UNIVERSITY OF CALIFORNIA, MERCED

Modeling Collaborative Virtual Human Agents

A dissertation submitted in partial satisfaction of the requirements for the degree
Doctor of Philosophy

in

Electrical Engineering and Computer Science

by

Xiumin Shang

2023

Copyright Notice

Dedication

To my family.

Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgment

This thesis is a culmination of a long journey which would not have been possible without the tremendous amount of support I have received from a number of people. I would like to take this opportunity to extend my gratitude.

I am deeply thankful to my advisors, Prof. Marcelo Kallmann for providing me with the opportunity to work on this thesis and for always providing his guidance, direction and support since the beginning of my thesis work all the way to its completion and Prof. Angelo Kyrilov for accepting the role of being my advisor during a difficult and uncertain period in my doctoral journey. I would like to thank Prof. David Noelle and Prof. Ahmed Arif for serving on my defense committee and providing valuable feedback on my work. I am extremely thankful to Prof. Ioannis Karamouzas for providing insightful guidance and valuable inputs which helped me achieve meaningful improvements in my work. I would also like to thank my former advisor from the Department of Mechanical Engineering, Prof. Yanbao Ma, for supporting me through a difficult transition to a new country and a new academic environment.

I would also like to thank all the faculty members with whom I've had the opportunity to take courses and advance my knowledge, as well as my colleagues and labmates who have provided valuable discussions and support to my work. I've been extremely fortunate to have a lot of wonderful friends who have helped me make this journey memorable through difficult as well as happy times, and I'm sincerely thankful to all of them. I would like to thank Google for providing research credits on the Google Cloud Platform that helped me conduct experiments for the final part of my thesis.

To my husband Ruchir Travadi, who was not only patient and supported all aspects of my life, but also constantly provided valuable discussions and feedback on my work, I'd like to say during all these highs and lows throughout this journey, your calmness and encouragement always made me feel at ease and brought me courage to continue this journey. You continued to find ways to cheer me up with your humorous and positive attitude, and of course your cooking was so good that my stomach also felt equally at ease.

Finally, I would like to thank my family. Throughout my life they have been an eternal source of support, understanding, encouragement and above all love. They have worked hard to provide me with every opportunity including this journey and I am truly indebted to them for that.

Publications

9. **Xiumin Shang**, and Marcelo Kallmann. Learning Robust Agent under Multi-Agent Framework. (To be submitted.)

8. **Xiumin Shang**, Tengyu Xu, Ioannis Karamouza and Marcelo Kallmann. Constraint-Based Multi-Agent Reinforcement Learning for Collaborative Tasks. *Computer Animation and Virtual Worlds, 2023*

7. Pei Xu, **Xiumin Shang**, Victor Zordan and Ioannis Karamouzas. Composite Motion Learning with Task Control. *ACM Transactions on Graphics, 2023*

6. **Xiumin Shang**, Marcelo Kallmann. Chapter 9: Collaborative Virtual Trainers in VR Applications. *Springer, Cham, 2021, https://doi.org/10.1007/978-3-030-79062-2.*

5. **Xiumin Shang**, Ruisheng Diao. Multi-stage Transmission Line Flow Control Using Centralized and Decentralized Reinforcement Learning Agents. *NeurIPS 2020 workshop on ML4Eng*

4. **Xiumin Shang**, Ruisheng Diao. Reinforcement Learning Based Solution to Power Grid Planning and Operation Under Uncertainties. *SC 2020 workshop on AI4S*

3. **Xiumin Shang**, Marcelo Kallmann, Ahmed Sabbir Arif. Designing Behaviors for Autonomous Virtual Trainers: The Effects of Correctness and Suggestive Feedback. *Preprint at https://arxiv.org/abs/1811.02693.*

2. **Xiumin Shang**, Marcelo Kallmann, Ahmed Sabbir Arif. Effects of Correctness and Suggestive Feedback on Learning with an Autonomous Virtual Trainer. *In Proceedings of the 24th International Conference on Intelligent User Interfaces Companion (IUI 2019 Companion). ACM, New York, NY, USA.*

1. **Xiumin Shang**, Marcelo Kallmann, Ahmed Sabbir Arif. Effects of Virtual Agent Gender on User Performance and Preference in a VR Training Program. *In Proceedings of the 2019 Future of Information and Communication Conference (FICC 2019). Springer-Verlag New York, Inc., New York, NY, USA.*

Abstract

Autonomous virtual agents have been employed in different areas, spanning applications from education and training to gaming and e-commerce. In particular, agents of human-like appearance, or virtual human agents, have been greatly improved over the years with the advancement of technologies in machine learning, natural language processing and computer graphics. This dissertation addresses topics related to the design and training of virtual human agents for collaborative tasks.

On the design front, this thesis presents user studies investigating the effect of agent gender and feedback strategies on instructional object manipulation tasks. The obtained findings show that, although agent gender has no significant effect on user preference or performance, users find female agents to be more attractive. Comparing suggestive feedback and correctness feedback strategies, it is found that correctness feedback is preferred by the users and leads to a 65% shorter task completion time.

On the training front, the focus is on collaborative tasks controlled by Deep Reinforcement Learning (DRL). Three important challenges are addressed: sequential multi-phase tasks, collaborative learning of motor tasks in a physics-based environment, and addressing realism and robustness. First, to train agents for collaborative multi-phase tasks the Constrained Multi-agent Soft Actor Critic (C-MSAC) approach is proposed. It is shown that this approach achieves better mean episode reward, generalizability and robustness to disturbance when compared with an unconstrained multi-agent learning baseline. Second, the original inverse kinematics approach is replaced with physics-based control which leads to more natural movements and improves robustness against variations to physical parameters of the environment.

Finally, the Constrained Robust Soft Actor Critic (C-RSAC) approach is proposed to train a single robust agent using the policies obtained from C-MSAC for initialization. C-RSAC uses noise augmentation in the action space of one of the agents to improve robustness of the collaborating agent. It is shown that C-RSAC leads to an improved mean episode reward compared to C-MSAC when collaborating with a noisy agent.

In summary, this thesis investigates several important aspects related to autonomous collaborative virtual human agents such as gender, appearance, feedback strategies, collaborative training, physics-based animation and robustness. The proposed C-MSAC approach for multi-phase multi-agent training and C-RSAC approach for multi-phase single agent robust training represent new contributions to the emerging area of autonomous virtual trainers. Overall the contributions from this thesis inform the design and modeling of collaborative virtual human agents, furthering the goal of enabling these agents to assist humans on various applications of virtual trainers.

Chapter 1

Introduction

A virtual agent is a computer program that provides assistance or guidance for humans to complete tasks. Virtual agents have been widely used in many fields such as gaming, robotics, training, e-commerce, etc. Examples include a virtual voice agent or chatbot (1) that can understand human voice commands and give appropriate human-like responses as well as a virtual human-like agent or a virtual character that can interact with humans to give visible instructions for tasks that need body movement or provide an immersive experience while interacting with them (112; 46). More recently, there has been a heightened interest in the areas of virtual as well as augmented reality. Virtual agents can play a crucial role in many important applications that leverage the capabilities of these platforms.

In this thesis, the focus is on modeling virtual agents with human-like non-verbal behaviors that can visually interact with humans to facilitate the completion of various manipulation tasks. Agents with a human-like appearance and behavior are crucial in many applications. For instance, in many games such as sports, automated human-like agents could be used to simulate players that are not human controlled. Having a human-like appearance could also help make the interaction seem more natural and familiar in applications involving an educational setting. While the ability to deploy automated human-like agents is very powerful, it is also very challenging to design and train such agents. The likeness to a human form leads to a very open design space with endless possibilities for appearance and behavior. In addition, in order to make the interaction with a human-like agent immersive, the agent needs precise control to achieve human-like motion while also making intelligent decisions to collaborate. Being able to do this accurately is a very challenging problem.

Building an immersive virtual environment requires making appropriate choices for a number of design variables including the agent's appearance as well as behavior of interaction. Appearance includes facial expression, body gestures, gender, and age among many other factors and it has been shown that appearance has an effect on a number of variables such as motivation, learning outcomes, persuasive effects, and the agent's overall evaluation (19; 55; 113).

User experience and outcomes are shaped not only by the agent's appearance, but also by the flow of interaction. So it is important to design the agent's behavior such that it can provide effective communication using appropriate interaction strategies.

Once the appropriate design choices have been made for the environment and the agent, the next step is to program the agent for execution of the required actions in the environment. One possibility is to use a hard-coded solution involving a step by step procedure for the virtual agent to follow. However, in such cases the virtual agent will not be able to adapt and execute a similar task in the same or different environment (137; 66). This is therefore an expensive and a task specific solution. The goal is to enable the agent to generalize in different environments and tasks. Different machine learning methods have been applied in order to increase the adaptability of virtual agents. A common approach is to apply reinforcement learning in order to improve the virtual agent's sequential decision making policy by interacting with the environment periodically (69). Previous work has demonstrated the effectiveness of using Deep Reinforcement Learning (DRL) for virtual and robotic agents (138; 101). For example, in the StarCraft II game environment, reinforcement learning agents have been rated at a grandmaster level and above 99.8 % of official human players(135).

The goals of this thesis are twofold:

- *To analyze the effects of agent design variables*: in order to determine the effect of agent design variable choices on user preference as well as performance on problem solving tasks.

- *To develop training methods enabling agents to collaborate in solving complex tasks*: in order to train agents that are robust to environmental noise and disturbances, and that avoid the need to collect expensive human interaction data.

1.1 Effects of Agent Design Variables

The space of design variables related to virtual human-like agents is very large. In this thesis, the focus is on two main directions: gender and feedback delivery. For both of these, object manipulation tasks are developed where humans can interact with the agent and user studies are performed while varying the relevant agent characteristics.

For studying the impact of agent gender, a Box and Blocks Test (BBT) is used where the agent instructs the user to quickly move specific color cubes from one side of the table to the other. User performance is then evaluated in terms of task completion time as well as error rate and users are also queried for their preferences.

For studying agent feedback delivery mechanisms, a training task is formulated where the role of the agent is to help the user sort a set of cards or cubes representing some information according to a specified sorting criterion. The agent provided one of two types of feedback: correctness feedback or suggestive feedback. For correctness feedback, when a wrong sorting was presented the agent would present the correct answer and let the user study it before continuing. For suggestive feedback, the agent would point out each pair of incorrectly ordered cards. Sorting scores and average execution times are then evaluated to compare performance. In the post study questionnaire, users were also asked for their preferences in terms of agent feedback strategy.

1.2 Training Agents for Collaborative Tasks

The focus of this thesis is on reinforcement learning techniques to train agents to collaborate on complex tasks. There are three main challenges associated with this task:

- *Reward design*: In a typical scenario, the completion of the task would be associated with a reward function in a reinforcement learning framework. However, complex tasks requiring collaboration often have distinct phases that have different goals or requirements for the agent. For such tasks it becomes important to address the multi-phase nature of the reward appropriately during the training process.

- *Natural animation*: Although optimizing the reward function leads to a higher task completion rate for the agent, it does not always necessarily lead to natural looking movements. It is important to develop the environment and the training process in such a way that the agent does not exhibit unnatural or physically impossible movements.

- *Data requirements*: In theory it is possible to train a single agent to directly collaborate with a human in a virtual environment. However, directly training such an agent would require a huge amount of human interaction data at a high frame rate. Furthermore, generic human data may not be applicable when targeting interactions with motion-impaired users. Instead, the focus is on developing training methods that avoid such difficulties related to data collection.

In order to address the problem of multi-phase tasks, the Constrained Multi-Agent Soft Actor Critic (C-MSAC) approach (123) is proposed motivated by the safe reinforcement learning approach (145). The original idea behind the safe RL concept was to define task constraints for the purpose of safety and stability. The reinforcement learning problem is formulated as a constrained optimization problem of maximizing the reward subject to a specified set of safety constraints. However, constraints can also be used to model different sub-objectives in a sequential task. All sub-objectives except the final objective are treated as safe constraints during learning, and the final objective is optimized only if none of the constraints have been violated. The proposed framework can be used to train multiple agents to collaboratively finish complex tasks with sequential objectives.

When the C-MSAC approach is used in an environment where the agent movements are controlled by inverse kinematics, the results show that the agent is able to achieve a high mean average reward on the provided task. However, simulation videos show that the agent sometimes exhibits unnatural oscillatory movements. Using C-MSAC with physics-based full arm joint-level control for the agents is proposed to mitigate this issue. Since the environment enforces the laws of physics, it prevents movements that are physically impossible. It is demonstrated that learning under physics also improves the ability of the agents to react to changes in the physical properties of the environment such as the mass of the ball.

To avoid the collection of expensive human interaction data, a new robust learning approach, Constrained Robust Soft Actor Critic(C-RSAC) is proposed, based on C-MSAC, that can be used to train a robust agent from a multi-agent framework for collaborative tasks. One shortcoming of C-MSAC training is that the agents might learn to optimize against each others' specific behaviors and therefore might not generalize well enough to directly collaborate with other agents or humans. This problem is addressed by improving agent robustness. In the C-RSAC framework the pre-trained policies obtained from C-MSAC are used to a noisy agent and initialization of a robust agent. The parameters of the noisy agent policy are frozen and noise augmentation is performed in the action space. These noisy actions can be used to simulate potential errors by a human subject. The policy for the robust agent is initialized from the pre-trained C-MSAC policy parameters and updated to maximize the expected reward. The results show that the proposed C-RSAC approach improves the agent robustness.

A tray balancing task is used to study the proposed approaches. The task involves two agents holding a tray with a ball moving freely on top of it. The goal of the agents is to lift the tray and control it precisely such that the ball can follow a specified trajectory. A physically simulated virtual environment is developed to simulate this task. The overall development process is divided into three main pieces of work:

- *Validating Constrained Multi-Agent Reinforcement Learning with Inverse Kinematics*: In this thesis the C-MSAC approach is proposed with a focus on training agents with constrained multi-agent reinforcement learning. There are two phases: appropriately lifting the tray and following the trajectory. The task of appropriately lifting the tray is specified as a constraint and following the trajectory is specified as the reward objective. To simplify the complexities of the task, Inverse Kinematics is used to control the motion of the arms. Agent's hands are assumed to always be attached to fixed anchor points on the tray. It is shown that compared to a baseline multi-agent learning algorithm, C-MSAC leads to a better mean average reward performance, better generalizability to unseen trajectories and higher robustness to environmental disturbance.

- *Physics-based Multi-Agent Reinforcement Learning*: Learning full arm control under physics using C-MSAC is proposed in this work. An initial additional phase is introduced for the arms to reach the anchor points on the tray and the reward function is designed to incentivize appropriate grabbing behavior. It is shown that compared to inverse kinematics, learning under physics leads to more natural movements and also improves robustness to variations in the physical properties of the environment such as changes to the ball mass.

- *Robust Agent Learning*: The ability to control full arms under physics enables their manipulation by adding noise for simulating errors. In this work, the C-RSAC approach is proposed. Pre-trained policies from C-MSAC are used for the noisy agent and noise augmentation is performed in the action space which results in disturbance to the handling of the tray. The other agent's policy from C-MSAC is used to initialize a robust agent which is then trained to maximize the reward while trying to mitigate the impact of disturbance from such noise. It is shown that the proposed approach leads to more robust control of the tray. This is important for applications of virtual trainers interacting with avatars controlled by real users.

1.3 Outline

This thesis is organized as follows:

- Chapter 2 provides a literature review of topics related to this thesis;
- Chapter 3 describes the user study conducted to investigate the effect of virtual agent's gender on user's task performance and preference;
- Chapter 4 explains the user study conducted to evaluate suggestive feedback and correctness feedback strategies with a virtual agent;
- Chapter 5 describes the construction of a multi-agent collaboration learning framework, and explains the proposed C-MSAC approach;
- Chapter 6 describes the extension of the C-MSAC approach to physics-based control and the proposed C-RSAC approach for robust learning;
- Chapter 7 summarizes the main findings from this thesis and provides potential future work directions.

Chapter 2

Literature Review

This chapter reviews prior work in the literature relating to the problem of developing human-like virtual agents. We start with a review of studies related to agent motion and character animation. That is followed by a review of learning strategies such as reinforcement learning, including multi-agent and multi-objective learning. Finally, we review studies related to human agent interactions.

2.1 Virtual Agents

In this thesis a virtual agent refers to a simulated human-like character that can collaborate with humans in order to complete a given task with the use of interactive verbal and/or non-verbal movements and behaviors. Virtual agents within a virtual training environment can simulate human behavior with recent libraries providing diverse sets of behavior characteristics (48; 56).

2.1.1 Agent Motion and Character Animation

One approach for obtaining realistic human-like motion is programming agents by direct demonstration using data recorded from actors or experts (35). For example, a powerful approach has been presented for learning from the recorded motion data, which includes a phase variable added to complete the virtual character behavior transition between different neuron networks trained by expert motion data for human characters (60) and as well animal characters (154).

Physics-based character animation refers to animating human dynamics under physics. It involves characters that are modeled as connected rigid bodies, and controlled by the joint torques or muscle models. While it is easy for humans to perform many different skills, articulating such control strategies as a virtual character remains a challenging problem. Although we can constrain the character model to behave realistically in terms of the properties of physics, that alone does not guarantee a realistic behavior in terms of the overall obtained motion.

Different approaches have been proposed to integrate physics with additional control, learning or optimization to produce natural motions. Some early works use hand-crafted feedback strategies, and build the control model with or without motion captured data in order to model the character dynamics. These include using random sampling approaches and forward dynamics to reconstruct or optimize a full body character controller from a group of captured motion data (84; 52; 5), optimizing trajectory planning with segmented motion data (102), using a finite state machine controller to calculate the force distance at individual joint (149), and optimizing the Proportional-Derivative (PD) controller in linear time to track the reference motions (151). These control approaches can be effective in performing different skills, but the control model relies heavily on the choice of constraints and usually needs a large motion data set covering broad animations in order to adapt to new skills.

Recently, more researchers have taken advantage of the development of deep learning, and are using reinforcement learning to model nonlinear controllers for animation tasks, as it needs minimal task-specific control signals to produce the expected behavior. These include learning locomotion behaviors (105), learning full-body behaviors from examples (104; 143), learning to play basketball (83), learning athletic jumping strategies (150), learning locomotion stepping stone skills (142) and learning climbing wall skills (94).

Animation-related control objectives are usually integrated as the reward function in a reinforcement learning framework, and designing a proper reward function is critical for achieving complex animation tasks. For example, in a biped walking on stones task (142), the reward function was decomposed into three different objectives: hitting the target, moving forward, and maintaining a balanced motion. Each objective needs a carefully designed reward function in order to train a good walking model. But for full-body character control, an objective-focused reward design may not be enough to guarantee realistic motions. Different strategies have been proposed, like integrating expert motions as reference motions in the reward design in order to train for learning complex skills (104; 83), using generative adversarial networks and expert data to iteratively improve the generative controller (143), utilizing phase functions to transition between motions (60; 154), constraining actions into a subspace of natural poses (150), curriculum learning to modify the task difficulty level (142; 150), and hierarchical learning to use multilevel control (105).

Currently, most character animation tasks focus on single character control, and they mostly need to use or learn from motion data in order to get a natural looking motion. Learning directly from motion data for a single character animation is possible, but for a multi-character environment it is difficult as multiple human players have to be involved in the procedure. So for multi-character simulations, character behavior could either be learned separately, and then transferred to a multi-character framework for advanced learning (139), or could be directly used in a multi-character environment as in crowd simulation (57). Our work approaches multi-character collaborative tasks by learning a control policy for each character's upper body movement at the joint level.

2.2 Learning Approaches

2.2.1 Reinforcement Learning

The basic idea behind many reinforcement learning algorithms is to estimate the action value function using the Bellman equation as an iterative update. Such value iteration algorithms converge to the optimal action-value function. In practice, this basic approach is impractical because the action-value function is estimated separately for each sequence without any generalization. Instead, it is common to use a function approximator to estimate the action value function. In the RL community this is typically a linear function approximator, but sometimes a non-linear function approximator such as a neural network is used.

2.2.2 Multi-agent Reinforcement Learning

There are many situations in real-world problems where a single RL agent would not be able to cope. In such situations, the application of a multi-agent System (MAS) approach is indispensable. The multi-agent learning problem is about multiple agents interacting with each other in order to solve cooperative or competitive tasks together. It has a long history in machine learning, and is much more challenging than single agent problems because of issues like agent scalability, non-stationarity of the environment and non-unique learning objectives. A few survey papers offer good resources for further information (157; 59; 97). Following the success of applying RL for single-agent learning, more and more researchers have become interested in studying RL in multi-agent environments.

Building a single agent RL environment could be easy, and there are also rich existing environments for researchers to experiment with different algorithms such as OpenAI Gym (26) with 2D and 3D environments, ML-Agents (67) with 3D environments, from both simple object control to complex human character control, Atari games (21) with real game environment and learning from pixels. For multi-agent problems, building the experiment environment could be more time consuming and challenging, as interaction strategies among agents and their environment are often more complex, and the reward design has to not only consider task objectives but also interaction strategies between agents. Some environments have been developed like the multi-particle environment (85) which is simplified to a 2D environment, a football player environment (76), and a hide-and-seek environment (15), with a focus more on the control and communication strategy between agents. The latter two works use simplified objects or animated characters as agents, and focus on exploring some emergent strategies during learning. There is a lack of environments available to experiment and perform human-like behaviors at the joint-level control. Recently, there is a proposed work (139) which created such an environment to train human-like characters to compete against each other in sports games like fencing and boxing. All these environments are more task focused, and the environment developed in this thesis expands the task types to include object manipulation tasks with human upper body movements. Full body movement could easily be included as future work.

When solving multi-agent control problems, it is natural to think of using single agent RL directly in a multi-agent environment, where all agents learn a shared policy and execute in a joint action space together, or make a direct extension of single agent RL solution where each agent learns independently by considering other agents as part of the environment, such as independent Q-learning (129). The above approaches are straightforward to implement, however will cause non-scalibility or non-stationary difficulties. Lowe et al. (85) proposed a parameter sharing approach to tackle this problem known as the Centralized Training and Decentralized Execution approach (CTDE). This is extended from the actor-critic framework, and each agent uses a centralized critic to access all agents observations and actions parameter, so that it can learn an approximate model of the online policies of other agents within a stationary environment. The learned policy only uses local information, so it can be used by each agent without the need for further communication with other agents. Thereafter, many other algorithms have been developed based on this framework by addressing different difficulties of multiagent systems, like credit assignment (39), scalability (148), value decomposition (128) and an attention mechanism (64) for faster and stable learning (110).

In the proposed environment, the focus is on two human-like agents cooperatively completing object manipulation tasks, and we choose the State of The Art (SOTA) CTDE framework combined with the single agent RL approach SAC (54), because of its ease of implementation and also guaranteed state of the art performance in continuous action spaces.

