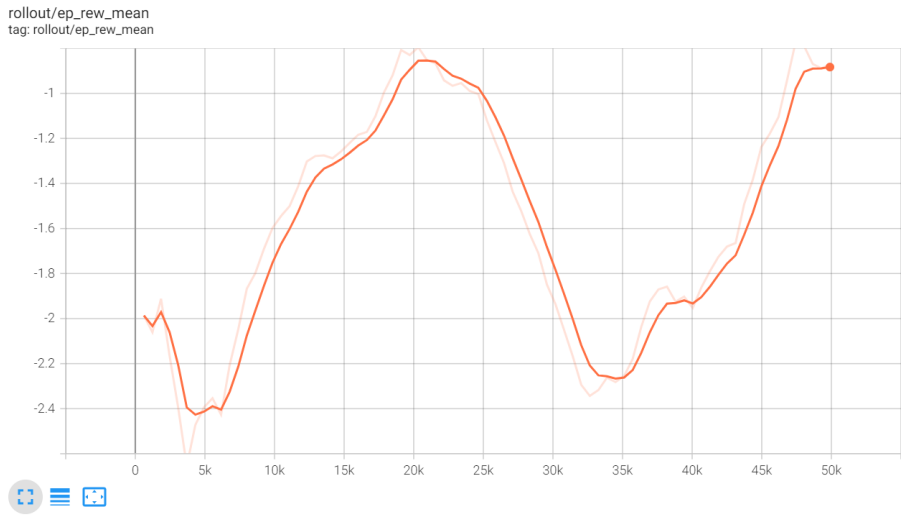

Figure 5.6: Navigation - Approaching a Randomized Human with Obstacle Avoidance - TD3

5.3.4 Object Manipulation - Pick and Place

Algorithm configuration for this task is very different, see Table 5.3. This is the complex nature of the manipulation task. This task involves multiple stages such as approaching the object, grasping, lifting, and shifting the object to the target. A gripper approaching an object takes much longer time than grasping, and these two actions can't be executed simultaneously. Thus one value from RL action space must be used to decide whether it is a pure gripper grasping movement, or a whole arm movement.

The parameter skip_frame (5.3) is set at a very large number 128. At each step, if it's a grasping action, only 8 frames will be consumed; otherwise, 120 frames will be allocated to provide enough time for the arm to reach a target position, which by practice takes around 100 frames time period to reach any pose.

Arm controller is subject to target position control, and it gives a reason to set the max_episode_length is at a very short frame interval of 256. Approaching the object takes 120 frames, grasping 8 frames, and lifting and reaching the target places takes another 120 frames. Thus under the target position control method, 256 is a perfect max episode length.

The batch size of PPO is set at one-quarter of the n_steps, a very large value compared to that of TD3. This may lead to a more stable gradient update. As shown in Figure 5.7 and 5.8, the PPO is much smoother than the TD3, although they have a similar average return of total reward.

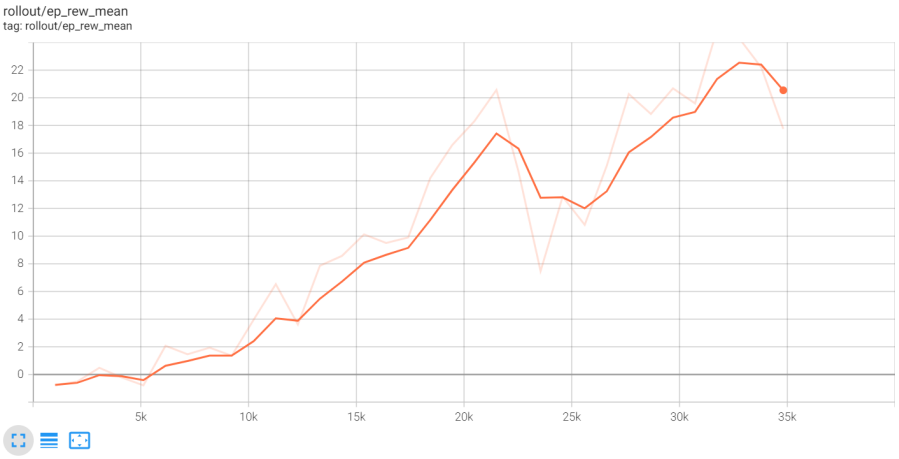

Figure 5.7: Object Manipulation - Pick and Place - PPO

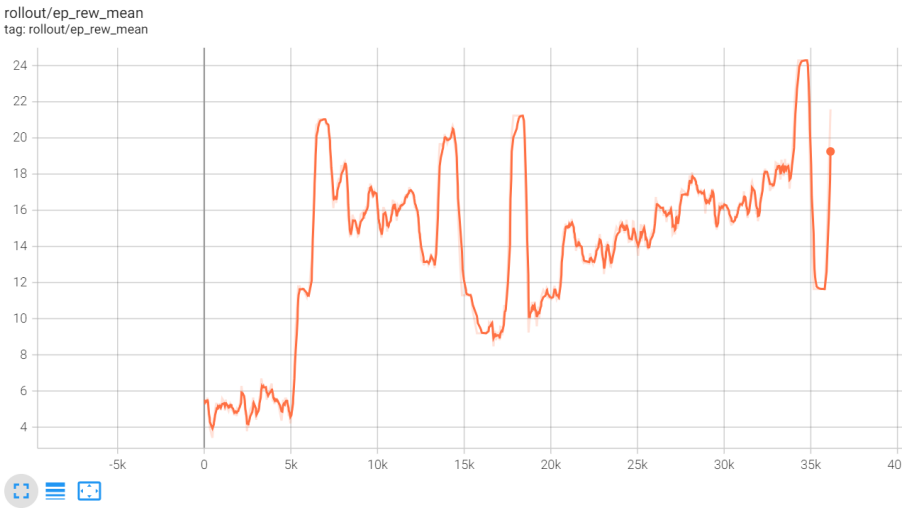

Figure 5.8: Object Manipulation - Pick and Place - TD3

```
total_timesteps = 35000
skip_frame=128,
physics_dt=1.0 / 60.0,
rendering_dt=1.0 / 60.0,
max_episode_length=128*2,

policy_kwargs_PPO = dict(activation_fn=th.nn.ReLU,
                            net_arch=[dict(vf=[128, 128, 128],
                            pi=[128, 128, 128])])
policy_kwargs_TD3 = dict(activation_fn=th.nn.ReLU,
                            net_arch=[128, 128, 128])


model = PPO(MlpPolicy,
            my_env,
            policy_kwargs=policy_kwargs,
            verbose=1,
            n_steps=128*4*2,
            batch_size=int(128*4*2/4),
            learning_rate=0.000125,
            gamma=0.9,
            ent_coef=0.01,
            clip_range=0.3,
            n_epochs=5,
            gae_lambda=0.95,
            max_grad_norm=0.9,
            vf_coef=0.95,
            device="cuda:0",
            tensorboard_log=log_dir,
)

model = TD3(MlpPolicy,
            my_env,
            policy_kwargs=policy_kwargs,
            learning_rate=0.000125,
            batch_size=64,
            buffer_size=1000000,
            learning_starts=5000,
            gradient_steps=5000,
            train_freq=1024,
            tensorboard_log=log_dir
            )
```

Table 5.3: Algorithm Configuration

5.3.5 Human Gesture Mirroring

Algorithm configuration for this task is very conservative. In each frame, a resolution of (256, 256) picture will be retrieved from RGB-D camera, and each pixel includes 5 channels, rgb-a and depth, as stated in Chapter 4.4.5. Thus each frame has a size of $256 * 256 * 5/1024/1024 = 0.3125$ MB, much larger than non-visual RL observation space. Thus a small batch size 32 is necessary to avoid GPU overflow.

Each episode lasts 256 frames, and the skip frame is 10, thus the HSR has around 25 times' chances in each episode to mirror its arm direction to where the human's. And each step's success will be rewarded 1 otherwise 0, as stated in chapter 4.4.5. Based on the above information, a robot should have at least half of 25, which is a 12.5 average reward in each episode. The PPO (see Figure 5.9) has shown HSR learns something from its camera data, but not that much, still outperforms TD3.