2.2.3 Multi-objective Learning

Multi-objective learning involves tasks that have two or more objectives to be optimized. It has the lifelong learning property (29), which means an agent can be trained on a sequence of relatively easy tasks to gain experience and develop a more informative measure of reward, which can then be leveraged when performing harder tasks.

It is common to break down complex tasks into subtasks based on their sub-objectives. When designing the objectives of a given task, the sub-objectives can be defined separately without a connection but can share the same action spaces. Examples include a robot arm picking and placing objects in different boxes (62), agents working together to push different objects into different locations (147), humanoid characters learning different skills from motion data and performing them based on user control signals (104). The sub-objectives can also be defined as sequential objectives, such that in order to finish the final objective, the previous objectives have to be completed first, this is also aligned with our task design. Examples of some tasks solved in this manner are: an agent moving to a target location while moving objects or obstacles along the moving path (93); a biped walking character walking on the ground while maintaining a natural gait motion with a walking target (105), playing basketball which has similar objectives where the full body character needs to play basketball while maintaining the walking motion (77); upper body animated characters wearing cloth (32) where it needs to reach, grasp and put the cloth on each arm, and an upper-level controller is necessary to instruct the motion sequence; a four legged robot walking to a target location where the trajectory to the target provides the reward signal for the robot to learn its movement during each time step (92).

Here the focus is on RL-related approaches where the objectives are generally interpreted as reward signals for training. In the reward design, each objective has a separate reward term which could be linearly added with a different weight to adjust the importance of each subobjective in the optimization function (32). The reward terms could also be optimized with separate RL models, and then a high-level controller could be used to choose which lower-level controller to use at each time step (18). The above mentioned work also constructs an upper-level controller combined with all low-level controllers to perform different tasks. Another work on wearing clothes also constructs a sequential model controller to choose the low-level RL controller to decide what action to perform to wear a cloth (32). For sequential objectives, hierarchical RL can be considered (18).

The reward term can still be optimized as separate RL models with different action spaces. Since the upper-level RL model is responsible for the final objective or planning, and its output actions will be used as the training target for a lower-level controller, while the lowest level RL controller outputs the actions to perform the task, it is usually trained in an end-to-end manner. However this can be time consuming, so a pretrained lower-level controller can be adopted. For example, in a biped or four-legged walking task, one can train the character to perform a simple stand up posture. Most existing works focus on optimizing multiobjective learning for a single robot or virtual character. In our work, we extend it to a multiagent multi-task setup, where two virtual characters need to control the force applied on each side of a tray to solve a balancing task.

2.3 Human Agent Interactions

The interaction behavior between agent and human collaborations has been studied, like using Colored Trails, an environment designed for evaluating decision making behaviors between players (134). It has been found that humans prefer to collaborate with more cooperative agents instead of egoistic agents. Different experiments have been conducted to study the human-agent interaction behaviors to increase collaboration performance, like to improve human awareness of agents during collaboration (28) and to improve the human trust in agents (132).

A number of other types of agents have also been designed for collaboration with humans, for instance in simplified 2D environments; however, not much immersive interactive collaboration can be experienced in such cases (111; 6). Development frameworks are currently available that can be used for building applications employing virtual trainers, for example: the Virtual Agent Interaction Framework (48), the Virtual Human Toolkit (56), and as well the Unity engine. These systems facilitate the implementation of virtual trainers and they also provide a rich collection of features to be used for developing research in the area.

In recent years, techniques for implementing direct collaboration with virtual trainers have been significantly implemented in the robotics field, including robotic trainers to help humans during object manipulation and assembling tasks in a physical environment (31; 116). Wang(137) has proposed a human-robot assembly system that targets production industry to save human labor and augment production by creating safe and reliable robot agents to collaborate with humans in the same environment. In these cases a robot can assist humans to move objects from a source place to a target place, or to assemble structures efficiently.

In this thesis the focus is on agents interacting on a collaborative task requiring fine movements in order to be completed. The presented scenarios have not been addressed before in the literature.

Chapter 3

Effects of Virtual Agent Gender

3.1 Introduction

Because virtual reality (VR) can give people a sense of presence in virtual worlds, it is becoming popular in various fields, including entertainment, education, engineering training, and rehabilitation. Virtual human-like characters are common in virtual worlds, which enable social interaction with other users or a computer system. Two different types of characters are commonly used, virtual avatars and virtual agents. An avatar is a virtual character that is controlled by an actual user (81). Avatars are very popular in video games, including social platforms such as High Fidelity (Fidelity), AltspaceVR (63), and Sansar (118). Agents, on the other hand, are virtual characters that are controlled by computer programs (81). 2D and 3D virtual agents have already been a powerful strategy in online enterprises, such as in advertising (23), shopping (7), education (79), and persuasion (153). However, not many studies have investigated virtual agents in the context of Virtual Reality Environments (VRE). This work explores social interactions among users and virtual agents in a virtual reality training scenario, where users enter the virtual world through a Head-Mounted Display (HMD). Particularly, it investigates whether virtual agent gender affects user performance and preference in a training program, where they have to take instructions from a virtual agent.

3.1.1 Virtual Trainer

Numerous social video games enable users to customize their own avatar appearance and interact with other users' avatars. Hence, many have studied the impact of avatar appearance on users' social behavior and social success in virtual worlds (16; 71; 88). In such an investigation, Banakou and Chorianopoulos found out that users with more elaborate avatars have a better chance at social encounters than users with the default avatars (16). Some found out that male users tend to speak more frequently with female avatars, while female prefers male avatars (16; 71; 88). Also, female users with attractive avatars interact with male avatars more frequently, which suggests "a self-confidence effect induced by the appearance of the personal avatar" (16).

Another work (65) reported that avatars that mimic the outfits of the users give them the highest sense of body ownership and presence, even when the avatars are cartoon-like characters. A study exploring the effects of gender on the perception of different hands reported that women dislike and feel lower levels of presence with male avatar hands, while men accept and feel presence with avatar hands of both genders (120). A different work explored the interpersonal distance in virtual worlds with both avatars and agents (14). Although in the real world men tend to give more interpersonal space to each other, while women usually stay closer together (25), such patterns are not prevalent in virtual worlds.

3.1.2 Virtual Agent

Several studies reported that agent appearance, particularly facial expression (117), gaze (8), voice (89), and body language (115) affect user performance in training scenarios. Guadagno et al. (50) studied the impacts of agent gender on persuasion, where they asked users to listen to persuasive communications from both female and male agents. They found out that users were more persuaded by same gender agents than opposite gender agents. Similar results were found with virtual agents in 3D screen environments (153). A different study reviewed visual stereotypes in agents and cautioned about the risks of applying visual stereotypes to pedagogical agents (53)[9]. Some studies reported that the effect of social agent gender varies in different age groups. For instance, female undergraduate students tend to find young, attractive, and cool female agent most effective since it can enhance their self-efficacy towards being successful as engineers (73; 114; 124). Middle school students also find female agents more effective than male agents (107).

3.2 Motivation

While many have explored the effects of *avatar* gender and appearance on performance and preference in different systems, not much work has investigated *agents*. Besides, most existing works exploring agents are conducted either in 2D or 3D environments, where only a part of the agent is visible. Therefore, the findings of these works may not be generalizable to systems that enable a full visual perception of the agents. In addition, to our knowledge, no prior work has investigated the agent gender effect in a VR interactive training scenario. An understanding of how users react to agents of different genders can facilitate the design and development of more effective and useful pedagogical agents, as well as give us an insight into the human psyche.

3.3 An Experiment

The goal of this study was to explore whether agent gender impacts user performance and preference in a VR training program.

3.3.1 Agent Selection

We conducted an informal study to decide on female and male agent appearance for the study. For this, we created six female and six male characters using the Autodesk Character Generator (Generator). Since *race* and *clothing* are outside the scope of this work, we used similar hair color, skin tone, and clothes for all characters. However, different facial features and body types were used for different characters (Fig 3.1).

Participants. Thirteen volunteers, aged from 20 to 22 years, average of 21.2 (SD = 0.8), participated in the informal study. Twelve of them were male and one was female. Nine of them had used an HMD prior to participating in the study, while the remaining four had no prior experiences.

Procedure. During the informal study, participants completed a questionnaire that asked them to rank the six female and six male characters (Fig 3.1) based on their femininity and muscularity, respectively. The questionnaire included headshots and full-body pictures of all characters. A scale of six was used, where *1* represented the *least* feminine or masculine and *6* represented the *most* feminine or masculine characters.

Results. Fig 3.1 presents results of the informal study, where one can see that *Female 4* and *Male 6* yielded the highest median compared to the other characters. Hence, these two were selected for the final study.

3.3.2 Apparatus

The VR system used in the final study consisted of one desktop computer, 4.20 GHz, 32.0 GB RAM, Windows 10, one Oculus Rift Headset, two Oculus Touch Controllers, and three Oculus Constellation Sensors (3). The three sensors were positioned at the same height (248 cm), two facing the user and another behind the user. The distance between the front sensors was 217 cm, and the distance between the front and back sensors was 455 cm (Fig 3.2). All sensors were tilted at a 30° angle. This setup was used as it yielded a relatively better performance than several other settings tested in a pilot.

We used a Perception Neuron full-body motion capture suit (96), 60 fps, 32 Neurons, for capturing real human motion and gestures in a pseudo training scenario (Fig 3.3). This was to make the virtual agents more human-like. We tested all recorded postures, movements, and gestures in multiple trials before using them in the final study.

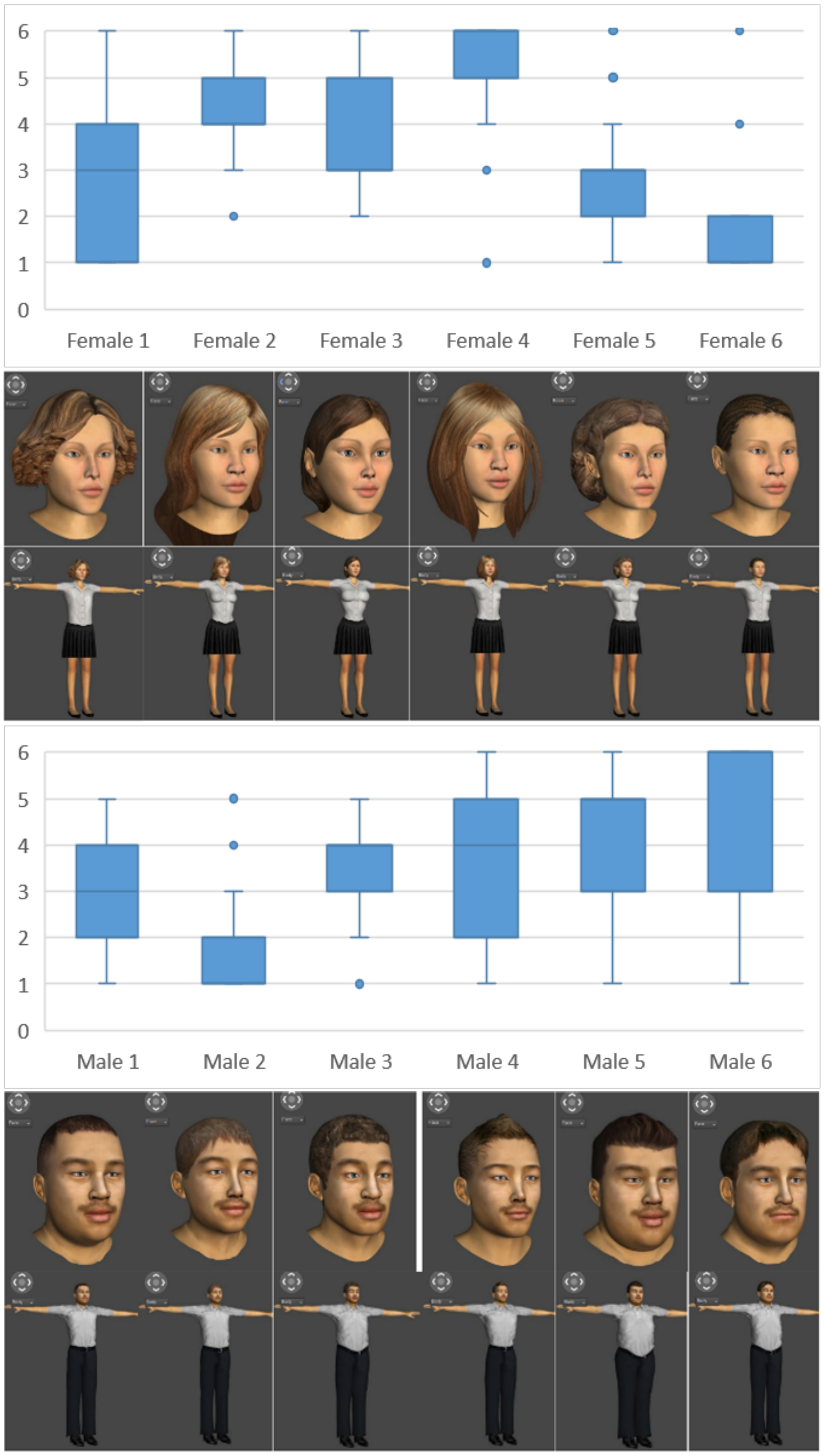
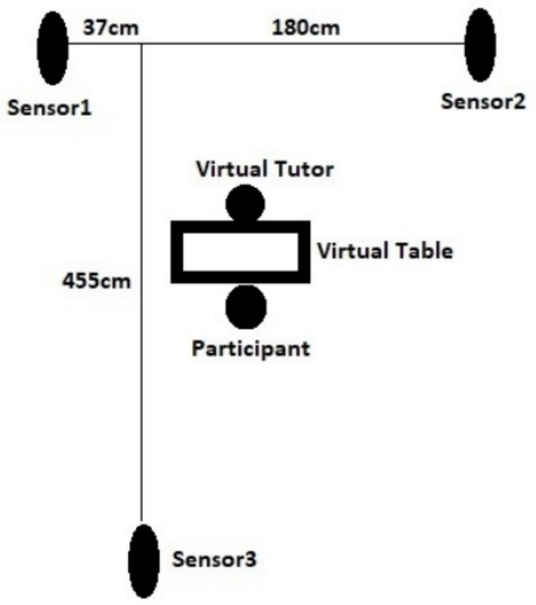


Figure 3.1: Results of the informal study, together with the pictures used in the questionnaire. Error bars represent interquartile range (IQR).



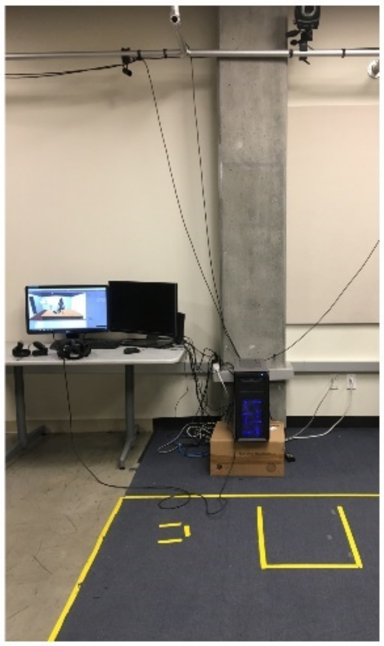


Figure 3.2: The Oculus Constellation Sensor (left) and the VR system setup (right) in the study.

We also tried to make the study environment as welcoming as possible by adding a rug, a water dispenser, and some house plants and wall paintings to the room (Fig 3.4). The study software was written in Unity.

3.3.3 Participants

Twelve volunteers, aged from 21 to 31 years, average 23.55 (SD = 2.62) participated in the study. None of them participated in the informal study. Two of them were female and ten were male. Only two participants had prior experience with HMDs, while the remaining ten had never used an HMD before. All of them were right-handed. They all received a small compensation for participating in the study.

3.3.4 Design

The study used a within-subjects design, where the independent variable was the agent gender and the dependent variables were the performance metrics. There were two levels in the independent variable (conditions): female agent and male agent. Each condition included six tasks. The conditions were counterbalanced to eliminate the effect of order. In each task, participants moved four different color cubes. Each agent used two audio-instructions per gender (in total, four instructions), which were also counterbalanced to reduce the effect of instruction. In summary, the within-subjects design was:

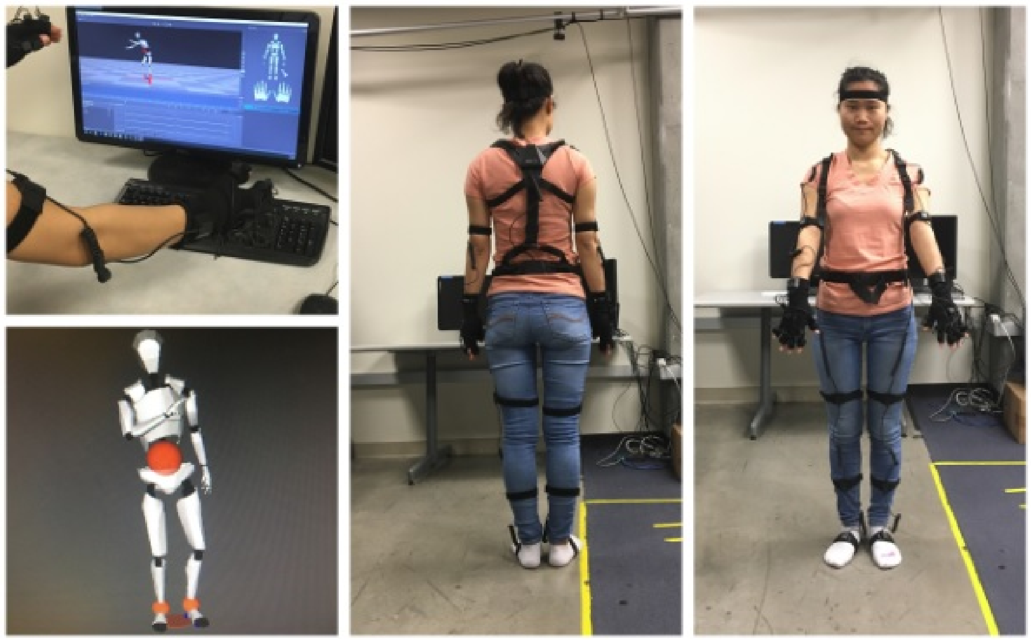


Figure 3.3: Perception Neuron full-body motion capture suit setup.

12 participants ×
2 conditions (counterbalanced) ×
6 tasks (with 2 audio instructions, randomized) ×
3-4 cubes = 432-576 cubes, in total.

3.3.5 Procedure

In the study, all participants took part in a customized Box and Blocks Test (BBT), where female and male agents instructed them to quickly move specific color cubes from one side of the table to the other side of the table (Fig 3.4). BBT is a functional test, commonly used in upper limb rehabilitation to measure the gross manual dexterity of a patient or a person using an upper limb prosthetic device (58). Participants picked up the cubes by gripping the Oculus Touch Controller (which triggered the "Grip" button) when it is in close proximity to the target item, then drop it by releasing the button. For proximity detection, we added a sphere collider to the controller (1.0 cm) and a box collider to the cubes (7.0 cm). The system enabled grasping a cube only when the cube and controller colliders collided. Inside the virtual training system, an agent first greeted the participant, then described the study, e.g., *"Hello! Welcome to the test. In this test, you have to complete three tasks. In each task, you will be asked to pick up four cubes of specific colors, as instructed, and place them on the other side of the table"*. The agent then started giving instructions on moving the cubes, e.g., *"Please pick up all red cubes one by one and put them on the other side of the table"*. The agent alerted the participant on all mistakes, e.g., *"This is not*

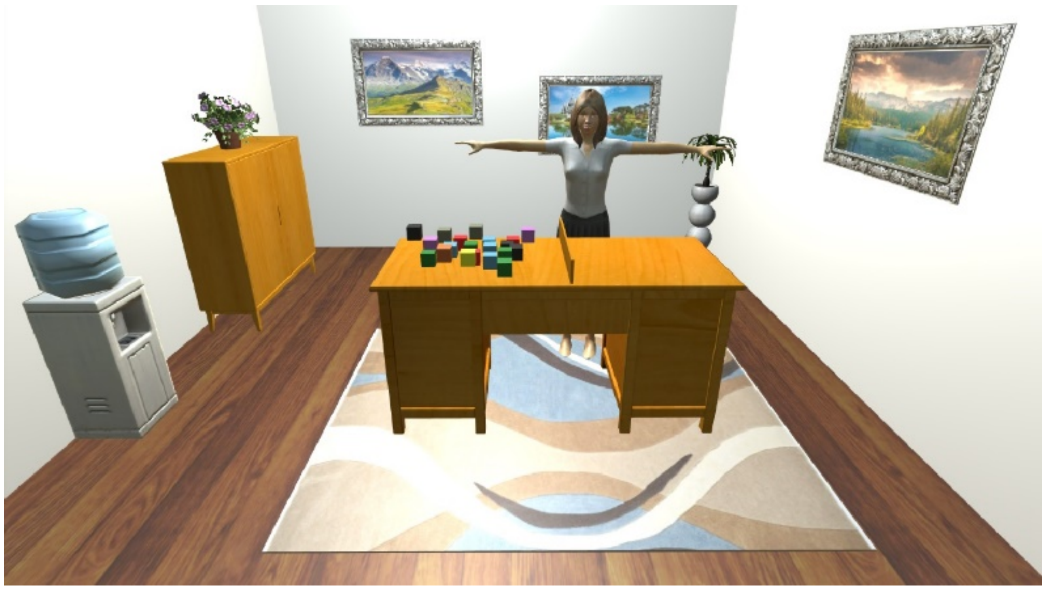


Figure 3.4: The virtual training environment used in the study.

a red cube. Please put it back, and pick the right one" and concluded each *session by thanking the participant, e.g., "Well done! You have completed all tasks. Thanks for supporting our work. Have a nice day!".* Both virtual agents used appropriate posture and hand gestures, recorded through the full-body motion capture suit with actual human subjects. All agent interactions including the instruction audio were identical for male and female agents.

The study took place in an academic research lab. Participants arrived individually to take part in the study. First, we explained the study procedure to all participants and collected their consents. They all filled out a short demography questionnaire. Then, we demonstrated the system, starting with the Oculus Rift. We showed them how to adjust it for a perfect fit and use the controllers. Since all our participants were right-handed, we only used the right-hand controller in the study. We then allowed the participants to practice and get comfortable with the system in a practice block, where they played the Oculus Sample Framework (Framework) for about five minutes. We used this game since, like the study, it involves picking up objects using the controllers. Finally, we started the study. The study software recorded all user actions with timestamps. In addition, we video-recorded all sessions for post-hoc analysis. Fig 3.5 illustrates the study setup.

Upon completion of the study, all participants completed the Simulator Sickness Questionnaire (SSQ) (70). They were also asked to fill out a custom questionnaire that asked them about their preference, and perceived helpfulness, professionalism, and attractiveness of the agents on 7-point Likert scale. Each session lasted for about 30 minutes, including demonstration, practice, breaks, and post-study questionnaires.