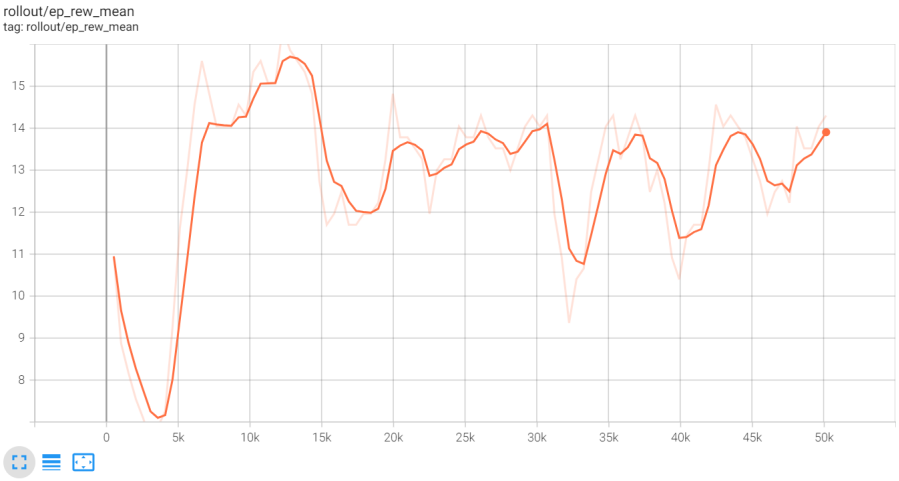

Figure 5.9: Human Gesture Mirroring - PPO

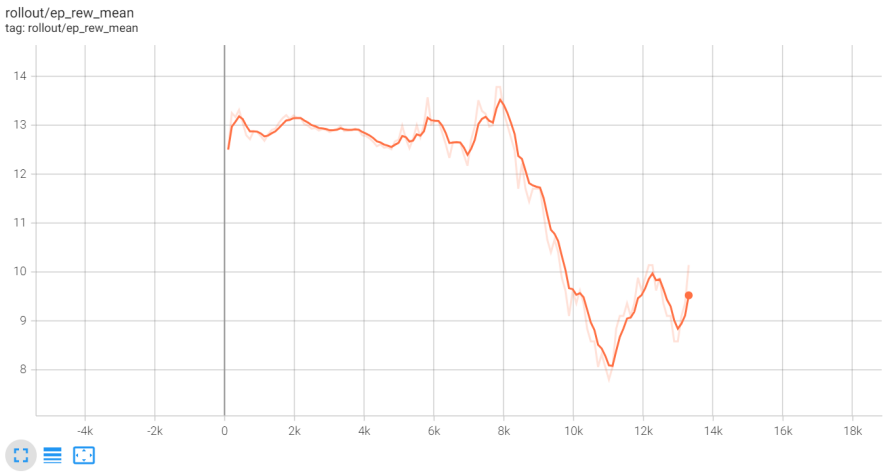

Figure 5.10: Human Gesture Mirroring - TD3

```
total_timesteps = 50000
skip_frame=10,
physics_dt=1.0 / 60.0,
rendering_dt=1.0 / 60.0,
max_episode_length=256,

policy_kwargs_PPO = dict(activation_fn=th.nn.ReLU,
                           net_arch=[dict(vf=[128, 128, 128],
                           pi=[128, 128, 128])])
policy_kwargs_TD3 = dict(activation_fn=th.nn.ReLU,
                           net_arch=[128, 128, 128])


model = PPO(MlpPolicy,
            my_env,
            policy_kwargs=policy_kwargs,
            verbose=1,
            n_steps=512,
            batch_size=32,
            learning_rate=0.000125,
            gamma=0.9,
            ent_coef=0.01,
            clip_range=0.3,
            n_epochs=5,
            gae_lambda=0.95,
            max_grad_norm=0.9,
            vf_coef=0.95,
            device="cuda:0",
            tensorboard_log=log_dir,
)

model = TD3(MlpPolicy,
            my_env,
            policy_kwargs=policy_kwargs,
            learning_rate=0.000125,
            batch_size=32,
            buffer_size=1000,
            learning_starts=5000,
            gradient_steps=5000,
            train_freq=512,
            tensorboard_log=log_dir
            )
```

Table 5.4: Algorithm Configuration

Chapter 6

Conclusion and Perspectives

This work explores advancing the social capabilities of the Toyota Human Support Robot (HSR) through the use of digital twin technology. The key contributions of this research are as follows:

- Offering an open-source USD model of the Toyota HSR, serving for both academia and industry usage.
- Creating a generalized Human-Robot Interaction benchmark to address the scarcity of this field, along with presenting baseline test results using popular reinforcement learning algorithms.
- Proposing solutions for integrating human animations in HRI simulators and benchmarks.
- Demonstrating a proof-of-concept of Digital Twin technology by training the robot using synthetic vision inputs.

While this research marks a significant stride forward, some limitations exist. Integrating ROS support, enhancing partial body contact modeling, and fully realizing real-time Digital Twin applications remain as future work.

In conclusion, this study marks an initial step toward leveraging digital twins for advancing household robotics. We hope the publicly available HSR model, environments, and results spur more research into high-fidelity simulation, powerful algorithms, and the fusion of digital twins, reinforcement learning, and robotics.

Bibliography

[1] Articulations. https://docs.omniverse.nvidia.com/py/isaacsim/index.html. (Cited on page 24.)

[2] Dynamic control. https://docs.omniverse.nvidia.com/py/isaacsim/index.html. (Cited on page 23.)

[3] Hsr meshes. https://github.com/ToyotaResearchInstitute/hsr_meshes. Accessed: 2023-08-12. (Cited on pages 4 and 21.)

[4] Hsr simulator. https://github.com/hsr-project/tmc_wrs_docker/tree/master. Accessed: 2023-08-12. (Cited on page 4.)

[5] Hsr urdf. https://github.com/ToyotaResearchInstitute/hsr_description. Accessed: 2023-08-12. (Cited on pages 4 and 21.)

[6] Isaac gym. https://developer.nvidia.com/isaac-gym. Accessed: 2023-08-12. (Cited on page 18.)

[7] Jetbot reinforcement learning with stable baselines3. https://docs.omniverse.nvidia.com/isaacsim/latest/advanced_tutorials/tutorial_advanced_rl_stable_baselines.html. Accessed: 2023-08-12. (Cited on page 19.)

[8] Machinima. https://www.nvidia.com/en-gb/omniverse/apps/machinima/. Accessed: 2023-08-12. (Cited on pages 19 and 33.)

[9] Nvidia digital twin. https://www.nvidia.com/en-gb/omniverse/solutions/digital-twins/. (Cited on page 8.)

[10] Nvidia isaac sim. https://developer.nvidia.com/isaac-sim. Accessed: 2023-08-12. (Cited on page 17.)

[11] Nvidia omniverse. https://www.nvidia.com/en-gb/omniverse/. Accessed: 2023-08-12. (Cited on page 16.)

[12] Nvidia omniverse. https://docs.omniverse.nvidia.com/nucleus/latest/index.html. Accessed: 2023-08-12. (Cited on pages 19, 27, 28 and 30.)

[13] Omniverse isaac gym. https://github.com/NVIDIA-Omniverse/OmniIsaacGymEnvs. Accessed: 2023-08-12. (Cited on page 18.)

[14] Openusd. https://openusd.org/release/index.html. (Cited on page 23.)

[15] Smpl. https://smpl.is.tue.mpg.de/. Accessed: 2023-08-12. (Cited on page 19.)

[16] Smplx. https://smpl-x.is.tue.mpg.de/. Accessed: 2023-08-12. (Cited on page 19.)

[17] Turbosquid. https://www.turbosquid.com/3d-models/shapespark-example-room-model-1964014. (Cited on page 30.)