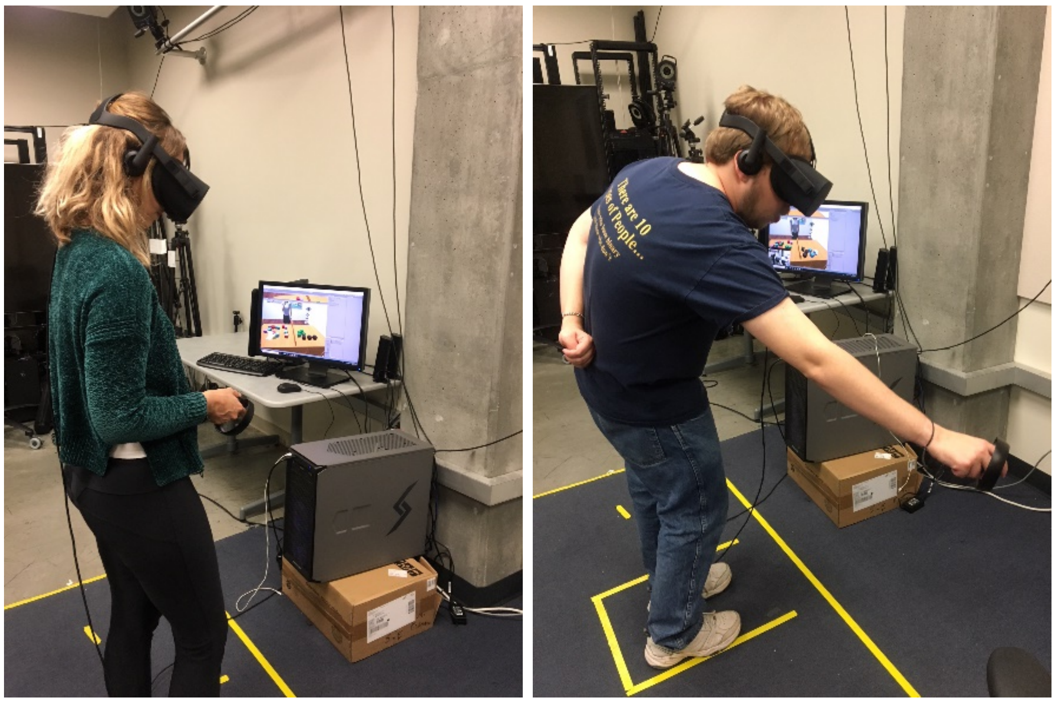


Figure 3.5: Two volunteers participated in the user study. The computer monitor in the background is displaying the participants' point of view (POV).

3.3.6 Performance Metrics and Results

The system automatically calculated and recorded the following metrics.

- **Task Completion Time (Seconds)** signifies the average time the user took to complete a task.
- **Error Rate (%)** represents the average percentage of errors committed by the user per task.
- **Error Correction Time (Seconds)** represents the average time the user took to correct mistakes in each task.

Results. We used a repeated-measures ANOVA on the quantitative data. The results are presented in Table 3.1. For task completion time, the data revealed that participants were 11% faster with the male agent than the female agent. However, an ANOVA failed to identify a significant effect of agent gender on task completion time ($F_{1,11} = 3.33$, $p = .09$). On average task completion time with female and male agents were 15.59 (SD = 4.08) seconds and 17.61 (SD = 11.30) seconds, respectively. For Error rate, participants were 4% more accurate with the male agent than the female agent. Yet, an ANOVA failed to identify a significant effect of agent gender on error rate ($F_{1,11}$= 0.01, $p = .95$). On average operations per task with female and male agents were 3.01% (SD = 2.1) and 2.89% (SD = 2.1), respectively. For error correction time, participants were 45% faster with error correction with the male agent than the female agent. However, an ANOVA failed to identify a significant effect of agent gender on error correction time ($F_{1,11} = 0.46$, $p = .50$). On average operations per task with female and male agents were 1.32 seconds (SD = 6.46) and 1.73 seconds (SD = 1.73), respectively.

Metrics	**Female Agent**		**Male Agent**		*p-value*
	Mean	*SD.*	*Mean*	*SD*	
Task Completion Time	15.59	4.08	17.61	11.30	.09
Error Rate	3.01	2.10	2.89	2.10	.95
Error Correction Time	1.32	6.46	0.73	1.73	.50

Table 3.1: Results of the user study. All times are in seconds. SD signifies standard deviation.

3.4 Qualitative Results

We collected simulator sickness data using the SSQ inventory (70). Individual and total severity scores were calculated using the convention established by Kennedy et al. (70). We used a Wilcoxon Signed-Rank Test on the custom questionnaire data.

3.4.1 Simulator Sickness

The average total simulated sickness score for the system was 10.52 (SD = 7.50). The average nausea score was 23.85 (SD = 15.46), while the average oculomotor score was 2.37 (SD = 4.42). None of the participants reported any disorientating symptoms. Table 3.2 presents the complete SSQ scale means for the system.

3.4.2 Helpfulness

About 83% participants ($N = 10$) rated the two agents equally in terms of helpfulness. The remaining 17% ($N = 2$, 1 *female*, 1 *male*) found the male agent more helpful. Fig 3.6 displays all user responses. However, a Wilcoxon Signed-Rank Test failed to find a significant effect of agent gender on helpfulness ($z = 1.41$, $df = 11$, $p = .18$). The median helpfulness ratings for both agents were 4.0.

3.4.3 Professionalism

About 67% participants ($N = 8$) rated the two agents equally in terms of professionalism, while 25% found the male ($N = 3$, 1 *female*, 2 *male*) and 8% ($N = 1$, 1 *male*)

SSQ Symptoms	Nausea	Oculomotor	Disorientation	Total Score
General discomfort	28.62	0	0	11.22
Fatigue	28.62	0	0	11.22
Headache	38.16	7.58	0	18.7
Eye strain	57.24	15.16	0	29.92
Difficulty focusing	19.08	7.58	0	11.22
Salivation increasing	9.54	0	0	3.74
Sweating	28.62	7.58	0	14.96
Nausea	9.54	0	0	3.74
Difficulty concentrating	19.08	0	0	7.48
Fullness of the Head	47.7	0	0	18.7
Blurred vision	28.62	0	0	11.22
Dizziness with eyes open	28.62	0	0	11.22
Dizziness with eyes closed	9.54	0	0	3.74
Vertigo	0	0	0	0
Stomach awareness	28.62	0	0	11.22
Burping	0	0	0	0
Mean	23.85	2.37	0	10.52
SD	15.46	4.42	0	7.50

Table 3.2: Simulator Sickness Questionnaire (SSQ) scale means for the experiment system.

found the female agent more professional. Fig 3.7 shows all user responses. However, a Wilcoxon Signed-Rank Test failed to find a significant effect of agent gender on professionalism ($z = 0.06$, $df = 11$, $p = .95$). The median professionalism ratings for both agents were 4.0.

3.4.4 Attractiveness

About 50% participants rated the two agents equally with regard to attractiveness. The remaining 50% ($N = 6$, 1 *female* , 5 *male*) found the female agent more attractive. Fig 3.8 shows all user responses. A Wilcoxon Signed-Rank Test found this to be statistically significant ($z = 2.42$, $df = 11$, $p = .01$). The median attractiveness

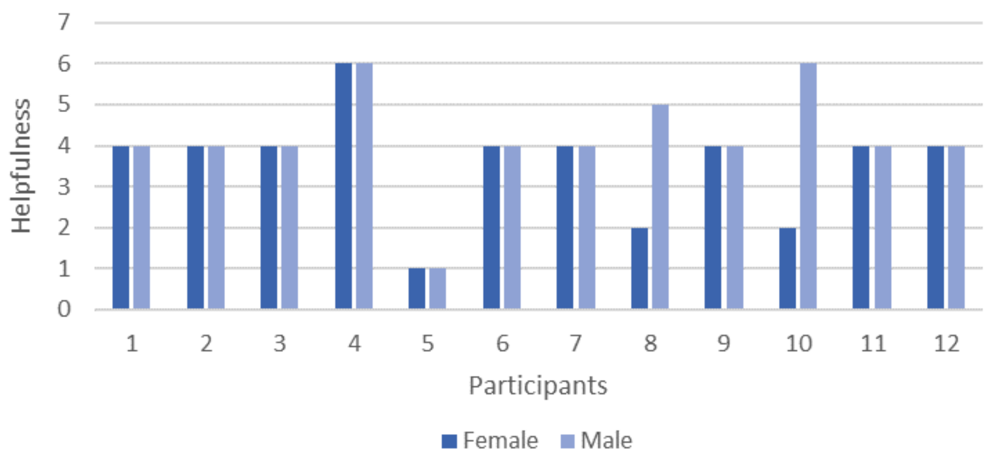


Figure 3.6: User ratings of the two agents with regard to "helpfulness" on 7-point Likert scale, where *1* represents the *least* and *7* represents the *most* helpful.

ratings for the female and the male agents were 4.0 and 3.5, respectively

3.4.5 Preference

About 67% participants ($N = 8$) preferred both agents, while 17% ($N = 2$, 2 *male*) preferred the female and 17% ($N = 2$, 1 *female*, 1 *male*) preferred the male agent. Fig 3.9 illustrates all user responses. A Wilcoxon Signed-Rank Test failed to identify a significant effect of agent gender on preference ($z = 0.06$, $df = 11$, $p = .95$). The median professionalism ratings for both agents were 4.0.

3.5 Discussion

Some participants reported slight discomfort with the experiment system, but none of the symptoms exceeded the nausea and oculomotor severity levels (Table 3.2). The total SSQ score for the system was 10.52, which is negligible (the maximum score possible on the SSQ is 300). Hence, it can be said that the experiment system was appropriate for the study since its side effects were not severe enough to impact user performance or preference. However, some participants criticized the reliability of the system. For example, one female participant remarked, *"The agents did not respond well* [...] *when I was* [*being*] *impatient"*. One male participant (22 years) commented, *"*[*the system must*] *display the actual distance between the hand and the cubes because sometimes I thought I* [*could*] *grab a cube but it turned out that my hand was too far"*. Further, we made some interesting observations during the study. We noticed that some users were rubbing their feet across the carpet to find out whether it felt real. We also noticed that users were making eye contact when the instruction started and

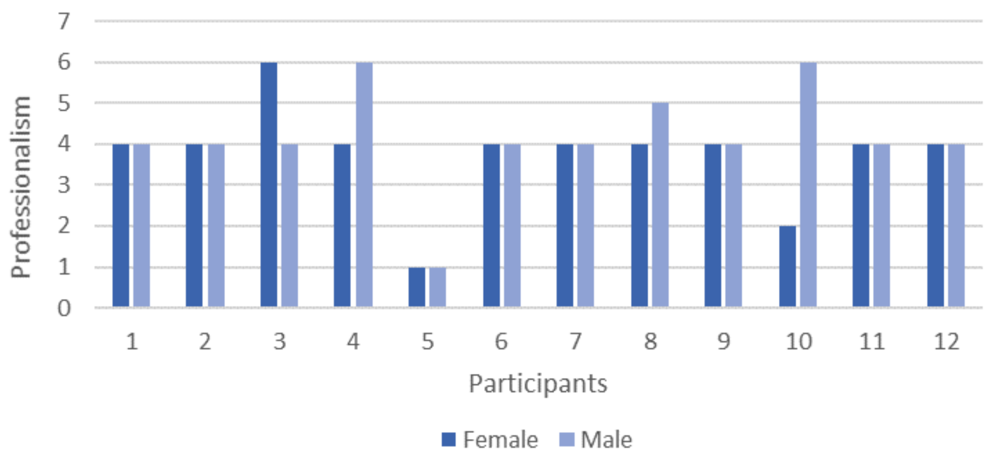


Figure 3.7: User ratings of the two agents with regard to "professionalism" on 7-point Likert scale, where *1* represents the *least* and *7* represents the *most* professional.

then focused on the task. They made eye contact again, when the agent warned them about a mistake.

Participants yielded a relatively better performance with the male agent than the female agent. On average, they were 11% faster and 4% more accurate with the male agent (Table 3.1). They were also 45% faster in correcting errors. Nevertheless, statistical tests failed to identify a significant effect of agent gender on performance. Interestingly, the effect of agent gender on task completion time almost reached significance ($p = .09$). Hence, it is possible that this effect will reach significance with an increased number of participants, which would conform to results from older studies (50; 153) that showed that users are more persuaded by same gender agents than opposite gender agents, as most of our participants were male.

Statistical tests also failed to identify significant effects of agent gender on user preference (Fig 3.9), perceived helpfulness (Fig 3.6), and perceived professionalism (Fig 3.7). Both agents yielded comparable ratings from the users, which suggests that participants were comfortable with taking instructions and working with both agents. One female participant commented, *"female agent [was] as helpful as male and [vice versa]. Both seemed very professional".* Interestingly, significantly more users found the female agent more attractive than the male agent. This is most likely due to our male dominated sample, as a similar trend was reported with virtual avatars, where male users preferred speaking with female avatars than male avatars (16; 71; 88). Further investigation is needed to fully explore this behavior.

3.5.1 Limitations

We acknowledge several limitations of the study. First, it failed to recruit adequate female participants to investigate any potential effects of user gender on performance

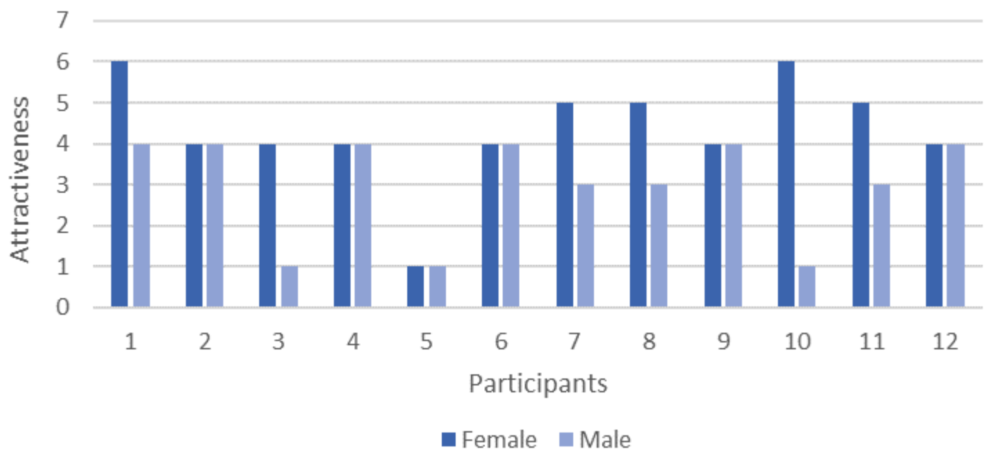


Figure 3.8: User ratings of the two agents with regard to "attractiveness" on 7-point Likert scale, where *1* represents the *least* and *7* represents the *most* attractive.

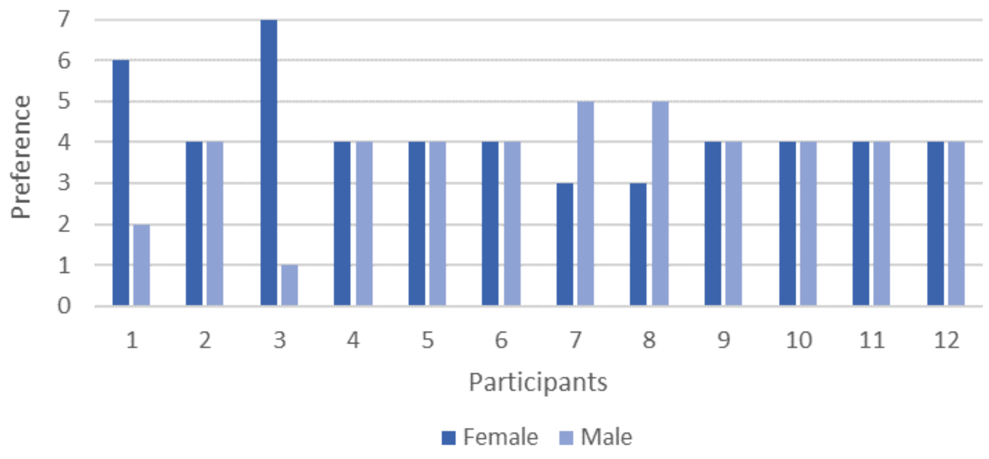


Figure 3.9: User ratings of the two agents with regard to "preference" on 7-point Likert scale, where *1* represents the *least* and *7* represents the *most* preferred.

and preference for different gender agents. Ten of our twelve participants (83%) were young male adults. Hence, we recommend caution in interpreting the results since they may not be generalizable to a larger population. Second, the study used a binary gender classification, mainly for simplicity, while both users and agents could have a range of gender identities and expressions, not just male and female.

3.6 Conclusion

This chapter presented the results of a small-scale exploratory study that studied the effects of virtual agent gender on user performance and preference in a VR training program. In the study, twelve participants (10 *male*, average age 24 years) participated in a custom Box and Blocks Test (BBT) under the guidance of a female and a male agent. On average, participants performed better with the male agent than the female agent, with respect to task completion time, error rate, and error correction time. However, no significant difference was identified. The study also failed to find a significant effect of agent gender on preference, perceived helpfulness, and perceived professionalism. Interestingly, significantly more participants found the female agent more attractive than the male agent, presumably due to the study's male-dominated sample.

This work merely scratches the surface of the topic. We hope the findings of this work will inspire others to fully explore any potential effects of agent gender on performance and preference of a diverse group of users, including users of different age, gender, education, culture, technological experience, and socio-economic background. Among other applications, advances in this area will inform the automatic selection of virtual trainers in order to improve interactions in VR systems.

Chapter 4

Effects of Virtual Agent Feedback Strategies

In this chapter we identify, model, and evaluate feedback behaviors for virtual trainers assisting users to accomplish a given task involving manipulation of objects in a virtual world. With the goal of minimizing task-specific characteristics, a simple sorting task was chosen based on sorting areas of countries represented on cubes. This task is composed of basic elements that are often present in a variety of scenarios: object manipulation for task execution, observation of results, and repetition in progressively difficult cases.

Given a task to be solved, we focus on the specification and evaluation of feedback behaviors for the virtual trainer, such that it can effectively demonstrate the task to be executed and provide feedback to assist the user to perform the task. A pilot study was first conducted with a human trainer in order to identify and specify feedback strategies. As a result of this study two feedback strategies were specified. In the first feedback strategy the virtual trainer provides Correctness Feedback (CF) by fully presenting the correct responses to the users at each stage of the sorting task. In the second feedback strategy the virtual trainer instead provides Suggestive Feedback (SF) by incrementally notifying the users if and how a current response is wrong in order to enable users to correct their responses by themselves.

An immersive VR system was then implemented to evaluate the two feedback behaviors. The system immerses the user with an Oculus Rift Head-Mounted Display (HMD), incorporates speech recognition and synthesis for communication, and implements a number of behaviors for replicating intuitive human-like interactions. See Figures B.1 and 4.2.

Both CF and SF strategies represent valid approaches for a virtual trainer to follow. While CF explicitly presents corrected results at each stage of the task, the SF strategy requires the user to be engaged in correcting results by themselves. To our knowledge this work is the first to investigate such strategies in the context of direct interaction with a human-like virtual trainer in an immersive 3D VR environment.

We have collected performance data from 14 participants using our system employing both feedback strategies. The CF strategy was preferred by participants, was more effective time-wise, and was more effective in improving task performance skills. The overall system was rated comparable to hypothetically performing the same task with real interactions. Additional findings include interaction with speech commands was rated unfavorably, and HMD visual resolution and quality were rated as not interfering or distracting from the task execution. A poster abstract was previously published about this project(121) and this chapter presents our full work.

4.1 Related Work

Virtual environments have been employed in a number of applications in education, training and beyond (33; 41; 24; 2; 122); and the inclusion of autonomous virtual humans in such types of applications is a natural way to achieve effective human-like interactions in virtual environments.

In order to be effective a number of specific behaviors have to be implemented in an autonomous virtual trainer. In this chapter we address the specification, implementation and evaluation of feedback behaviors, and we also present the complete integration of the evaluated behaviors in our virtual human training system.

4.1.1 Virtual Agents and Training Systems

Virtual agents can help users to learn by utilizing a variety of behaviors based on gestures, natural language, gaze, and facial expressions (78; 72; 27; 20; 122; 127). Well-designed non-verbal behaviors for virtual agents are in particular important as they can increase the user's attentiveness, positivity, and also rapport (131), which defines the ability to maintain harmonious relationships based on affinity (47).

A number of training systems relying on virtual characters acting as interactive demonstrators, virtual teammates, virtual teachers and other roles have been developed. Steve (112) represents one of the first systems implementing an autonomous virtual trainer specifically designed to train people to operate ship engines. Another example, among many others, is AutoTutor (46) which is a virtual tutor designed to teach concepts in science and mathematics with multiple design strategies like using dialogue, feedback, corrective statements, hints, fill-in-the-blank questions, and requests for more information from the user. Significant past research has been dedicated to developing the necessary movement synthesis and behavioral modeling algorithms for accomplishing autonomous virtual humans and a number of software solutions have been developed for facilitating their integration in applications, such as the Virtual Agent Interaction Framework (VAIF) (48) and the Virtual Human Toolkit (56).

4.1.2 Feedback Strategies

Several works have evaluated different types of feedback strategies in the context of classroom learning (51; 95; 125). Shute (125) provides an elaborate review of feedback types, organizing them as: no feedback, verification, correction, try again, error flagging, elaborate, attribute isolation, topic contingent, response contingent, hints/cues/prompts, bugs/misconceptions, and informative tutoring. Informative tutoring is the most complex type of feedback which was defined as "information communicated to the learner that is intended to modify his or her thinking or behavior to improve learning". It represents and includes verification feedback, error flagging, and strategic hints. Some studies (17; 108) have reported that giving learners informative feedback was more effective than only giving them correct answers directly.

Previous works have also investigated feedback strategies in computer-based learning systems. Attali (11) has tested four feedback types: no feedback, immediate knowledge of the correct response, multiple-try feedback with knowledge of the correct response, and multiple-try feedback with hints after an initial incorrect response, which was found to be the most effective type of feedback. Another study (133) reported that elaborated feedback during learning was superior to only providing answer correctness. Falcao (36) combined interactive tangible tabletops and physical actions to study the discovery-based hands-on learning among children with intellectual disabilities. These works however have mostly studied feedback strategies in scenarios involving complex rules, complex knowledge retention, or physically involved tasks. For simple memory-based tasks direct display of correct solutions has been identified as an effective feedback (125).

While these previous studies provide several guidelines that can be used for implementing feedback behaviors, no previous work has directly investigated this topic in the context of a virtual trainer interacting with users in an immersive VR-based system.