[18] Abeyruwan, S. W., Graesser, L., D'Ambrosio, D. B., Singh, A., Shankar, A., Bewley, A., Jain, D., Choromanski, K. M., and Sanketi, P. R. i-sim2real: Reinforcement learning of robotic policies in tight human-robot interaction loops. In Conference on Robot Learning (2023), PMLR, pp. 212–224. (Cited on page 10.)

[19] Ahumada-Newhart, V., Kashyap, H. J., Hwu, T., Tian, Y., Mirzakhanian, L., Minton, M., Seader, S., Hedden, S., Moore, D., Krichmar, J. L., et al. Evaluation of the toyota human support robot (hsr) for social interaction and learning. International journal of technology, knowledge and society 19, 1 (2023), 21. (Cited on pages 14 and 16.)

[20] Aroor, A., Esptein, S. L., and Korpan, R. Mengeros: A crowd simulation tool for autonomous robot navigation. In 2017 AAAI Fall Symposium Series (2017). (Cited on page 8.)

[21] Biswas, A., Wang, A., Silvera, G., Steinfeld, A., and Admoni, H. Socnavbench: A grounded simulation testing framework for evaluating social navigation. ACM Transactions on Human-Robot Interaction (THRI) 11, 3 (2022), 1–24. (Cited on page 8.)

[22] Bonetto, E., Xu, C., and Ahmad, A. Grade: Generating realistic animated dynamic environments for robotics research, 2023. (Cited on page 8.)

[23] Casalino, A., Zanchettin, A. M., Piroddi, L., and Rocco, P. Optimal scheduling of human–robot collaborative assembly operations with time petri nets. IEEE Transactions on Automation Science and Engineering 18, 1 (2019), 70–84. (Cited on page 7.)

[24] Chao, Y.-W., Paxton, C., Xiang, Y., Yang, W., Sundaralingam, B., Chen, T., Murali, A., Cakmak, M., and Fox, D. Handoversim: A simulation framework and benchmark for human-to-robot object handovers. In 2022 International Conference on Robotics and Automation (ICRA) (2022), IEEE, pp. 6941–6947. (Cited on pages 9 and 11.)

[25] Cheng, Y., Sun, L., and Tomizuka, M. Human-aware robot task planning based on a hierarchical task model. IEEE Robotics and Automation Letters 6, 2 (2021), 1136–1143. (Cited on page 7.)

[26] Chibane, J., Engelmann, F., Tran, T. A., and Pons-Moll, G. Box2mask: Weakly supervised 3d semantic instance segmentation using bounding boxes, 2022. (Cited on page 10.)

[27] Christen, S., Yang, W., Pérez-D'Arpino, C., Hilliges, O., Fox, D., and Chao, Y.-W. Learning human-to-robot handovers from point clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2023), pp. 9654–9664. (Cited on page 11.)

[28] Clegg, A., Erickson, Z., Grady, P., Turk, G., Kemp, C. C., and Liu, C. K. Learning to collaborate from simulation for robot-assisted dressing. IEEE Robotics and Automation Letters 5, 2 (2020), 2746–2753. (Cited on page 9.)

[29] Deitke, M., Han, W., Herrasti, A., Kembhavi, A., Kolve, E., Mottaghi, R., Salvador, J., Schwenk, D., VanderBilt, E., Wallingford, M., et al. Robothor: An open simulation-to-real embodied ai platform. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (2020), pp. 3164–3174. (Cited on page 8.)

[30] Elbasheer, M., Longo, F., Mirabelli, G., Nicoletti, L., Padovano, A., and Solina, V. Shaping the role of the digital twins for human-robot dyad: Connotations, scenarios, and future perspectives. IET Collaborative Intelligent Manufacturing 5, 1 (2023), e12066. (Cited on page 10.)

[31] Erickson, Z., Gangaram, V., Kapusta, A., Liu, C. K., and Kemp, C. C. Assistive gym: A physics simulation framework for assistive robotics. In 2020 IEEE International Conference on Robotics and Automation (ICRA) (2020), IEEE, pp. 10169–10176. (Cited on page 9.)

[32] Fan, L., Zhu, Y., Zhu, J., Liu, Z., Zeng, O., Gupta, A., Creus-Costa, J., Savarese, S., and Fei-Fei, L. Surreal: Open-source reinforcement learning framework and robot manipulation benchmark. In Conference on Robot Learning (2018), PMLR, pp. 767–782. (Cited on page 9.)

[33] Fernández, O. H. C., Favier, P. T. S. A., Matellán, V., and Alami, R. Imhus: Intelligent multi-human simulator. (Cited on page 8.)

[34] Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. CoRR abs/1802.09477 (2018). (Cited on page 37.)

[35] Gao, X., Gong, R., Shu, T., Xie, X., Wang, S., and Zhu, S.-C. Vrkitchen: an interactive 3d virtual environment for task-oriented learning. arXiv preprint arXiv:1903.05757 (2019). (Cited on page 8.)

[36] Gloor, C. Pedsim: Pedestrian crowd simulation. URL http://pedsim. silmaril. org 5, 1 (2016). (Cited on page 8.)

[37] Huang, Z., Shen, Y., Li, J., Fey, M., and Brecher, C. A survey on ai-driven digital twins in industry 4.0: Smart manufacturing and advanced robotics. Sensors 21, 19 (2021), 6340. (Cited on page 10.)

[38] James, S., Ma, Z., Arrojo, D. R., and Davison, A. J. Rlbench: The robot learning benchmark & learning environment. CoRR abs/1909.12271 (2019). (Cited on page 9.)

[39] Kaur, P., Liu, Z., and Shi, W. Simulators for mobile social robots: State-of-the-art and challenges. In 2022 Fifth International Conference on Connected and Autonomous Driving (MetroCAD) (2022), IEEE, pp. 47–56. (Cited on page 10.)

[40] Kühnlenz, B., Erhart, M., Kainert, M., Wang, Z.-Q., Wilm, J., and Kühnlenz, K. Impact of trajectory profiles on user stress in close human-robot interaction. at-Automatisierungstechnik 66, 6 (2018), 483–491. (Cited on page 7.)

[41] Kumar, S., Savur, C., and Sahin, F. Survey of human–robot collaboration in industrial settings: Awareness, intelligence, and compliance. IEEE Transactions on Systems, Man, and Cybernetics: Systems 51, 1 (2020), 280–297. (Cited on page 10.)

[42] Lee, Y., Hu, E. S., and Lim, J. J. IKEA furniture assembly environment for long-horizon complex manipulation tasks. In IEEE International Conference on Robotics and Automation (ICRA) (2021). (Cited on page 9.)

[43] Li, C., Xia, F., Martín-Martín, R., Lingelbach, M., Srivastava, S., Shen, B., Vainio, K., Gokmen, C., Dharan, G., Jain, T., Kurenkov, A., Liu, C. K., Gweon, H., Wu, J., Fei-Fei, L., and Savarese, S. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. CoRR abs/2108.03272 (2021). (Cited on page 8.)

[44] Li, J., Bian, S., Xu, C., Chen, Z., Yang, L., and Lu, C. Hybrik-x: Hybrid analytical-neural inverse kinematics for whole-body mesh recovery. arXiv preprint arXiv:2304.05690 (2023). (Cited on pages 10 and 11.)

[45] Li, Z., Müller, T., Evans, A., Taylor, R. H., Unberath, M., Liu, M.-Y., and Lin, C.-H. Neuralangelo: High-fidelity neural surface reconstruction. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2023). (Cited on page 10.)

[46] Lin, X., Wang, Y., Olkin, J., and Held, D. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In Conference on Robot Learning (2020). (Cited on page 9.)

[47] Melesse, T. Y., Di Pasquale, V., and Riemma, S. Digital twin models in industrial operations: State-of-the-art and future research directions. IET Collaborative Intelligent Manufacturing 3, 1 (2021), 37–47. (Cited on page 3.)

[48] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. CoRR abs/2003.08934 (2020). (Cited on page 10.)