One particular type of gesture that is useful for delivering feedback behaviors is pointing. Several works have investigated pointing gestures for virtual humans to become able to identify spatial positions, a capability which is very important and which has been explored in several previous works (61; 112). However, the use of pointing in feedback behaviors has not been investigated. In our work we focus on interactive tasks where the user is required to manipulate virtual objects and the virtual trainer employs pointing as one way to deliver feedback during task execution.

In this chapter we identify, specify and evaluate two feedback strategies of broad applicability: Correctness Feedback (CF), which focuses on only confirming correct answers, and Suggestive Feedback (SF), which provides informative feedback during the learning activity in our virtual training system. We also present results that favor the use of the CF strategy.

4.2 Pilot Study

In order to specify effective feedback strategies we have conducted a pilot study with initial versions of our training scenario between two persons without the use of any computer assistance.

4.2.1 Apparatus

No computer devices were used during the pilot study. A real human played the role of the virtual trainer. Printed paper cards were used to represent information to be sorted. The material sets were printed on a square paper of 7 cm wide, and the material sets used in this study were not repeated in the final study. The sorting task was based on sorting the information illustrated in the material sets.

4.2.2 Design

In the considered sorting task the users needed to memorize a limited amount of information. The entire task was executed in about 10 to 20 minutes. As a tutoring system, the goal was to design a simple scenario that also offered some challenges (126).

Memorial Materials and Questionnaires

Two different material sets were used during the study, each material set included a group of 9 pictures. The first set integrated the display of country maps, flags, and names, as shown in Figure 4.1-top. The second set displayed ancient buildings and their names, as shown in Figure 4.1-bottom. The first material set had to be sorted from largest country land area to smallest country land area. The second material set had to be sorted from oldest building to newest building. In both cases, the ordering was from the user's left side to the right side. To quantitatively evaluate the user's performance we have applied pre- and post-test questionnaires. The pre-test questionnaire had one material set randomly printed, and the user needed to sort it based on his/her own previous knowledge. The post-test questionnaire had a new random material set, and was sorted by the user after the interactive activity described below.

Task Design

Each sorting task consisted of sorting the cards gradually through interaction with the human trainer. Each sorting activity was organized in 4 stages. In stage 1, three cards were presented to the user on the table in random order. The user would pick the cards and sort them according to the sorting criterion. The human trainer then provided some sort of feedback until the three cards were sorted correctly. At this point, the task would proceed to the next stage where the same cards of the previous stage would appear again, but with two additional cards. In this way the user was

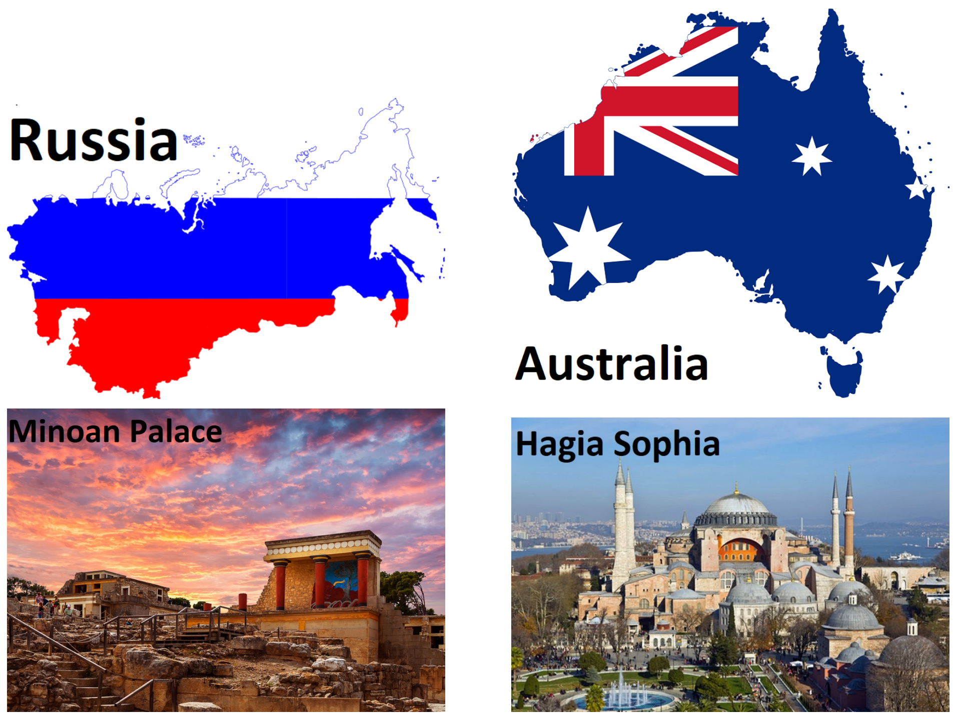


Figure 4.1: Example pictures used in the pilot study.

supposed to incorporate the information on the two new cards when sorting the entire set of cards. Additional pairs of cards were added at each new stage until reaching stage 4 when nine cards were sorted by the user.

Study Design

First, the members of our research team have executed our pilot study scenario and two overall feedback strategies were identified: Correctness Feedback (CF) was based on correcting responses at each sorting stage, and Suggestive Feedback (SF) was based on just notifying, at each stage, the cards that were sorted incorrectly.

The pilot study then evaluated these strategies using two independent variables:

- Feedback. The two conditions of the independent variable "feedback" were: correctness feedback and suggestive feedback.
- Stage. The four conditions of the independent variable "stage" were: stage 1, stage 2, stage 3, and stage 4.

The two dependent variables used in this study were:

- Performance Improvement. The pre-test and post-test questionnaires were used to record the users' sorting scores of the material sets, and the task performance improvement of a participant was represented by his or her sorting scores.

- Stage execution time. The time duration is taken for producing a sorting for each stage of the task. This time excludes any interaction time between the user and trainer.

A within-subjects design method was used. Each participant experienced the two feedback conditions and each with the two material sets. The order of feedback conditions and material sets was counterbalanced to reduce their effect on the dependent variables. In summary, the within-subjects design was: 4 participants × 2 conditions (with two material sets) × 4 task stages × 3, 5, 7, 9 country cards = 192 cards in total.

4.2.3 Participants

Four volunteers were involved in our pilot study: two males and two females, with ages ranging from 27 to 32 years old, and all were English speakers. One of our researchers played the role of the human trainer. Each participant performed the activity twice, each time experiencing a different feedback strategy and a different material set. The order of delivering the two strategies was counterbalanced among the participants.

4.2.4 Procedure

The study was conducted in an empty room. The human trainer first collected the oral consent from the participant and explained the study. Then, for each material set and feedback strategy, the participant performed the pre-test questionnaire, the main sorting activity while receiving feedback from the human trainer, and the post-test questionnaire.

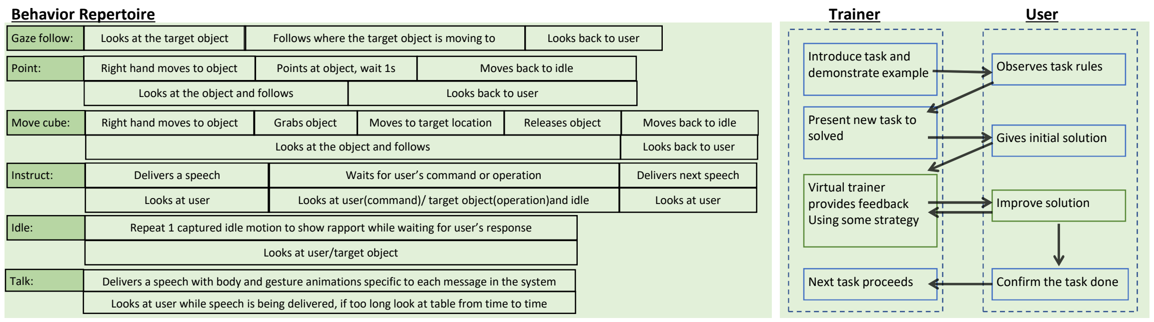

Behavior Repertoire
Gaze follow:
Looks at the target object
Follows where the target object is moving to
Looks back to user
Point:
Right hand moves to object
Points at object, wait 1s
Moves back to idle
Looks at the object and follows
Looks back to user
Move cube:
Right hand moves to object
Grabs object
Moves to target location
Releases object
Moves back to idle
Looks at the object and follows
Looks back to user
Instruct:
Delivers a speech
Waits for user's command or operation
Delivers next speech
Looks at user
Looks at user(command)/ target object(operation)and idle
Looks at user
Idle:
Repeat 1 captured idle motion to show rapport while waiting for user's response
Looks at user/target object
Talk:
Delivers a speech with body and gesture animations specific to each message in the system
Looks at user while speech is being delivered, if too long look at table from time to time
Trainer
User
Introduce task and demonstrate example
Observes task rules
Present new task to solved
Gives initial solution
Virtual trainer provides feedback Using some strategy
Improve solution
Next task proceeds
Confirm the task done

Figure 4.2: Left: the behavior repertoire available to the virtual trainer. Each behavior (except the first one) implements an action synchronizing arm motion and gaze movement as illustrated in the horizontal bars next to each behavior. Right: the main interaction phases between the virtual trainer and the user.

With CF, when a wrong sorting was presented, the human trainer would correct the sorting and say "Here is the correct order, please study it and let me know when to continue”. With SF, the human trainer would point to each pair of cards incorrectly ordered and say “These two cards are in the wrong order, please correct them”. The whole study took around 30 minutes to finish.

4.2.5 Results

In order to evaluate the results from the pre-test and post-test questionnaires, we defined a “sorting score” to quantify a participant’s performance improvement. For a set of 9 cards, the total possible combination of pairs is C(9,2)=36. The sorting score is defined as the number of pairs in the correct order divided by the total number of pairs (36) and multiplied by 100 in order to express values as percentages.

With this definition, the mean scores of CF in the pre-tests and post-tests were 56.94% and 95.83% respectively. The mean scores of SF were 69.44% and 99.31% respectively. The mean time to complete the entire task with the CF strategy was 229.8s, and with the SF strategy was 223.9s. These results show that both CF and SF were effective as they led to 68.30% and 43.02% performance improvement. However, participants needed increasingly more time to complete each stage under SF than under CF, given that a growing number of suggestions were needed before completing each stage.

All participants reported that the elements on the cards helped them memorize the needed information. For example, when the country material set was used, one participant said “I can’t remember the country’s name, but I do remember the colors and shape of the country I was sorting”. Figure 4.1-top illustrates the used shapes and colors. The land area of Russia is larger than Australia. In the building material set, one participant said “It is hard to tell how old the building is, but those devastated buildings must be older than those completely built”. Indeed, as is the case in Figure 4.1-bottom, the Minoan Palace is older than Hagia Sophia.

In our main study we have changed the material sets and redesigned the questionnaires in order to better constrain the participant to assimilate the intended information. We have also updated the delivery of the SF strategy to reduce duration times and participant stress due to a possible large number of suggestions. These changes are presented in Section 4.4.

4.3 System Implementation

A training system was implemented providing the needed capabilities for executing the required interactions observed in our pilot studies. The system was built with the Unity game engine connected to an Oculus Rift. We use the Oculus Unity Integration plugin to connect to the Rift controllers. The general system is designed as follows: the system monitors the user input, runs a task-dependent action decision model, and then decides which behavior the virtual trainer has to execute. This process is repeated until the simulated task ends. This interactive process is illustrated in Figure 4.2-right.

In order to be able to execute all needed actions, the virtual trainer was equipped with six basic behaviors, as detailed in Figure 4.2-left. The movement of the "gaze follow" behavior is controlled by Inverse Kinematics (IK); for "point" and "move cube" the right arm is also controlled by IK while the hand shape is interpolated between a neutral hand shape and the target hand shape (pointing or grasping shape). The "talk behavior" includes lip syncing implemented with the Unity plugin SALSA. User voice commands are analyzed by the Windows Speech Recognition module. The "idle" behavior is animated with motion-captured animations captured with a Perception Neuron full-body motion capture suit. In order to be realistic most of the behaviors synchronize arm movements with a gaze attention model specifically designed for each behavior as depicted in the "timeline boxes" in Figure 4.2-left. This behavior repertoire was sufficient to implement the target scenarios and feedback strategies.

4.4 User Study

Our main user study took place at our research laboratory equipped with a high-end computer station connected to the Oculus Rift and running our simulation system.

4.4.1 Design

Our simulation system replicated the same overall scenario of the pilot study, but with several improvements.

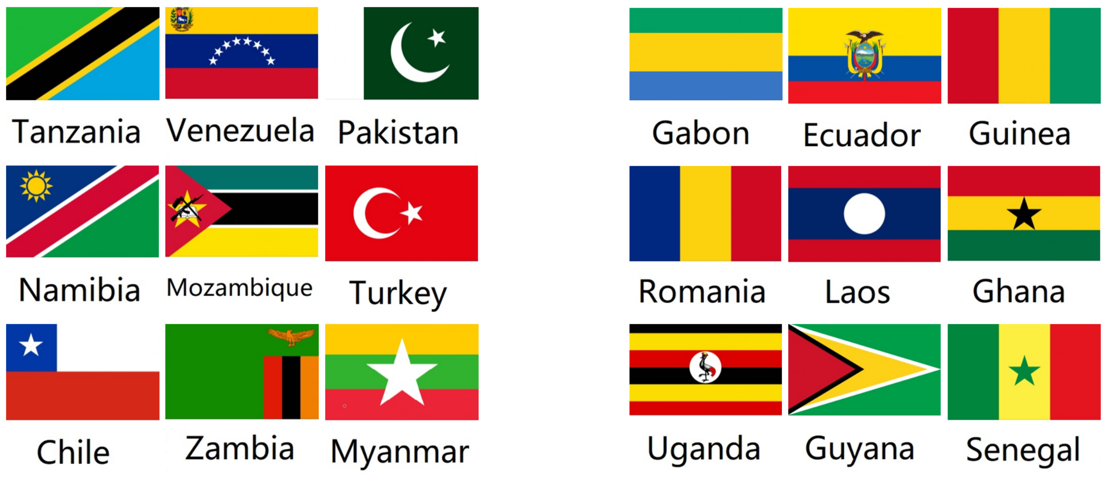


Figure 4.3: Country material sets used during main study

Memorial Materials and Questionnaires

Only country materials were used. We used one material set in order to evaluate users on the same topic and discarded the building material set because it allowed different types of information to influence the participants. The country material set was also updated to reduce non-applicable information influence, and country border outlines were removed. Two non-intersecting sets of countries were used (see Figure 4.3-left and Figure 4.3-right). The sets were chosen such that the countries have similar land areas. The pre-test and post-test questionnaires have been correspondingly updated to only list countries by names.

Task Design

The overall task procedure is the same as performed in the pilot study; however, the task now is based on sorting virtual cubes representing countries instead of sorting cards.

Feedback Strategies

Based on feedback obtained during the pilot study, the SF strategy was updated in the following way: instead of showing the wrong pairs of items one at a time, the virtual trainer now points to all the wrong items at the same time. In this way, when there are more than 2 wrong cubes, the feedback is delivered in a shorter time and participants have to be more engaged since correcting the answer is not anymore obvious and mechanical. When the user provided a wrong sorting the virtual trainer would say “Attention, those [two/three.../nine] countries are in the wrong order” while pointing at those wrong countries one by one. In order to facilitate the perception of the users white squares marking the location of the cubes turned red each time the virtual trainer pointed at them. Figure 4.4 illustrates the execution of both strategies by the virtual trainer in our system.

Study Design

The same within-subjects design was used in the main study but with 14 participants: 14 participants $\times$ 2 conditions (with different learning material sets) $\times$ 4 task stages $\times$ 3, 5, 7, 9 country cubes=672 cubes in total.

4.4.2 Participants

14 volunteers from the university community participated in the user study. None of them were involved in the pilot study. Their age ranged from 17 to 25 years old, with an average of 20.14 years (SD=1.99). They were all fluent in English, 5 of them were female and 9 male, 5 were left-handed and 9 right-handed. Only 3 participants had experienced VR before, among them only 1 had used Oculus Rift before participating in our study. Participants received $5 cash as compensation for their time in our study.

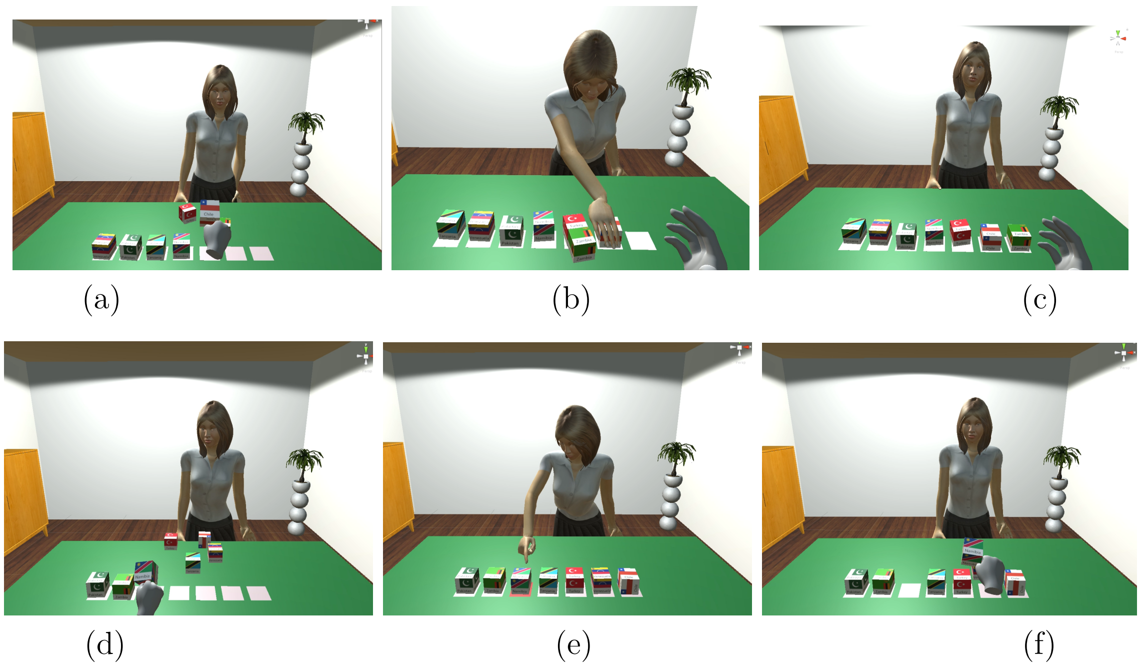

(a) (b) (c)
(d) (e) (f)

Figure 4.4: The Correctness Feedback (CF) strategy is illustrated in images a-c. After the user arranges the cubes (a) and completes an incorrect sorting in the current task stage, the virtual trainer will manipulate the cubes and re-arrange them in the correct order (b). The user can then observe the correct solution as long as needed (c) until saying "continue" in order to move to the next stage. The Suggestive Feedback (SF) strategy is illustrated in images d-f. After the user manipulates the cubes (d) and completes an incorrect sorting of the cubes in the current task stage the virtual correcter will then point to the cubes which are in the wrong position (e). The virtual trainer will point to all of the cubes that are wrong and say that their positions are wrong. The user will then rearrange the cubes (f) to propose a new sorting. The process repeats until all cubes are placed in the correct locations. When this is detected the virtual trainer notifies that the solution is correct and the user can then observe the correct solution as long as needed until saying "continue" in order to move to the next stage.

4.4.3 Procedure

Before the test day participants received an email explaining the purpose and main procedures of the study. Participants came in one by one at their scheduled times. They were introduced again to our study, signed the consent form, and filled out a demography questionnaire. The user then completed two learning activities (one with SF and the other with CF) including the pre-test and post-test questionnaires. Two additional post-study questionnaires were included: a simulator sickness questionnaire (SSQ) and a questionnaire asking users to rate the feedback strategies, the virtual trainer character, and the overall VR experience with 7-point Likert scales. The system recorded all completion times for later analysis. Overall, each participant

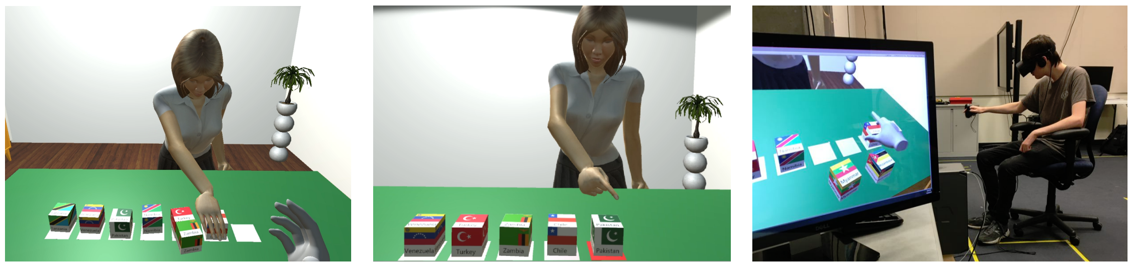


Figure 4.5: We present an autonomous virtual trainer that can assist users to learn a given task by using different feedback strategies involving manipulation (left) and pointing (center). Immersed users interact with the virtual trainer with voice commands while manipulating virtual cubes using Rift controllers in order to learn the task (right).

spent around 50 minutes going through all the activities. Figure 4.5-right illustrates one participant in the study.

4.5 Results

We summarize in this section our results.

4.5.1 Qualitative Results

The Simulator Sickness Questionnaire (SSQ) obtained the average total simulated sickness score of 9.82 (SD=9.8); the average nausea score of 19.08 (SD=18.78), with 11 participants reporting this symptom; the average oculomotor score of 3.79 (SD=5.99), with 6 participants reporting this symptom; and the average disorientation score of 1.75 (SD=4.62), with only 2 participants reporting this symptom.
The Simulator Sickness Questionnaire (SSQ) analyzed scale means is shown in Table 3.2 (in Appendix). The average total simulated sickness score for the system is 9.82 (SD=9.8). The average nausea score is 19.08 (SD=18.78), 11 participants reported this symptom; the average oculomotor score is 3.79 (SD=5.99), 6 participants reported this symptom; and the average disorientation score is 1.75 (SD=4.62), only 2 participants reported this symptom.

4.5.2 Quantitative Results

We used a repeated-measures ANOVA with alpha of 0.05 for all analyses. ANOVA failed to identify a significant effect of feedback on performance improvement ($F_{1,13}$=1.66, p=.22). ANOVA failed to identify the significant impact of feedback and pre-test and post-test questionnaires test order on the user's performance improvement, the result is:$F_{1,13}$=0.12, p=.73. ANOVA identified a significant impact of pre-test and post-test questionnaires test order on user's performance improvement, the result is:

$F_{1,13}$=49.44, p=0.000009. The sorting scores representing the performance improvement of all 14 participants are displayed in Table 4.1.