[49] Mirsky, R., Baraka, K., Faulkner, T., Hart, J., Yedidsion, H., and Xiao, X. Human-interactive robot learning (hirl). In 2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI) (United States, Sept. 2022), ACM/IEEE International Conference on Human-Robot Interaction, IEEE Computer Society, pp. 1278–1280. Publisher Copyright: © 2022 IEEE.; 17th Annual ACM/IEEE International Conference on Human-Robot Interaction, HRI 2022 ; Conference date: 07-03-2022 Through 10-03-2022. (Cited on pages 7 and 12.)

[50] Müller, T., Evans, A., Schied, C., and Keller, A. Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans. Graph. 41, 4 (July 2022), 102:1–102:15. (Cited on pages 10 and 11.)

[51] Myrvold, J. U. Digital twin of the kuka kmr iiwa: Developing a simulation framework using ros 2 and nvidia isaac sim. Master's thesis, NTNU, 2023. (Cited on page 8.)

[52] Pérez-Higueras, N., Otero, R., Caballero, F., and Merino, L. Hunavsim: A ros 2 human navigation simulator for benchmarking human-aware robot navigation. arXiv preprint arXiv:2305.01303 (2023). (Cited on page 8.)

[53] Puig, X., Ra, K., Boben, M., Li, J., Wang, T., Fidler, S., and Torralba, A. Virtualhome: Simulating household activities via programs. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2018), pp. 8494–8502. (Cited on page 8.)

[54] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A. Y., et al. Ros: an open-source robot operating system. In ICRA workshop on open source software (2009), vol. 3, Kobe, Japan, p. 5. (Cited on pages 4, 8, 10 and 13.)

[55] Robinson, N., Tidd, B., Campbell, D., Kulić, D., and Corke, P. Robotic vision for human-robot interaction and collaboration: A survey and systematic review. ACM Transactions on Human-Robot Interaction 12, 1 (2023), 1–66. (Cited on page 7.)

[56] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. CoRR abs/1707.06347 (2017). (Cited on page 37.)

[57] Semeraro, F., Griffiths, A., and Cangelosi, A. Human–robot collaboration and machine learning: A systematic review of recent research. Robotics and Computer-Integrated Manufacturing 79 (2023), 102432. (Cited on page 7.)

[58] Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. CoRR abs/1912.01734 (2019). (Cited on page 9.)

[59] Szot, A., Clegg, A., Undersander, E., Wijmans, E., Zhao, Y., Turner, J., Maestre, N., Mukadam, M., Chaplot, D., Maksymets, O., Gokaslan, A., Vondrus, V., Dharur, S., Meier, F., Galuba, W., Chang, A., Kira, Z., Koltun, V., Malik, J., Savva, M., and Batra, D. Habitat 2.0: Training home assistants to rearrange their habitat. In Advances in Neural Information Processing Systems (NeurIPS) (2021). (Cited on page 8.)

[60] Vuong, A. D., Nguyen, T. T., VU, M. N., Huang, B., Nguyen, D., Binh, H. T. T., Vo, T., and Nguyen, A. Habicrowd: A high performance simulator for crowd-aware visual navigation. arXiv preprint arXiv:2306.11377 (2023). (Cited on pages 9 and 11.)

[61] Wu, Y. Toyota hsr urdf model. https://github.com/jaswu51/Toyota_HSR_HumanSupportRobot_ROS1. Accessed: 2023-08-12. (Cited on page 21.)

[62] Xiang, F., Qin, Y., Mo, K., Xia, Y., Zhu, H., Liu, F., Liu, M., Jiang, H., Yuan, Y., Wang, H., Yi, L., Chang, A. X., Guibas, L. J., and Su, H. SAPIEN: A simulated part-based interactive environment. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020). (Cited on page 9.)

[63] Yamamoto, T., Terada, K., Ochiai, A., Saito, F., Asahara, Y., and Murase, K. Development of human support robot as the research platform of a domestic mobile manipulator. ROBOMECH journal 6, 1 (2019), 1–15. (Cited on pages 14 and 16.)

[64] Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. CoRR abs/1910.10897 (2019). (Cited on page 9.)