Participant #	**CF Sorting Score (%)**		**SF Sorting Score (%)**	
	pre-test	*post-test*	*pre-test*	*post-test*
1	36.11	97.22	41.67	69.44
2	41.67	100.00	50.00	100.00
3	33.33	80.56	66.67	100.00
4	30.56	63.89	66.67	38.89
5	36.11	100.00	36.11	91.67
6	38.89	72.22	55.56	100.00
7	55.56	97.22	55.56	86.11
8	52.78	44.44	38.89	61.11
9	50.00	86.11	69.44	100.00
10	80.56	80.56	44.44	80.56
11	63.89	100.00	63.89	80.56
12	47.22	36.11	27.78	75.00
13	27.78	66.67	50.00	52.78
14	33.33	63.89	50.00	100.00
Mean	44.84	77.78	51.19	81.15
SD	14.16	20.20	12.19	19.05

Table 4.1: Mean and standard deviation for obtained scores.

The average execution time for each stage of the task has been computed and the result is shown in Figure 4.6. The average total time needed to complete the tasks with CF was 339s, while with SF was 559s, which is 65% higher than the CF time.

Two linear regressions were calculated to predict the average stage execution time based on stage under both feedback strategies. For CF, no significant regression equation exists (F(1,2)=13.71, $p>.05$). But for SF, a significant regression equation was found (F(1,2)=365.36, $p<.05$) with R^2 of .995.

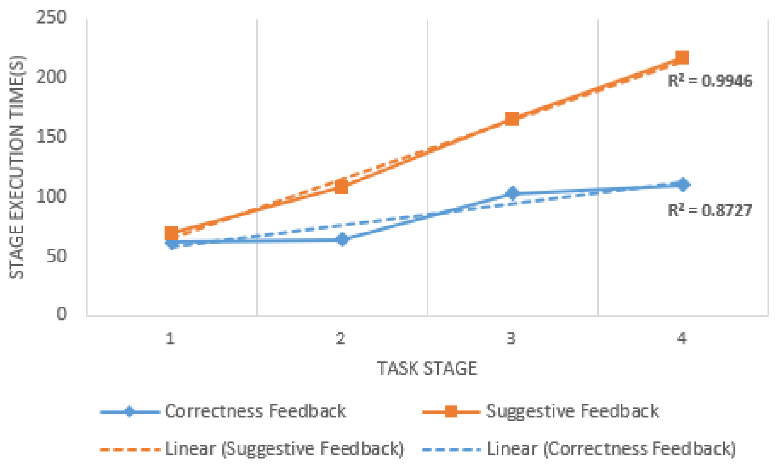


Figure 4.6: Average stage execution times with CF and SF.

4.5.3 Questionnaire Analysis

In addition, our post-study questionnaire provided 14 questions to evaluate several aspects of the system. The Wilcoxon Signed-Rank test was conducted for questions 1-5 but failed to identify a significant impact of feedback on participants' choices. However, for questions 1,2 and 4, the median values of CF are higher than SF. Questions 6-14 are single questions about participants' opinions on the system, virtual trainer, and the overall VR experience. The results are listed in Table 4.2. The 14 questions are listed below:

1, I think feedback CF/SF was effective for learning the given task;

2, I would imagine that most people would like to use feedback CF/SF for learning the task;

3, Feedback CF/SF made me feel bored/tired sometimes;

4, I prefer to use feedback CF/SF for learning instead of learning from a real human;

5, I think having the animated character in feedback CF/SF did not help to complete the task;

6, I like to use speech commands with both feedback;

7, I prefer using button commands to speech commands in the system;

8, I liked the way the character looked at me when it talked with me;

9, It would be important to see more human-like behaviors from the virtual character;

10, I would like to interact more with the virtual trainer in order to complete the tasks in the virtual environment;

11, I felt comfortable/relaxed while doing the tasks in the virtual environment;

12, The visual display quality interfered with or distracted me from performing the activities;

13, I was more proficient in interacting with the virtual environment at the end of the experience than start;

14, I felt completely immersed/involved in the virtual environment during the execution of the tasks.

Question Number	z	p	**Median**$_{CF}$	**Median**$_{SF}$	**Median**	**Mean**
1	-0.63	.53	6	5	-	-
2	-0.47	.64	5.5	5	-	-
3	-0.51	.61	3	3	-	-
4	-0.45	.65	4.5	3.5	-	-
5	-0.05	.96	3.5	3.5	-	-
6	-	-	-	-	4	4
7	-	-	-	-	5.5	5.21
8	-	-	-	-	4	4.21
9	-	-	-	-	5	4.57
10	-	-	-	-	4	4.14
11	-	-	-	-	6	5.5
12	-	-	-	-	2.5	2.86
13	-	-	-	-	5.5	5.64
14	-	-	-	-	6	5.64

Table 4.2: Post-study questionnaire results.

4.5.4 Discussion

Some participants reported slight discomfort for experimenting with our system, but none of the symptoms exceeded nausea, oculomotor, or disorientation severity levels. The total SSQ score of the system was 9.82, which is negligible (the maximum score possible on the SSQ is 300). Hence the system was appropriate for the study since its side effects were not severe enough to impact user performance or preference.

For both CF and SF strategies, the mean post-test sorting scores were significantly higher compared to their respective mean pre-test sorting scores after experiencing the learning activity in our system. Statistical tests failed to identify a significant difference between CF and SF strategies on user's performance outcome; however, the increments were 72.79% for CF and 58.53% for SF, showing that CF was more effective than SF for increasing the performance outcome. A significant linear regression equation for stage execution time was found for SF, but not for CF, and the total average time needed to complete a task with SF was 65% longer than with CF. The reason is that numerous suggestions were given by the virtual trainer, in particular as the stage number increased.

Overall, although no significant difference has been identified for both strategies, CF is more effective in the mean score increment aspect. From the analysis of task execution time, CF was more efficient time-wise for our task scenario. These findings agree with the notion that "complexity of feedback may be inversely related to both ability to correct errors and learning efficiency" (75; 125).

For the post-study questionnaire analysis (Table 4.2), no significant impact of feedback strategies on user choice has been identified. However, comparing the median values of CF and SF, more participants thought CF was more effective for performing the sorting task and more participants mentioned a preference to use CF for performing the sorting task. Besides, our annotations show that 5 participants were uncomfortable when the virtual trainer repeatedly pointed out they did wrong sortings in a given task stage, which made them not proceed with the task session. One of the participants demonstrated significant anxiety at the last stage of a SF session even though the participant insisted on finishing the task. No one reported negative comments about the CF strategy during the user study. One participant said "The first test (the CF test) was much faster and easier for me to use". These points show that the CF strategy was preferable to the SF strategy in our system.

Interestingly we found out that the speech recognition model in the system was not as welcomed as we expected. In question 6 participants rated the speech command in the median and average values both at 4, which means slightly approving it. However in question 7, when asked about the option of button commands, participants answered they would prefer using button commands to speech commands. This could be explained by some participants having different language backgrounds, but even if not, they all had to pronounce a standard English accent in order for the system to correctly understand commands, otherwise they had to repeat voice commands several times until recognition in order to proceed with the session. We noticed that after three times repeating the command "continue" without being recognized the participant's voice volume would become lower and lower, demonstrating frustration, and in this case the experimenter manually instructed the system to proceed with a keyboard key without the participant noticing it. Participants rated the gaze tracking behavior as 4, probably because most of their attention was focused on the task state on the virtual table. In any case most of them expected to see a more human-like virtual trainer than the one we presented in our system.

Another interesting finding is that participants rated very low the possibility that the visual display quality interfered with or distracted them from performing the required activities. The median value was only 2.5. Perhaps, given that only 3 of the 14 participants had previous experience with a VR system, participants felt more excited to experience the VR display than frustrated by the drawbacks of using it.

Participants rated highly (rating 6) that they felt comfortable, relaxed and immersed while performing the tasks in the virtual environment.

Question 8 evaluated the gaze behavior incorporated in our system and participants rated it as an effective behavior of the virtual trainer. The evaluations of questions 10, 11 and 14 indicate that the design of the virtual trainer in our system was well received and effective for conducting the learning tasks. The evaluation of question 13 also indicates user improvement after being immersed in our VR system. In these questions the mean and median ratings show positive ratings from users.

Overall our results indicate that in a short-memory training task such as the one we have used, the virtual trainer would be better designed with the correctness feedback strategy. This result matches well with Shute's conclusion in traditional teaching feedback strategies (125): that when the task is simple, memory-based, and non-physical, the best performance is obtained with simple feedback features: correct solution, computer-delivered, and goal setting.

A number of interesting directions are possible for future work. It will be interesting to investigate further variations of feedback strategies and as well to apply the strategies to other types of tasks involving more complex object manipulation where task completion involves more complex physical movements instead of only memorization of concepts or properties. Furthermore, adaptation of the feedback strategies to the progress achieved during long-term tasks is an important area to be investigated.

4.6 Conclusion

This chapter presents a VR training system featuring an autonomous virtual trainer able to interact with users and provide feedback during task execution. The effectiveness of two feedback strategies (CF and SF) was evaluated. Although no statistically significant difference has been identified between CF and SF with respect to performance outcome, CF was quantitatively found to lead to higher overall scores used to measure task outcome. CF also showed to be the superior design in terms of being more efficient time-wise and being the preferred strategy of the users.

Our findings also support that in general interactions with virtual trainers were rated by users as comparable in preference to hypothetically performing the same task with real interactions. These results match with previous work suggesting the effectiveness of interactions with virtual humans. Overall our evaluations indicate that our system was well designed and incorporates effective strategies for interaction with users. We believe our results can find several applications in a number of interactive scenarios and applications.

Chapter 5

Learning Collaborative Multi-Agent Manipulation Tasks

In the previous chapters we focused on the effects of agent gender, appearance and feedback strategies on user performance and preference. From this chapter, we shift our focus to the problem of agent training. Our end goal is to train agents that can assist other agents or humans on different problems. With that goal in mind, we focus on reinforcement learning approaches for collaborative tasks. In particular, we focus on multi-agent problems, i.e. those involving more than one agent and on multi-phase tasks, i.e. those involving more than one phase where different phases have different objectives.

5.1 Introduction

In recent years deep Reinforcement Learning (RL) approaches have shown great potential in training control policies used in gaming and robotics. In particular, a family of actor critic algorithms has been developed for tackling tasks with complex continuous action spaces which have become widely used in the field of deep RL. Extensive research and development have been dedicated to such methods, which include Deep Deterministic Policy Gradient (DDPG) (85), Proximal Policy Optimization (PPO) (119), and Soft Actor Critic (SAC) (54). In this chapter we address the use of SAC for multi-phase collaborative tasks.

Many applications such as gaming and robotics often depend on multiple agents cooperating together to solve a complex task. Some of these tasks are sequential in nature and have distinct phases that are governed by different objectives as the task progresses. While it is generally easy for humans to discover cooperative interaction strategies, training agents to handle such tasks is a fairly complex problem, mostly because of two fundamental challenges. The first is related to building a framework where the agents can interact in a collaborative manner, and the second pertains to the training of the agents allowing them to take appropriate actions while intelligently cooperating with each other in order to achieve a common task objective.

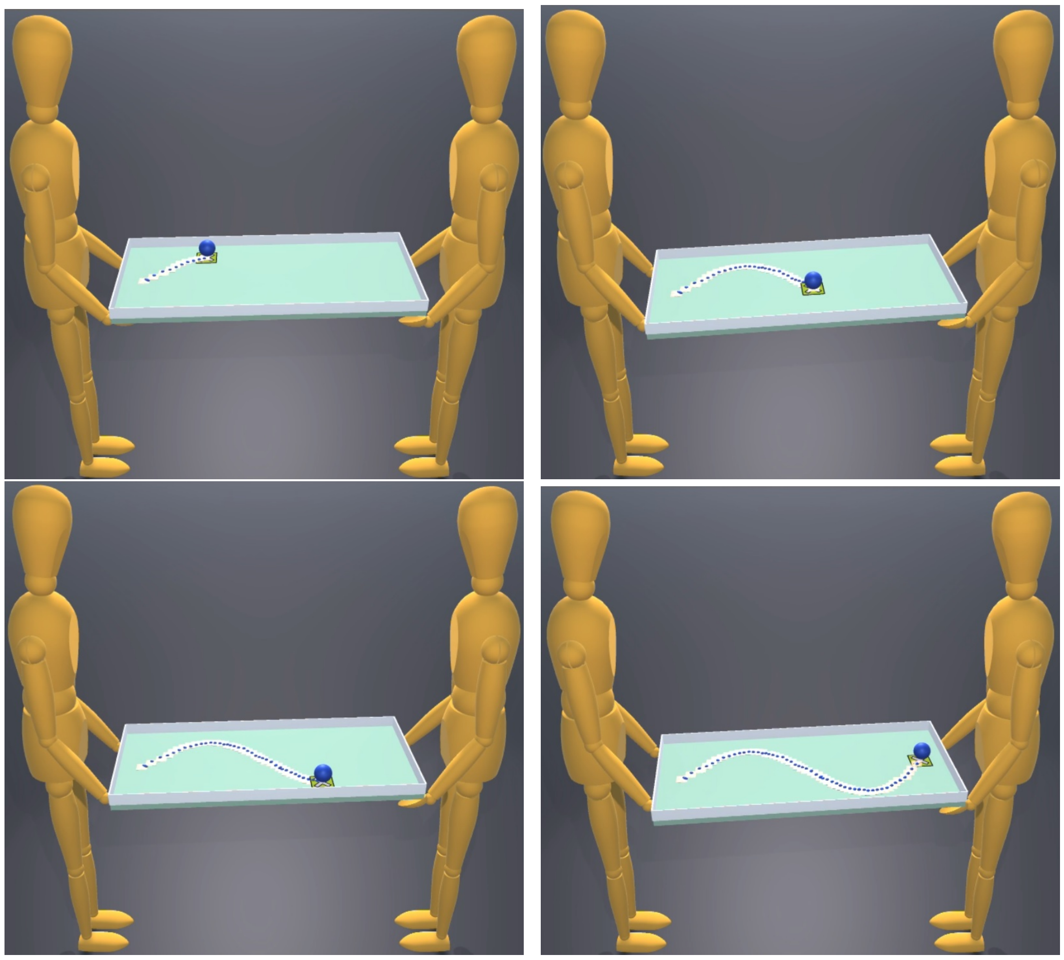


Figure 5.1: Agents collaboratively balancing a tray for a ball to follow a moving target.

With respect to the first challenge, there is a rich collection of available frameworks for experimenting with single-agent reinforcement learning algorithms, such as OpenAI Gym (26), RoboGym(99), ML-Agents (67), and Atari games (21). These frameworks have been used for controlling both simple objects and complex human-like characters or robots, as well as for learning policies in an end-to-end manner. There are also frameworks developed for specific purposes. Examples include controlling an agent to play Go at a world-class level (90), controlling a robotic arm to grasp or manipulate objects (9), and simulating physical characters to perform realistic human-level skills (104). However, building a framework for multi-agent problems is significantly more challenging because the behavior of one agent is affected by the other agent, requiring synchronizing movements in the same dynamic environment and designing a reward function that incentivizes coordination or competition according to possibly different objectives.

With respect to the second challenge, single-agent RL approaches have already been extended to the multi-agent domain for solving tasks that need multi-agent interactions, such as for two-player competitive sports (139) and cooperative agent communication (85; 64; 152). In such domains, the centralized training and decentralized execution (CTDE) paradigm, and its integration with DDPG (80), called MADDPG (85), has been widely used. In this paradigm, agents are trained using centralized information but execute separate policies in a decentralized manner. We adopt a similar architecture in our work but rely on the SAC learning method (54) because of its superior ability to solve tasks in continuous action spaces over other actor critic learning methods.

While CTDE can learn both cooperative and competitive strategies, directly using it to solve a complex task that can be decomposed into multiple phases is not straightforward due to the complexity of designing a proper multi-objective reward function. A common solution is to use hierarchical RL controllers to solve each sub-objective separately with lower-level controllers, and then train an upper-level controller to provide intermediate goals for the lower-level ones (92; 38). However, implementing multi-level controllers introduces complexity to the architecture and requires additional hyperparameter tuning.

To address these issues, we adopt the safe RL concept from Xu et al. (145). The original idea behind safe RL is to define task constraints for the purpose of safety and stability. However, constraints can also be leveraged to model the different sub-objectives of a task which can be decomposed into sequential sub-tasks or phases. In this chapter we propose to treat all sub-objectives except the final one as constraints inside a multi-agent learning framework, and optimize the final objective only if none of the constraints' objectives are violated for each agent.

We define a tray balancing task (Figure 1) in order to train and evaluate the proposed training approach. To solve it, two agents need to cooperate to control the position and orientation of a tray. They need to control it precisely so that a ball rolling on the surface of the tray can follow a pre-defined target trajectory. We define two phases: the objective of the first phase is for the agents to lift the tray appropriately, and the objective of the second phase is to control the tray in order to make the ball follow its target trajectory. We study the performance of our model by analyzing the objective rewards for different trajectories. We also evaluate the model on trajectories that are unseen during training in order to study the generalization capability of our method, and we analyze our model's robustness by evaluating its performance in the presence of disturbance forces.

Overall, our contributions to this work can be summarized as follows:

- we develop a multi-agent framework able to physically simulate collaborative tasks in a shared environment,
- we propose a constraint-based algorithm to learn policies for multi-agent cooperative tasks with sequential objectives, and
- we evaluate the trained policies with tray balance tasks in order to study their performance, generalization, and robustness.

5.2 Related Work

5.2.1 Multi-Agent Environment

Building multi-agent environments is very challenging since the reward design has to not only consider task objectives but also interaction strategies between agents. Works on multi-agent learning have addressed control and communication strategies between 2D agents (85), and also emergent strategies during learning (76; 15). There is however a lack of frameworks for experimentation with human-like behaviors achieved from joint-level control. Recently, one work has created such an environment for training human-like characters to compete with each other in sports games like fencing and boxing (139). While these environments are task-focused, our environment expands the possibilities for accomplishing cooperative tasks that include human upper-body and arm movement.

5.2.2 Multi-Agent Reinforcement Learning

The multi-agent learning problem is challenging because of difficulties with agent scalability, non-stationarity of the environment, and non-unique learning objectives. A straightforward approach to solving multi-agent control problems is to train a joint action policy for all agents using a single-agent RL algorithm, or directly extend single-agent RL algorithms where each agent learns independently by considering other agents as part of the environment, such as by independent Q-learning (129). However, the above solutions suffer from scalability and stability issues. Lowe et al. (85) proposed a parameter sharing approach to tackle this problem, called centralized training and decentralized execution approach (CTDE). This is extended from the actor-critic framework, where each agent uses a centralized critic to access all agents' observations and action parameters so that it can learn an approximate model of the other agents' online policies within a stationary environment. The learned policy only uses local information so it can be used by each agent without further communication between agents. Thereafter, additional algorithms have been developed based on the CTDE framework in order to address different aspects of typical multi-agent systems, for example, credit assignment with integration of the independent actor-critic method (39), scalability by adding mean field Q learning and actor-critic learning (148), stability with SAC learning (86), and incorporation of attention with SAC learning (64) in order to enable faster and more stable learning (110).

Our work also adopts the concept of improving the stability in multi-agent learning (86) by integrating SAC (54) into the CTDE framework. However, in comparison to other recent approaches (86; 30; 140), we study the applicability of CTDE with SAC in a complex human coordination task that is decomposed into multiple phases. As evaluation (see Section 5.6) we use CTDE with SAC as a baseline to evaluate our proposed constraint-based learning model.

5.2.3 Multi-Objective Learning

Multi-objective learning involves learning tasks that have two or more objectives to optimize. It has the lifelong learning properties (29), which means an agent can be trained on a sequence of relatively easy tasks to gain experience and develop a more informative measure of reward, which can then be leveraged when performing harder tasks. Complex tasks that need to be broken down into sub-tasks based on their sub-objectives are quite common. When designing the objectives of a given task, the sub-objectives can be defined as separate objectives without connection but sharing the same action space, like a robot arm picking and placing objects in different boxes (62), agents working together to push different objects into different locations (147). Sub-objectives can also be defined as sequential objectives, such that in order to finish the final objective, the previous objectives have to be completed first, which is aligned with our task design. Example tasks solved in this manner are: an agent moving to a target location while moving objects or obstacles along the moving path (93), a biped character walking and needing to maintain natural gait motion with a walking target (105), and a four-legged robot walking to a target location (92). While most of existing works focus on optimizing multi-objective learning for a single agent, in this chapter, we extend the approach to a multi-agent multi-task setup, where two virtual agents need to control the force applied on each side of a tray in order to solve a cooperative tray balancing problem.

5.2.4 Reinforcement Learning with Constraints

Learning while considering constraints is a popular approach used in safe reinforcement learning (42), where the focus is on preventing the agent from taking actions or entering states that are too risky, since ensuring safety is always critical when the learned strategies need to be deployed to real world systems. Different ways of specifying constraints have been proposed by previous researchers, including using constraints on the expected return or cumulative costs (4; 145), and defining regions of the state space that will result in agent failure. In our work, we focus on addressing a new formulation for a cooperative task that involves two sequential objectives, and we propose to define our constraint as a threshold on the expected return from the first objective. The learning process will advance to optimize the final objective only when the constraint has been satisfied.

5.3 Overview

We are interested in solving multi-agent collaborative tasks in a physically-simulated environment, with focus on a tray balancing task. In this section we provide details of our task design. The overall framework used to train and evaluate our method is illustrated in Figure 5.2.

5.3.1 Environment and Tasks

The environment is simulated inside the physics simulator of the Unity game engine, and it includes two virtual agents, a ball, a moving target, and a tray with four anchor points at each corner. We consider two trajectories for the tray balancing task: an ellipse trajectory and an S-curve trajectory as shown in figures 5.3 and 5.4. The details of these trajectories are described in Section 5.6.1. In order to complete the task, the two agents start with a standing initial pose along the two sides of the tray, and then outstretch their arms to reach the two anchor points on their side of the tray. The arms of the humanoids are controlled by inverse kinematics (IK) (74), the end effectors are their hands' position and rotation, and the two anchor points' position and rotation are their desired targets. After reaching the anchor points, we require the agents to perform the tray balancing task in a sequential manner in 2 phases: in phase 1, they need to lift up the tray to a fixed height while maintaining the balance of the tray, and in phase 2, they need to manipulate the tray to guide the ball to follow the target on a moving trajectory. The task process inside the physics simulator in Figure 5.2 illustrates this process.

5.3.2 Framework

Our framework has three main components, the physics simulator, the MARL (Multi-Agent Reinforcement Learning) controller, and the IK (Inverse Kinematic) controller. The physics simulator is used to simulate different task environments, the MARL controller is the Python API that runs our multi-agent learning approaches during both the training and execution processes, and the IK controller is used to control the humanoid arms during both processes. At each learning step, the MARL controller receives the states and rewards information and sends out the actions to the physics simulator via the communication channels provided by the Unity ML-Agents plugin (68). In the meantime, the MARL controller collects the anchor points' position and rotation information and sends them to the IK controller as its desired control target for the agents' arm movement.

5.4 Approach

5.4.1 Problem Formulation

We formulate our problem as a two-agent cooperative game, modeled as a partially observable Markov Decision Process (MDP); our approach can be easily extended to more agents. At each time step $t \in [0, T]$, the MDP tuple is defined as $\{\mathcal{O}, \mathcal{S}_i^t, \mathcal{A}_i^t, \mathcal{S}_i^{t+1}, \mathcal{R}_i^t, \mathcal{T}\}$, where T is the max steps of each episode, $i \in \{0, 1\}$ is the index of the agent. Agent i observes partial state $s_i^t \in \mathcal{S}_i^t$ from environment $\mathcal{O}$, takes action $a_i^t \in \mathcal{A}_i^t$ that leads to a new state according to the dynamics model $\mathcal{T}$, and receives a scalar reward signal from its reward function $r_i^t = \mathcal{R}(s_i^t, a_i^t)$, and $r_i^t \in [0, 1]$.

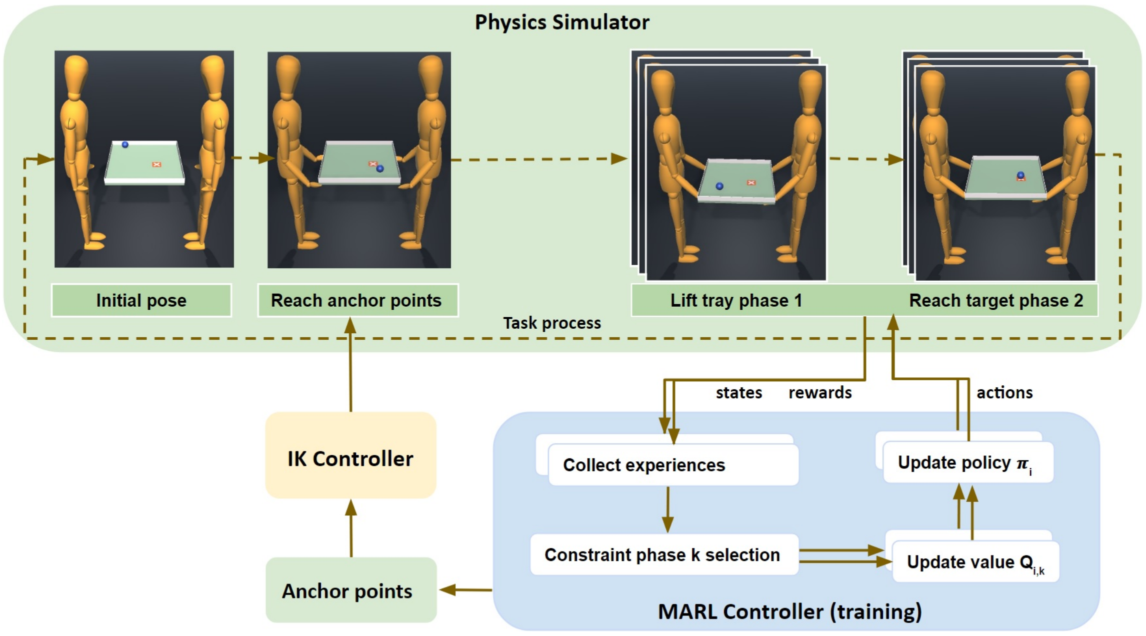


Figure 5.2: The overview of our environment and framework.

For each agent i, the goal is to find an optimal policy $\pi_{\theta_i}(a|s)$ that maximizes its expected accumulated reward J_i, where θ_i denotes the parameters of the policy.

5.4.2 Multi-Agent Soft Actor Critic Learning (MSAC)

In our multi-agent setup, we adopt the CTDE framework incorporated with SAC algorithm (54) because its extra entropy term increases the policy's exploration ability and robustness, and we name it Multi-agent Soft Actor Critic algorithm (MSAC). During training, we jointly conduct policy evaluation and policy iteration, where for each agent i we concurrently learn a stochastic policy π_{θ_i} and two Q-functions with parameters $\phi_{i,j}$, j is identifier of Q function, in the range of $[1, 2]$.

During policy iteration, we use Eq 5.1 to optimize, in the direction of maximizing the accumulated reward, the policy function $J(\theta_i)$:

$$\nabla_{\theta_i} J(\theta_i) = \nabla_{\theta_i} \frac{1}{|\mathcal{B}_i|} \sum_{\mathcal{B}_i} \{\min_{j=1,2} Q_{\phi_{i,j}}(\mathbf{s}^t, \mathbf{a}^t) - \alpha \log \pi_{\theta_i}(a_i^t|s_i^t)\}. \tag{5.1}$$

The policy function for each agent is parameterized as θ_i, $\mathcal{B}_i$ denotes a batch of experiences sampled from the replay buffer, $\log \pi_{\theta_i}(a_i^t|s_i^t)$ is the entropy term to increase the variation of action space, and α is a learned variable indicating the contribution of an entropy regularization term. $Q_{\phi_{i,j}}(\mathbf{s}^t, \mathbf{a}^t)$ is calculated from the states $\mathbf{s}$ and actions $\mathbf{a}$ of all agents, as a centralized evaluation operation.

During policy evaluation, we optimize the loss function $\mathcal{L}(\phi_{i,j})$ using Eq 5.2 to evaluate the learned policies by minimizing the difference of the value function $Q_{\phi_{i,j}}(\mathbf{s}^t, \mathbf{a}^t)$ and its target value y_i for each agent:

$$\nabla_{\phi_{i,j}}\mathcal{L}(\phi_{i,j}) = \nabla_{\phi_{i,j}}\frac{1}{|\mathcal{B}_i|}\sum_{\mathcal{B}_i}(Q_{\phi_{i,j}}(\mathbf{s}^t,\mathbf{a}^t) - y_i)^2. \tag{5.2}$$

The target value y_i is calculated with a separate target value network $Q_{target}(s^{t+1}, a^{t+1})$ in order to maintain stability when updating policy network, as shown in Eq 5.3:

$$y_i = r_i + \gamma\{\min_{j=1,2} Q_{\phi_{i,j,target}}(\mathbf{s}^{t+1},\mathbf{a}^{t+1}) - \alpha\log\pi_{\theta_i}(a_i^{t+1}|s_i^{t+1})\}. \tag{5.3}$$

Both value functions $Q_{\phi_{i,j}}$ and their target value functions $Q_{\phi_{i,j,target}}$ share the same network structure, and they are parameterized as ϕ_i and $\phi_{i,target}$ separately.

5.4.3 MSAC with Constraints

Constraints used in safe RL problems can be categorized into primal and primal-dual approaches. Primal approaches focus on the design of objective functions and do not need Lagrange multipliers as dual variables during the optimization process, thus simplifying the implementation process and reducing training time (42). Due to its proven global convergence, we adopt the primal approach (145) in our framework and call it Constrained Multi-agent Soft Actor Critic (C-MSAC).

We assume the task can be divided into n sequential phases; we consider objectives in the first $n-1$ phases as constraints while the n^{th} phase objective determines the final objective. The idea is to optimize the final objective function with reward accumulated from the n^{th} phase while guaranteeing that all constrained objectives from previous phases can be satisfied. Our multi-agent problem with constraints can thus be formalized as the following optimization problem:

$$\max_{\theta_i} J_{i,n}(\theta_i), s.t. J_{i,k}(\theta_i) >= d_{i,k}, k < n, \tag{5.4}$$

where $J_{i,k}$ denotes the total expected return of the k-th constraint phase, and $d_{i,k}$ is a threshold that is calculated from a Monte Carlo estimate return with step reward $r_0 = 0.9$, and discount parameter $\gamma = 0.99$ for the k-th constraint:

$$d_{i,k} = \sum_{t=0}^{T-1}\gamma^t r_0. \tag{5.5}$$

Hence, when all constraint phases reach their respective thresholds $d_{i,k}$, we find the optimal policy $\pi_{\theta,i}$ for each agent. For each phase k, we learn a separate value function $Q_{i,k}$, that is, we learn n Q functions for each agent, and during each optimization step, one Q function will be optimized based on Eq 5.7. In this section, we use $Q_{i,k}$ as the shortened version of $Q_{\phi_{i,j,k}}$ for simplicity.

At phase k, the expected return $J_{i,k}$ is calculated as:

$$J_{i,k}(\theta_i) = \frac{1}{|m|}\sum_{s_i^0,a_i^0\sim\xi,\pi_{\theta_i}}^{m}\rho_{i,k}Q_{i,k}(s_i^0,a_i^0). \tag{5.6}$$

The term $Q_{i,k}(s_i^0, a_i^0)$ computes the accumulated return of a rollout starting from (s_i^0, a_i^0) while following policy π_{θ_i}. State s_i^0 denotes an initial state that is sampled uniformly from an initial distribution ξ and a_i^0 is the corresponding action based on the latest policy π_{θ_i}. $\rho_{i,k}$ is a weight ratio for each rollout, where a value of 1 is used in our experiments. We average $m = 40$ rollouts to calculate $J_{i,k}$.

To combine the constraints with policy iteration and policy evaluation, we integrate the constraint phases into the MARL process. At each optimization step, we first choose a phase k to optimize by selecting a Q function using Eq 5.7:

$$select(Q) = \begin{cases} Q_{i,k} & \text{if } \exists k < n : J_{i,k}(\theta_i) < d_{i,k}, \\ Q_{i,n} & \text{else } k = n. \end{cases} \tag{5.7}$$

To perform policy iteration, we then replace $Q_{\phi_{i,j}}$ in Eq 5.1 with the chosen value function $Q_{i,k}$. For policy evaluation, we replace $Q_{\phi_{i,j}}$ in Eq 5.2 with the value function $Q_{i,k}$ of the chosen constrained phase, leading to the following critic update:

$$\nabla_\phi \mathcal{L}(\phi_{i,k}) = \nabla_\phi \frac{1}{|\mathcal{B}_i|} \sum_{\mathcal{B}_i} (select(Q) - y_{i,k})^2, \tag{5.8}$$

where $y_{i,k}$ is the target value calculated in the same way as in Eq 5.3, but specifically for the ith agent at phase k. If multiple constrained phases are not satisfied, we can choose any phase to maximize its expected return.

5.4.4 Algorithm

Algorithm 1 summarizes the procedure of our proposed approach C-MSAC during the training process. Lines 4-9 show the process of collecting samples from multiple environments for both agents in order to enrich the training dataset and speed up the training. Lines 10-17 perform the learning process in that we iteratively select a phase k to optimize until all phases have been saturated with their own thresholds based on Eq 5.4.

5.5 Training

5.5.1 State and Action Representation

In our environment, the state information includes the tray, the ball, the moving target, and the two anchor points for each agent. Let p, q, v, qv denote position, rotation, velocity, and angular velocity respectively. The state is represented as $s_i = [s^B, s^T, s^t, s^{Bt}, s^{Tf}]$, where $s^B = [p_B, q_B, v_B, qv_B]$ represents the state of the ball, $s^T = [p_T, q_T, v_T, qv_T]$ represents the state of the tray, s^t is the position of the moving target, s^{Bt} contains the Euclidean and quaternion distance between the ball and

Algorithm 1 Constrained Multi-agent Soft Actor Critic Algorithm (C-MSAC)

```
1: Initialize policy network $\theta_i$, value estimation networks $\phi_{1,i}$, $\phi_{2,i}$ with random
   weights for each agent;
2: Initialize replay buffer $\mathcal{D}$ as empty dictionary ;
3: for each step do
4:     for each environment m do
5:         Sample action $a^t$ from policy $\pi(a^t_{m,i} \middle| s^t_{m,i})$;
6:         Proceed one step in the environment;
7:         Observe reward $r^t_{m,i,k}$ and next state $s^{t+1}_{m,i}$ from environment;
8:         Concatenate all data into tuple ($s^t_m$, $a^t_m$, $r^t_{m,k}$, $s^{t+1}_m$) and send to the replay
   buffer $\mathcal{D}$;
9:     end for
10:    if step > batch_size then
11:        Sample a group $s^0_i$ from $\xi_i$;
12:        Calculate total return $J_{i,k}$ with Eq 5.6 for all phases $k$ ;
13:        Sample a batch data from $\mathcal{D}$ ;
14:        Select a phase $k$ to optimize with Eq 5.7 ;
15:        Evaluate policy with Eq 5.8 ;
16:        Optimize policy with Eq 5.1
17:    end if
18: end for
19: Output optimal policy $\theta_i$ for agent $i$.
```

the moving target that the ball needs to follow for, and s^{Tf} includes the Euclidean distance between the tray and a fixed target position that is slightly higher than the tray's initial position.

The action a^k_i in our task is defined as the force applied at each holding point on the tray in x, y, z direction. It is calculated by multiplying a fixed scalar by the normalized [-1,1] control signals learned from the RL policy. We use 100 as the scaling constant.

5.5.2 Reward Design

The tray balance task includes two phases: lifting up the tray and reaching the target, we need to design reward functions for each phase based on its own objective, the details are described below.

Lifting the Tray

In this phase, the agent's objective is to lift up the tray to the fixed target position p_0 while maintaining its balance relative to the identity quaternion q_0. To satisfy this objective, the reward function is designed as:

$$r_{lift} = 0.8 * r_{dist} + 0.2 * r_{ang}.$$

Here r_{dist} is the Euclidean distance between the tray and the fixed target position, to encourage the tray to move closer to the fixed target position, and it is calculated:

$$r_{dist} = \exp[-5||p_T - p_0||^2].$$

r_{ang} is the quaternion orientation difference between the tray and the identity quaternion, to encourage the tray to balance itself, it is calculated:

$$r_{ang} = \exp[-20(1 - ||q_T \ominus q_0||^2)].$$

Reaching the Target

In this phase, the agent's objective is to adjust the tray position and orientation to control the ball to follow a moving target on the tray. To encourage the ball to stay on the moving target, we use the Euclidean distance between the ball and the moving target, as represented in the reward function:

$$r_{target} = \exp[-25||p_B - s^t||^2].$$

The target's moving path is controlled by curve equations. We use two different curves: an ellipse (where the major and minor axes are randomized at the beginning of each training episode) and an S-curve which is defined by a cubic equation.

The agents have to collaborate to finish both phases' objectives, leading to a Nash equilibrium (100), meaning that each agent is taking their best policy to respond to the other agent, and its gains will be undermined if a different policy is taken. Considering the equilibrium, the reward we use for each agent in each phase is the shared team average reward, and calculated as the average of both agents in that phase (136): $r = \frac{1}{N}\sum_{i \in N}(r_i)$.

5.5.3 Early Termination

Early termination (104; 12) is a common technique used in reinforcement learning, to help RL agents stop learning bad behaviors and favor the collected samples more efficiently, thus achieving significantly faster learning. In our task, we introduced three early termination conditions to terminate the current episode during training:

- first, when the ball touches the edge of the tray, because it can easily get stuck in the corner of the tray;
- second, when the ball or tray drops on the ground, as it will never get recovered from this state;
- third, when the height of any anchor point goes below a threshold (0.85m in our experiments), the agent's hands will not be able to reach them with IK solution).

5.5.4 Training Details

Network Architecture

For each agent, we use the same network structure as SAC (54), but add $(n-1)$ value networks and target value networks to represent the constraint objectives of each phase. All networks consist of three fully connected layers with 256 hidden units in the first two layers, and Relu is used as the activation function. The last layer outputs the mean and log value of the policy. The learning rate is 0.0003, while the gamma is 0.99.

Multi-Environment Training

One of the learning thresholds for an RL approach is sample efficiency. To speed up the training we implement our environment as a multi-environment setup allowing us to collect 4 times training data per frame rate.

5.6 Experiments and Results

5.6.1 Target Trajectories

In our environment, the xoz and $y-up$ coordinate system is being used. We train both models on two types of parametric trajectories, as shown in Figure 5.3 and Figure 5.4:

1) *Randomized ellipse*: described with the ellipsoid equation:

$$f(t_i) = (a\cos(\theta_1(t_i))\cos(\theta_2(t_i)),\, b\sin(\theta_1(t_i)),\, b\cos(\theta_1(t_i))\sin(\theta_2(t_i)),$$

where θ_1 is the longitude angle change of meridian plane xoy in radian, θ_2 is the latitude angle change of the equatorial plane xoz in radian. a, b are the major axis and minor axis in the range of $[0.1, 0.3]$ for the ellipse shape randomization. The time step term t_i is the ratio of the current training step and total steps for each episode.

2) S-curve: described as a cubic Bèzier curve with control points P_0, P_1, P_2, and P_3.

$$f(t_i) = (1-t_i)^3 P_0 + 3(1-t_i)^2 t P_1 + 3(1-t_i)t^2 P_2 + t_i^3 P3.$$

In our task, we fixed points P_0 and P_3, and adjusted P_1, P_2 to get the desired S-shaped curve.

For the evaluation of unseen trajectories, we use two additional types of curves: square and triangle, as shown in Figure 5.5 and Figure 5.6, they are generated by connecting three(triangle) and four (square) fixed positions relative to the position of the tray.

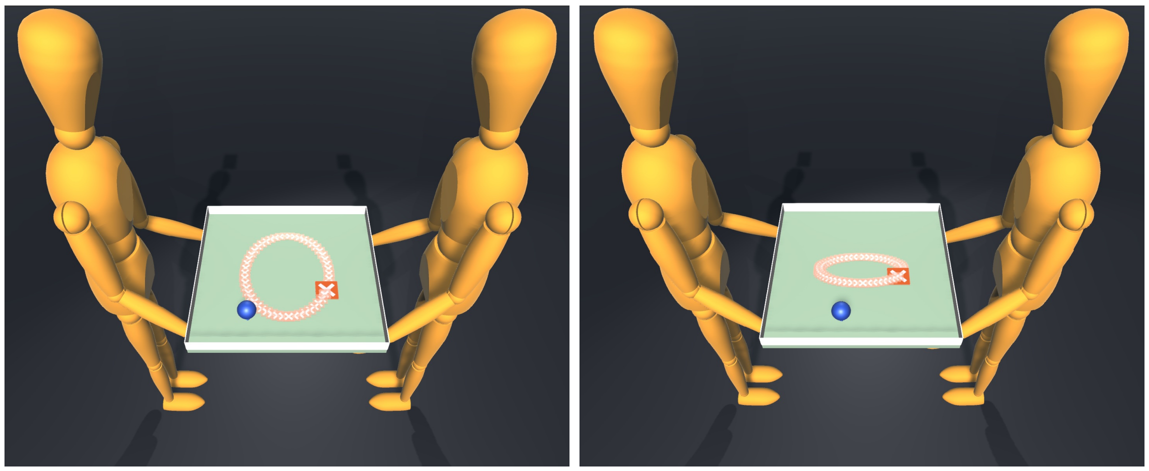


Figure 5.3: Ellipse examples, left with equal a and b, right with non-equal a and b

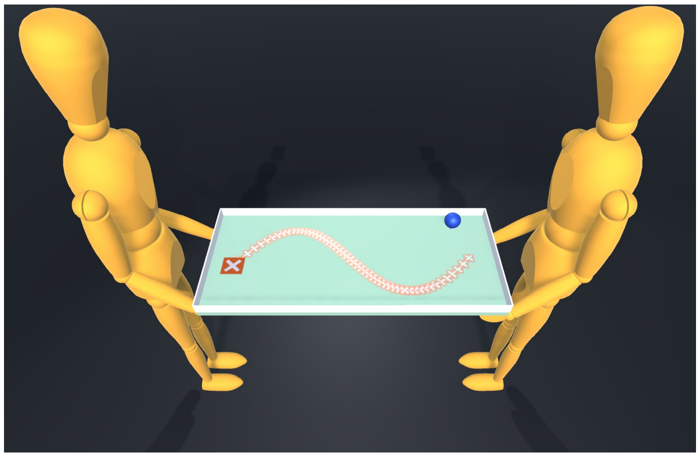


Figure 5.4: S-curve example.

5.6.2 Evaluation Metrics

We compare our proposed approach of multi-agent soft actor critic with constraints (C-MSAC) to MSAC approach. For MSAC we used a linear combination of rewards from each phase to calculate the total reward for each agent:

$$r_i = 0.85 r_{lift} + 0.16 r_{target} + (-0.1) r_{time}$$

where the optimization process is calculated according to Section 5.4.1. For C-MSAC each phase is optimized separately based on the reward accumulated from this phase as discussed in Section 5.4.3.

We compare both models on three criteria:

- **Mean target reward and on-target performance**: where we evaluate the models on the same family of trajectories that were used to train the models.

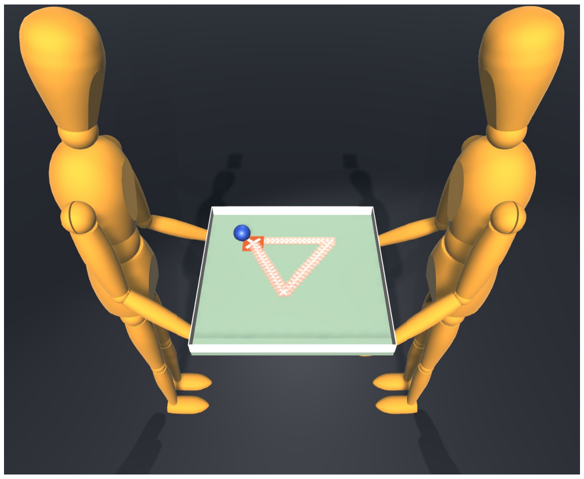


Figure 5.5: Generalization to unseen triangle trajectories.

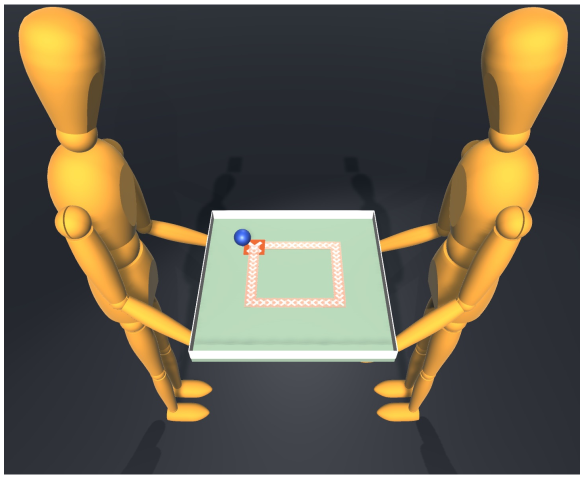


Figure 5.6: Generalization to unseen square trajectories.

- **Generalization ability**: where we evaluate the models on unseen trajectories.
- **Robustness**: where we introduce extra disturbances to the environment and analyze the ability of the model to restore balance.

5.6.3 Results

Mean Target Reward

We train both MSAC and C-MSAC for 45,000 episodes each on the randomized ellipse trajectory and the S-curve trajectory tasks. After every 2,000 episodes of training, we run validation on 100 episodes to measure the target reward (r_{target}) on the objective phase. Figure 5.7 and Figure 5.8 show the mean and standard deviation of the normalized reward for MSAC and C-MSAC on the ellipse and S-curve trajectories.

For the ellipse trajectory, the C-MSAC model starts off slightly lower than the MSAC model in terms of target reward. This makes sense since the C-MSAC model initially focuses on the tray lifting constraint objective. Eventually, the C-MSAC model surpasses MSAC yielding a significantly higher target reward. Furthermore, C-MSAC is much more stable, exhibiting similar performance across different evaluation trials as compared to MSAC's performance which is characterized by large variance.

For the S-curve trajectory, both models have a similar target reward in the initial stages of training. However, in the later stages, the C-MSAC model surpasses the MSAC model yielding a policy very close to generating the maximum possible normalized reward (1.0).

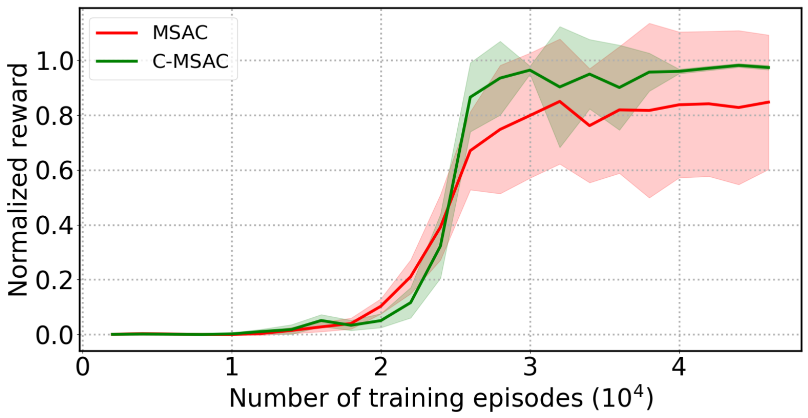


Figure 5.7: Normalized reward on test episodes for the target objective phase (phase 2) with ellipse trajectory.

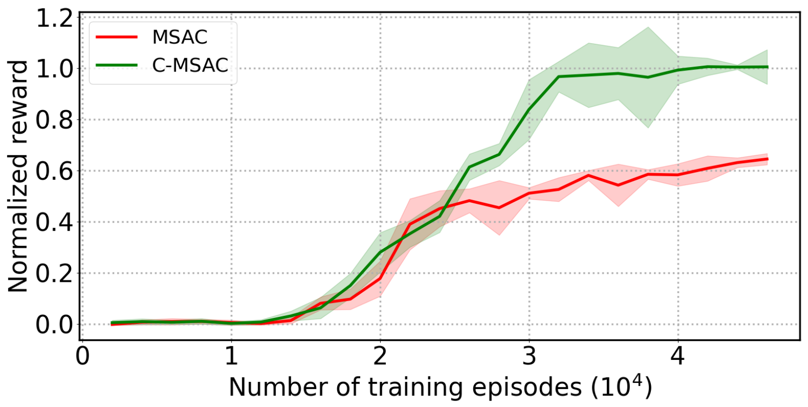


Figure 5.8: Normalized reward on test episodes for the target objective phase (phase 2) with S-curve trajectory.

Phase Analysis

We also measured the target reward for the tray lifting constraint phase (phase 1) for both models on the 100 test episodes. Figure 5.9 and Figure 5.10 show the mean and standard deviation of the normalized constraint reward r_{lift} for both models on the ellipse and S-curve trajectories. The trend here is fairly similar to that for the objective phase reward. In the initial stages of training, the constraint phase reward is almost the same for the MSAC and the C-MSAC models. As the training progresses, the constraint phase reward for the C-MSAC model increases more rapidly and exhibits a smaller variance compared to the MSAC model. Towards the middle of the training process, as the C-MSAC model starts exceeding the threshold value for the constraint phase, the corresponding reward starts to saturate.

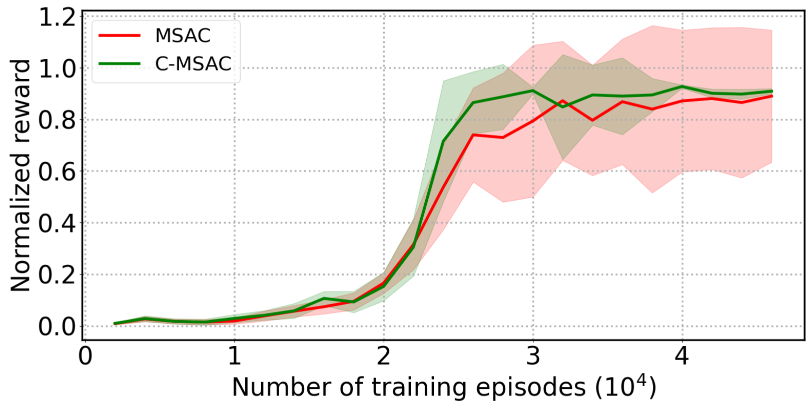


Figure 5.9: Normalized reward on test episodes for the constraint phase (phase 1) with ellipse trajectory.

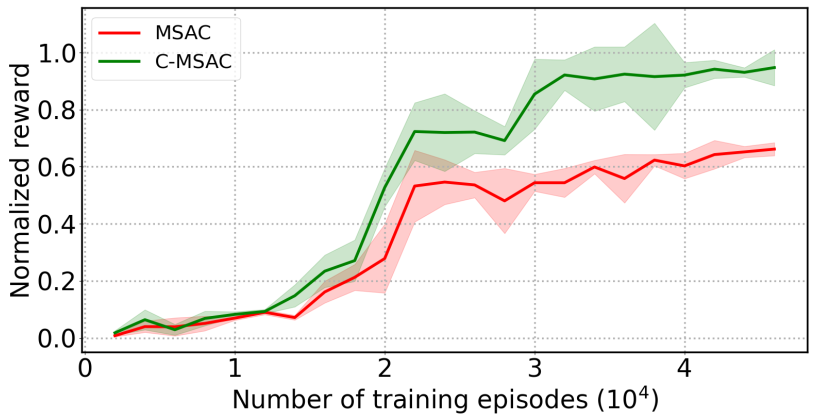


Figure 5.10: Normalized reward on test episodes for the constraint phase (phase 1) with S-curve trajectory.

On-target Performance

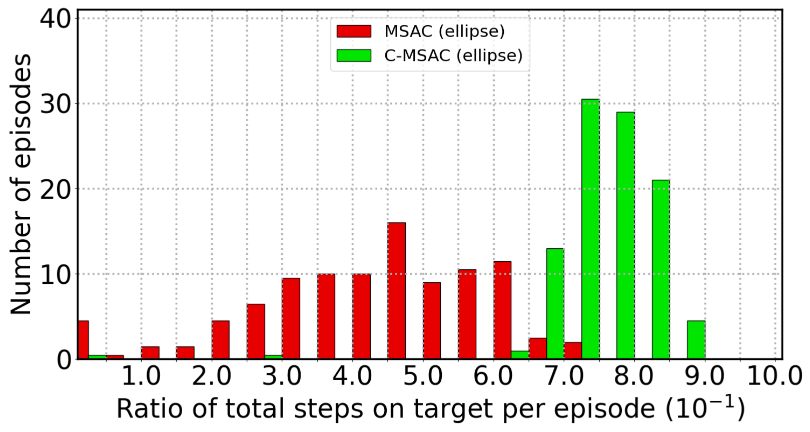


Figure 5.11: Histogram of on-target steps ratio with ellipse trajectory.

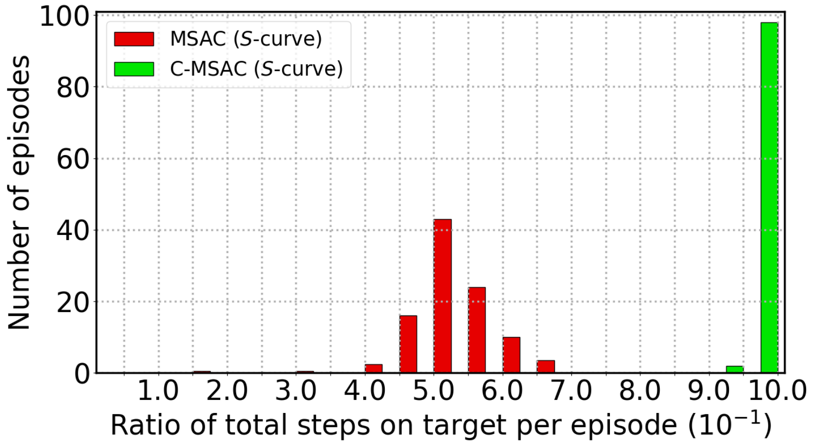


Figure 5.12: Histogram of on-target steps ratio with S-curve trajectory.

In addition to the target reward, we also measured the on-target performance for both the MSAC and C-MSAC models for the ellipse and the S-curve trajectories. We define the ball to be on-target for a given step if the target reward (r_{target}) is higher than 0.95, and then measure the percentage of steps for which the ball stays on target for a given episode. To quantify the distribution of the on-target percentage, we draw a histogram of the number of episodes against the percentage of on-target steps for a given episode. Figure 5.11 and Figure 5.12 show the histograms for the ellipse and the S-curve trajectories, respectively.

It is clear from the figure that the average on-target time for the C-MSAC model is much higher than that for the MSAC model for both the ellipse and S-curve trajectories. This shows that the C-MSAC model is able to learn the fine-grained

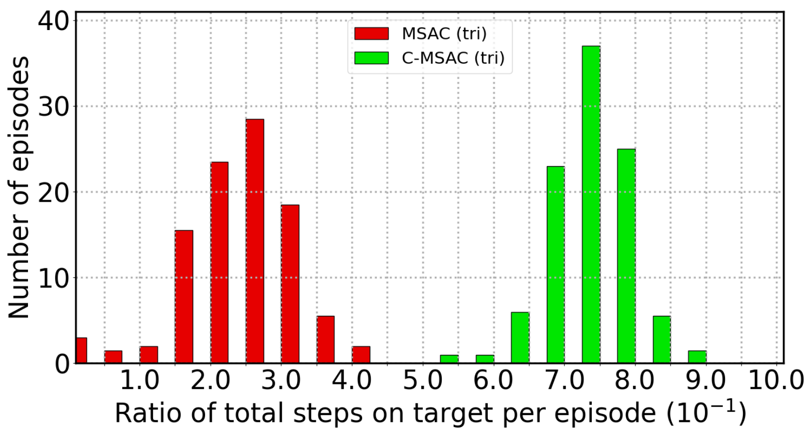


Figure 5.13: Histogram of on-target steps ratio with triangle trajectory.

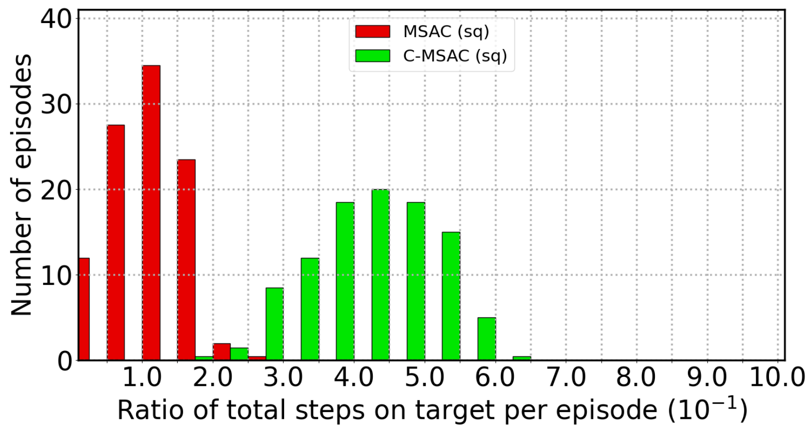


Figure 5.14: Histogram of on-target steps ratio with square trajectory.

controls necessary to keep the ball on target most of the time and follow the curve smoothly. Our supplemental video also demonstrates this capability visually.

Generalization Ability

After every 2,000 steps of training, we also evaluated the MSAC and C-MSAC models (trained on randomized ellipse trajectories) on two types of unseen trajectories: square and triangle. We observe that both MSAC and C-MSAC models are able to generalize to the unseen trajectories; however, the C-MSAC model yields a better average reward (Figures 5.15 and Figures 5.16) and on-target performance (Figure 5.13 and Figure 5.14) in comparison to the MSAC model. This shows that the C-MSAC model also exhibits better ability to generalize in comparison to the MSAC model.

One interesting finding here is that the performance on the square trajectory is slightly worse than that on the triangular trajectory for both models. Upon observation, we noted that this is likely due to the presence of an extra corner on the square compared to the triangle. Both models handle smooth curves and straight lines relatively well, but sometimes have difficulty handling sharp corners. The extra vertex on the square increases the chance of the ball moving far away from the target. We suspect that the difficulty in handling sharp corners might be because of the fact that our training data only includes smooth curves. It might be interesting to include trajectories with corners in the training data in future experiments.

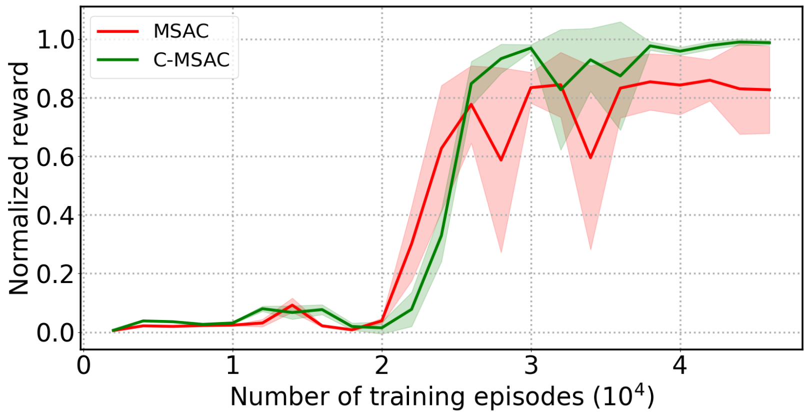


Figure 5.15: Normalized target reward on test episodes with triangle trajectory.

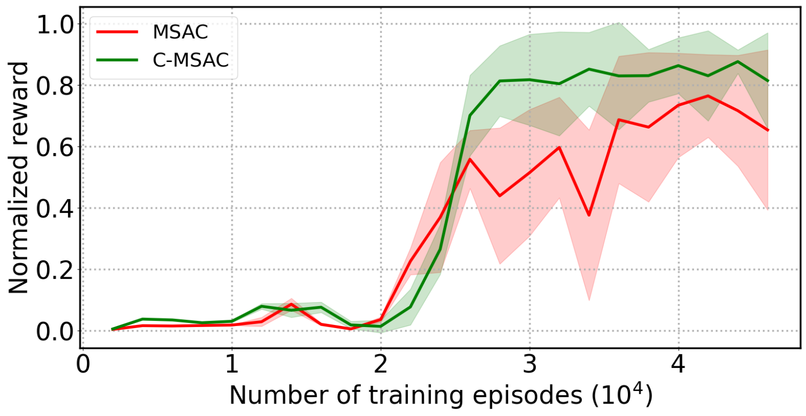


Figure 5.16: Normalized target reward on test episodes with square trajectory.

Robustness

We have studied the robustness of the two models against unexpected disturbances. For some early exploration in this direction, we hit the tray with additional balls at different angles and positions on the tray in order to disrupt the task. From our observations, the C-MSAC model exhibited a superior capability to recover from such unexpected disturbances.

To formalize our observations, we set up an experiment to statistically analyze and compare the robustness of the C-MSAC and MSAC models. In this experiment, we introduce a disturbance force to the agent actions every 10 steps during the test episodes. We sample this force from a Gaussian distribution, scale it up with a magnitude factor, and add it to the agent actions. We evaluate both models on 100 test episodes and obtain the mean and standard deviation of the target reward for different values of the disturbance force magnitude.

Figure 5.17 and Figure 5.18 show the results for the MSAC and C-MSAC models on the ellipse and *S*-curve trajectories. As the force magnitude increases, the target reward for both models reduces and eventually approaches zero. However, the mean reward for the C-MSAC model is consistently higher than that for the MSAC model. The C-MSAC model reward also has a smaller standard deviation compared to the MSAC model, indicating higher stability and consistency across different test episodes.

We note here that we did not apply any disturbance to the environment during training time. We hypothesize that the robustness of these models arises from the entropy term used in the SAC algorithm which encourages exploration.

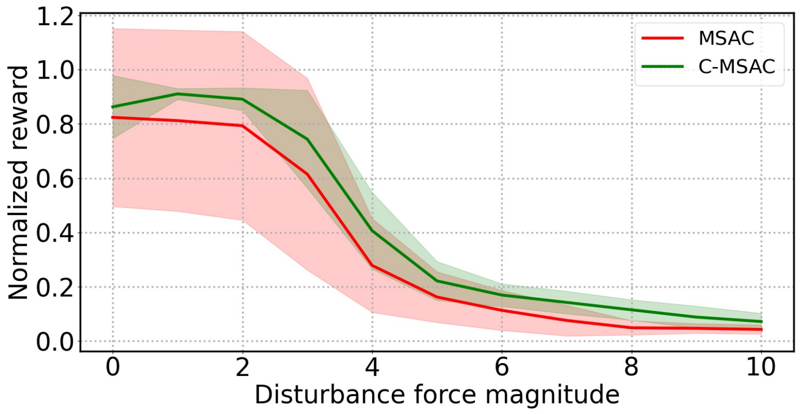


Figure 5.17: Normalized target reward for different disturbance force magnitudes with ellipse trajectory.

5.7 Conclusions

In this chapter, we propose a constraint-based multi-agent reinforcement learning approach to solve multi-phase collaborative tasks. We present a framework to simulate tray balancing and target following tasks with different trajectories. We evaluate the proposed constraint-based (C-MSAC) model on this task and compare it against a baseline that does not employ constraints (MSAC). Our results show that the proposed model is able to exhibit better on-target performance, better generalization ability, and improved robustness in comparison to the baseline.

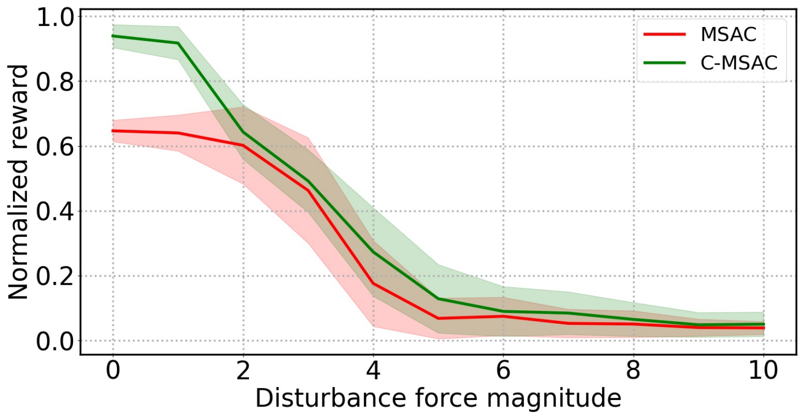


Figure 5.18: Normalized target reward for different disturbance force magnitudes with S-curve trajectory.

Two main limitations can be observed in our present work. While the proposed constraint-based framework is very general and applicable to complex multi-phase tasks, in this work we have only explored a task of two phases: a tray lifting constraint phase and a tray balancing target phase. Additionally, the arm movements in our environment are not controlled by an RL policy but rather calculated by Inverse Kinematics (IK) based on the anchor points transformation. This sometimes results in some glitchy movements, for example when the agents suddenly move the tray to avoid the ball from going out of bounds, resulting in a sudden pose change due to the arm having to follow the anchor points.

In the next chapter, we extend this work by using physically simulated humanoids with joint-level RL-based control for the arms instead of using IK. This provides us the opportunity to fully leverage the power of the constraint-based framework by introducing additional phases to the task such as an initial constraint for the arms to reach the tray anchor points. While we focus on collaborative tasks, our framework can also be generalized to competitive tasks, opening interesting avenues for future work.

Chapter 6

Addressing Realism and Robustness

6.1 Introduction

In the previous chapter, we saw that C-MSAC can be used to train multiple agents to collaborate with each other on complex multi-phase tasks. The results also show that the proposed technique leads to strong statistical results in terms of the mean episode reward, generalizes well to unseen trajectories, and naturally introduces robustness to environmental noise. However, there are two important shortcomings that need to be addressed. First, while the statistical results from this method are strong, simulation videos show that the agent movements could sometimes appear unnatural and oscillatory. This is because the approach relied on inverse kinematics rather than purely physics. Second, since each agent is trained to collaborate with only one other agent, there is a possibility that this agent might fail to generalize against other unseen agents or humans with different behaviors. In this chapter we focus on overcoming these shortcomings.

6.1.1 Physics-based Character Animation

A number of techniques have been proposed for creating realistic character animations using kinematics (49; 10; 146). However, many of these methods are often prone to unnatural movements. In physics-based character animation, characters are modeled as rigid bodies with mass (104; 83). Because the environment enforces the laws of physics, it prevents movements that are physically impossible. While it does not prevent all unrealistic motions, appropriate learning constraints can be used to achieve natural looking motion. In this work, we use C-MSAC with physics-based full arm control for the agents. We introduce an extra phase for the arms to reach the tray contact points, and design our reward function appropriately to incentivize natural arm movements for controlling the tray. We demonstrate that learning under physics also improves the ability of the agents to react to changes in the physical properties of the environment such as the mass of the ball.

6.1.2 Robustness

Robustness has been studied extensively in optimization and optimal control (156; 22). In a game theoretic setting such as a two-player zero-sum game, controlling the uncertainty/disturbances through an adversarial opponent can produce a robust strategy for a player (106). In a reinforcement learning setup, a general solution for the robustness problem is to formulate an adversarial agent for introducing uncertainty in transition probabilities (13; 98), or disturbance to the environment dynamics (106; 91), states (87; 45) or actions (130). Especially, introducing action perturbations to the environment can be a natural approach to simulate the environmental changes to a certain extent.

In our work, the goal is to learn a robust human-like agent under physics that can generalize to collaborate with other agents or humans. To achieve this goal, we propose the Constrained Robust Soft Actor Critic (C-RSAC) learning algorithm based on our previous constrained multi-agent learning approach C-MSAC (123). In this work, we first train two cooperative agents using the C-MSAC framework under physics with joint-level arm control, to adopt two pre-trained policies for each agent. We then modify the C-MSAC multi-agent learning framework to the C-RSAC single-agent learning framework such that one of the agents executes a pre-trained policy with additional noise augmented to its action space, while the other agent takes another pre-trained policy and continuously updates this policy to respond to the first agent's actions. Henceforth in this chapter, we refer to the former as the noisy agent and the latter as the robust agent in this new single-agent learning framework. We also integrate a user interface within our framework to allow manually controlled noise addition. We investigate the performance of C-RSAC on the tray balance task introduced in 5.3.1 with an ellipse trajectory. To evaluate the performance we measure the mean episode reward on test episodes at regular intervals during the training procedure. We show that as the training progresses, the mean episode reward for the robust agent increases which indicates higher robustness compared to the agent trained using just C-MSAC.

6.2 Related Work

6.2.1 Character Animation

Different works have been done for single character animation, such as learning locomotion behaviors (104), learning full-body behaviors from example (104; 144), learning to play basketball (83), learning athletic jumping strategies (150), learning locomotion stepping stone skills (142), learning climbing wall skills (94). Most single character animation work needs to use or learn from motion data for natural behavior simulation. To learn directly from motion data for single character animation is possible, but for a multi-character environment is difficult, as multiple human players have to be involved in this procedure. So for multi-characters simulation, the character's behavior could either be learned separately and then transferred to the multi-character framework for advanced learning (139), or could be directly used in the multi-character environment as in crowd simulation (57). Our work tries to solve a multi-character collaborative task by learning the control policy for both characters' upper body movements at the joint level.

6.2.2 Robust Reinforcement Learning

Robust reinforcement learning problems usually target solving uncertainties/disturbances introduced from different aspects of the Markov Decision Process(MDP) (109), they can be categorized into three directions. First, from the transition model or environment dynamics, the uncertainties of the environment dynamics can be modeled by uncertainty set or direct disturbance to the environment and solved by constructing an adversary agent from the uncertainty sets (13; 98; 155), i.e. Zhang et al. (155) bring a "nature player" into multi-framework to serve as adversary agent to each agent to tackle the uncertainty transition probability. Second, from the environment state, adding noise to the state can trick the agent into wrong beliefs about the environment's current state (141; 103; 44). Third, from the agent action, adding noise to the action can bring the agent in the wrong direction to affect the transition dynamics, i.e. Tessler et al (130) has studied adding an adversary agent into the protagonist agent learning framework by adding disturbances to the protagonist action space directly to simulate the constantly small disturbance from the environment and by adding the adversary agent policy linearly to the protagonist agent's policy to simulate sudden disturbance in the environment. In our work, we choose to add noise to the action, but instead of adding to the robust agent itself, we add to the noisy agent.

6.3 Approach

We formulate our problem as single-agent robust learning for multi-phase collaborative tasks in a multi-agent cooperative Markov game set (82), and we propose a Constrained Robust Soft Actor Critic(C-RSAC) approach to solve it. In this problem set, the definition of MDP tuple is the same as in Chapter 5.4.1, but the agent index i needs to be specified for the robust agent and the noisy agent. Here $i = 0$ refers to the noisy agent, and $i = 1$ refers to the robust agent. We first train the two joint level controlled agents under C-MSAC framework to get two pre-trained policies π_0, π_1, and then train the robust agent under C-RSAC framework, that takes policy π_0 as the noisy agent's cooperative policy, and takes policy π_1 as the initial policy for the robust agent.

The noisy agent observes state $s_0^t \in \mathcal{S}_0^t$ from environment $\mathcal{O}$, and uses policy π_0 to get action $a_0^t \in \mathcal{A}_0^t$, receives a reward $r_0^t = \mathcal{R}(s_0^t,\, a_\mu^t, a_1^t)$. For the robust agent, it starts with policy π_1, and takes state $s_1^t \in \mathcal{S}_1^t$ to get action $a_1^t \in \mathcal{A}_1^t$, receives a reward $r_1^t = \mathcal{R}(s_1^t, a_\mu^t, a_1^t)$. We define the optimal action-value $Q_1^*(s_1, a_1)$ as the expected return for the robust agent, the goal is to find the optimal robust policy π_1^* that can maximize Q_1^*. At each time step t, it can be described according to the equation below:

$$Q_1^*(s_1, a_1) = \max_{\pi_1^*} \left(\mathbb{E}\left[|r_1^*| |\pi_0(s_0, a_\mu), \pi_1^*(s_1, a_1) \right] \right)$$

a_0^μ is the noisy action from the noisy agent, and it can be calculated as follows, ω is the noise ratio factor, $\mathcal{U}$ is the uniform distribution,

$$a_\mu = (1-\omega) \cdot a_0 + \omega \cdot \mu, \quad \mu \sim \mathcal{U}(0,1).$$

During training for the robust agent, we jointly conduct policy evaluation and policy iteration and concurrently update the stochastic policy π_1 and learn the two Q-functions $Q_{1,j}$ from both policies, j in the range of $[1, 2]$.

For policy iteration, we use Eq 6.1 to optimize the loss of policy function J_{π_1}, in the direction of maximizing the accumulated reward:

$$\nabla J_{\pi_1} = \nabla \frac{1}{|\mathcal{B}_1|} \sum_{\mathcal{B}_1} \{ \min_{j=1,2} Q_{1,j}(\mathbf{s}^t, a_\mu^t, a_1^t) - \alpha \log \pi_1(a_1^t | s_1^t) \}. \tag{6.1}$$

$\mathcal{B}_1$ denotes a batch of experiences sampled from the replay buffer of the robust agent, $\log \pi_1(a_1^t|s_1^t)$ is the entropy term to increase the variation of action space, and α is a learned variable indicating the contribution of an entropy regularization term. $Q_{1,j}(\mathbf{s}^t, a_\mu^t, a_1^t)$ is calculated from the joint states $\mathbf{s}$ and new joint actions (a_μ^t, a_1^t) of robust and noisy agent after adding noise disturbance.

For policy evaluation, we optimize the loss function $\mathcal{L}_{1,j}$ using Eq 6.2 to evaluate the learned robust policy by minimizing the difference of the value function $Q_{1,j}(\mathbf{s}^t, a_\mu^t, a_1^t)$ and its target value y_1 for robust agent:

$$\nabla \mathcal{L}_{1,j} = \nabla \frac{1}{|\mathcal{B}_1|} \sum_{\mathcal{B}_1} (Q_{1,j}(\mathbf{s}^t, a_\mu^t, a_1^t) - y_1)^2. \tag{6.2}$$

The target value y_i is calculated with a separate target value network $Q_{target}(\mathbf{s}^{t+1}, a_\mu^{t+1}, a_1^{t+1})$ in order to maintain stability when updating policy network.

6.4 Learning Agent under Physics

In this section, we describe the framework pipeline we use for training the agent under physics, details of task-related representation of states and actions, and the reward design for learning the tray balance task.

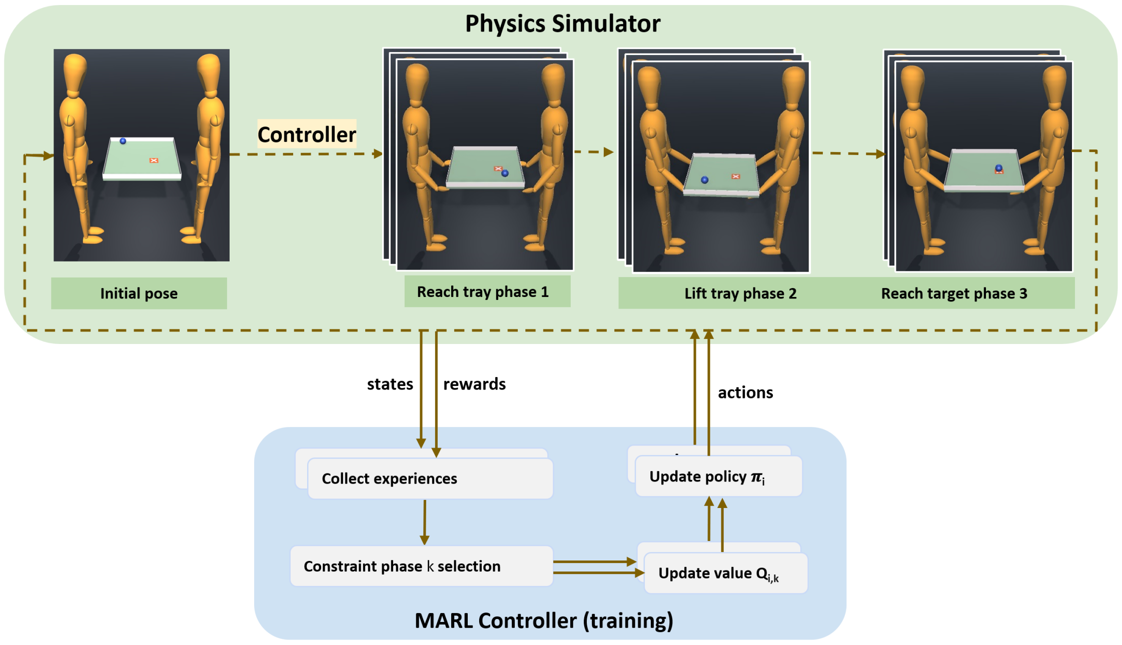


Figure 6.1: The framework overview for training collaborative agents under physics.

6.4.1 System Overview

When modeling the agent under physics, we adopt the previous C-MSAC framework where the two agents learn the control policy from the multi-agent controller (MARL Controller) during training. The physics engine in Unity provides the task environment and two joint-level controlled agents under physics to simulate the experiment. For completing the tray balancing task, three phases need to be finished. Both agents start from an initial standing pose. In the first phase, both agents need to stretch out their arms and try to reach the anchor points set on the tray anchor point. Once both hands grasp the tray, in phase 2 both agents start controlling the arm movement to lift up the tray to a target height level. After getting close enough to the target tray height, in phase 3 both agents try to balance the tray to allow the blue ball to follow the red target's moving trajectory. The arm movement is controlled by a PD controller from Unity. The overview is depicted in Figure 6.1.

6.4.2 State Representation

In our environment, we employ two 3D humanoid characters as agents, each featuring 20 joints encompassing the head, pelvis, arms, and legs. Specifically, our focus centers on the six arm joints, including the upper arms, lower arms, and hands, as illustrated in Figure 6.2. The states describe the configuration of the agent arms joint feature and the tray balance task properties. For each joint, the states include the distance between the joint and the tray, the rotation of the joint, and the angular velocity difference between the joint and the tray; for the hand joint, it also includes the distance and angle difference between the joint and the tray's anchor points. The tray balance task, as shown in the same figure, includes a green tray, a blue ball, and a red target on the surface of the tray. So for the task, the states include the distance between the ball and the target, ball and tray, tray and tray height target; the rotation of the ball and tray; the velocity of ball, and the velocity difference between the ball and tray. The rotations are expressed in quaternions.

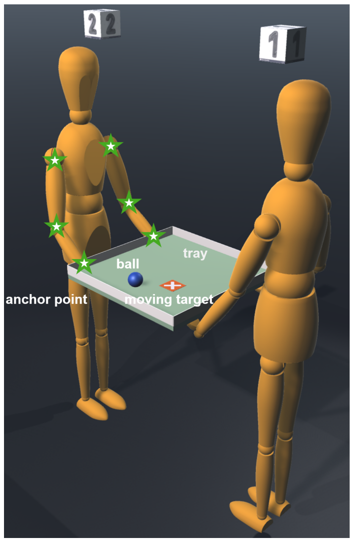


Figure 6.2: The task environment.

6.4.3 Action Representation

The physical movement of the agent is controlled by the PD controller at each joint. The actions provide the joint target orientation and the maximum joint driving force for the PD controller. In our environment, we use the Unity integrated PD controller to compute the torque at each joint. The target orientation is represented by a linear interpolation between the minimum and maximum axis angle for each joint, with the action signal as the interpolation factor. Thus, the torque applied at each joint is calculated by a quaternion spherical linear interpolation with the target orientation in the maximum drive force range. The action is queried at 10 Hz, and the simulator runs at 50 Hz, so at each query time t each action a_i^t is executed 5 times in the simulator. For each arm, there are three joints: upper arm, lower arm, and hand, and we use x, y, z orientation from the upper arm, y, z orientation from the lower arm, z orientation for hand to represent the arm movement, as shown in Figure 6.3.

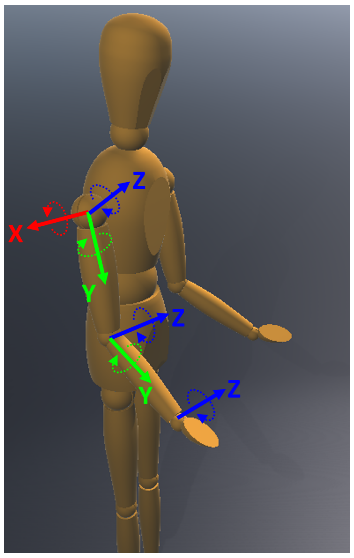


Figure 6.3: The agent arm joint-level orientations.

6.4.4 Reward Design

The tray balance task includes three phases. In phase 1, the hands need to reach the tray's anchor point. In phase 2, the arms need to lift up the tray. In phase 3, the arms need to move the tray to allow the ball to reach the target. Based on each phase's task objective, different reward functions have been designed. Besides the three phases, we also designed rewards to include some techniques for speeding up training, such as early termination and behavior regulation.

So for the tray balance task, we calculated the reward r_t at each time step t as follows:

$$r_t = select\{\ r_{hand},\quad r_{lift},\quad r_{target}\}\ + r_{behavior} + r_{terminate}.$$

r_{hand}, r_{lift}, r_{target} represent the reward for phases 1, 2, and 3.

The $select\{\ r_{hand},\quad r_{lift},\quad r_{target}\}$ term shows the phase selection process, based on the constraint approach proposed in (123), during each training update iteration, only one phase's reward will be selected for updating the agent's policy. $r_{behavior}$ is the behavior regulation reward, and $r_{terminate}$ is the early termination reward.

The details of the reward design are described below.

Phase 1: Hands Reach the Tray

In this phase, the agent's objective is to stretch both arms to allow each hand to reach the tray close to the anchor point, while maintaining the hand palm facing up to the tray bottom, to simulate human hand behavior while grasping the tray.

To satisfy this objective, the reward function is designed as:

$$r_{hand} = a * r_{dist} + b * r_{ang}.$$

Here, a, b are the reward ratio parameters, we used 0.5 for both in our experiments. r_{dist} is the Euclidean distance between each hand p_{lhand} and its corresponding anchor point $p_{lanchor}$, as the sketched yellow arrow in Figure 6.4, to encourage the hand to move closer to the anchor point, and it is calculated:

$$r_{dist} = \exp[-10 * (||p_{lhand} - p_{lanchor}||^2 + p_{rhand} - p_{ranchor}||^2)].$$

r_{ang} is the quaternion orientation difference between the hand palm $q_{lanchor}$ and its corresponding anchor point q_{rhand}, it is sketched as the red arrow in the figure, to encourage each to be able to grasp the tray properly, it is calculated:

$$r_{ang} = \exp[-1 * [(1 - ||q_{lhand} \ominus q_{lanchor}||^2) + (1 - ||q_{rhand} \ominus q_{ranchor}||^2)]].$$

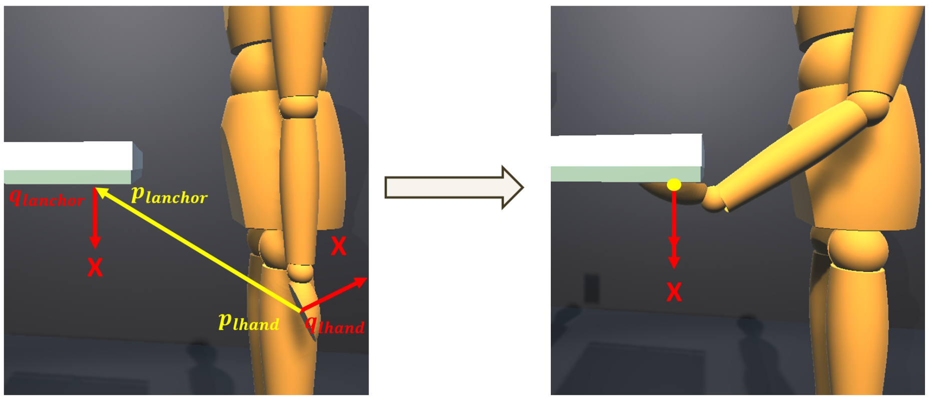


Figure 6.4: The representation for computing the hand reward. The red arrows indicate the orientation difference between the hand palm and tray anchor point, the yellow arrow indicates the hand distance to the anchor point. The left figure shows a random state of the agent. The right figure shows a target state for the agent's hand to achieve the maximum step reward of 1.0 for phase 1 (where the red arrows overlap and the yellow arrow reduces to a point).

Phase 2: Hands Lift the Tray

In this phase, the agent's objective is to lift up the tray to the fixed target position p_0 while maintaining its balance relative to the identity quaternion q_0. To satisfy this objective, the reward function is designed as:

$$r_{lift} = \exp[-5||p_T - p_0||^2].$$

It describes the Euclidean distance between the tray and the fixed target position, to encourage the tray to move closer to the fixed target position. In Figure 6.5 the sketched green arrow shows the moving direction.

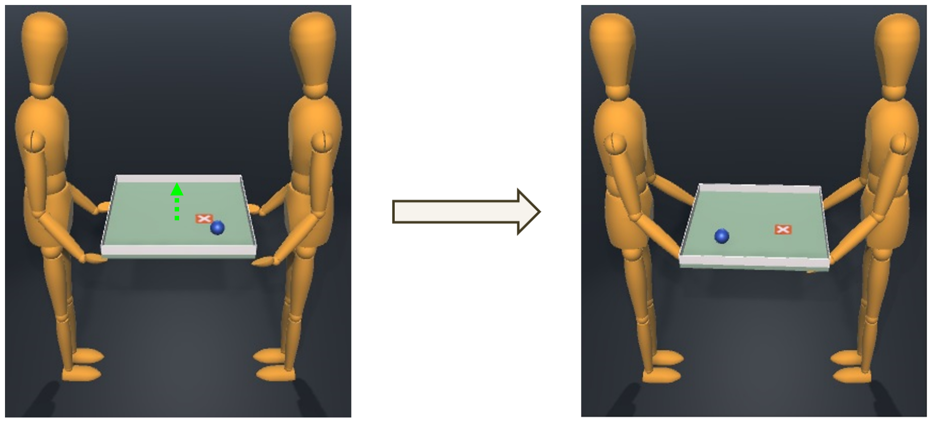


Figure 6.5: The representation for computing the tray lifting reward. The green arrow indicates the tray's moving direction. The left figure shows a random state of the agents. The right figure shows the target state to achieve the maximum step reward of 1.0 for phase 2.

Phase 3: Ball Reaches the Target

In this phase, the agent's objective is to adjust the tray position and orientation to control the ball to follow a moving target on the tray, as shown in Figure 6.6. To encourage the ball to stay on the moving target, we use the Euclidean distance between the ball and the moving target, as represented in the reward function:

$$r_{target} = \exp[-25||p_B - s^t||^2].$$

The target's moving path is controlled by an ellipse equation (where the major and minor axes are randomized at the beginning of each training episode).

Rewards for Speeding up Training

To speed up the training process, we use behavior regulation and early termination techniques.

For Behavior Regulation, we add a small penalty when the tray touches any lower arm of the agent, as shown in Figure 6.7, to avoid the tray being supported by lower arms and instead encourage the hands to be the only support for tray movement. The reward function $r_{behavior}$ is described as below:

$$r_{behavior} = \begin{cases} -0.1 & \text{if tray hits lower arm,} \\ 0 & \text{else.} \end{cases}$$

This relates to the training process of C-MSAC algorithm (123) because the training is focused on a single selected phase, and can advance to the objective phase when it satisfies a threshold from Monte Carlo

$$d = \sum_{t=0}^{T-1} \gamma^t r_0.$$

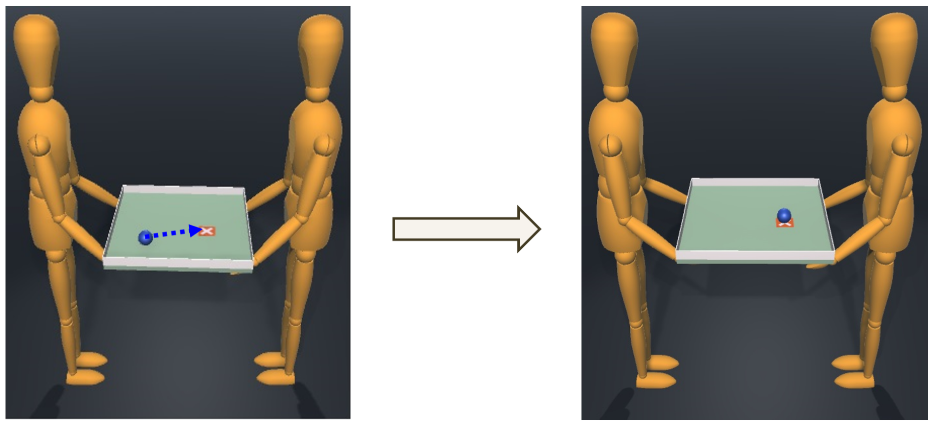


Figure 6.6: The representation for computing the ball reach the moving target reward. The blue arrow indicates the ball's moving direction. The left figure shows a random state of the task. The right figure shows the target state for the ball to reach a position that achieves the maximum step reward of 1.0 for phase 3.

where γ is the discount factor and r_0 is the step reward. We use $\gamma = 0.9$ in our experiment. For phase 1 (hand reaches tray), the step reward of r_{hand} can still be close to 0.9 when the arms support tray around this location as shown in Figure 6.7. This can cause the shown misbehavior from the agent for grasping the tray, thus leading the agent to update policy in the wrong direction and eventually making it hard to complete objectives for phases 2 and 3.

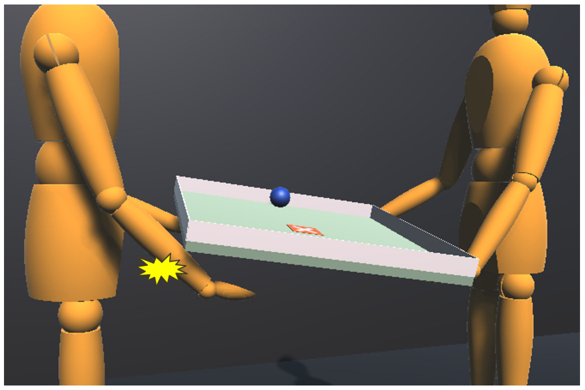


Figure 6.7: The tray hits the lower arms of the agent.

For Early Termination, reinforcement learning is an episodic training process, and terminating the training episode earlier is a common technique (104; 12) to help RL agents stop learning bad behaviors and favor the collected samples more efficiently, thus achieving significantly faster learning. In our task, we introduced two early termination conditions: first, when the ball touches the edge of the tray, because it can easily get stuck at the anchor point of the tray, as shown in Figure 6.8; second, when the ball or tray hits the ground, as the agents can not recover from this state, as shown in Figure 6.9. The reward $r_{terminate}$ can be described as follow:

$$r_{terminate} = \begin{cases} -1 & \text{if meet early termination conditions,} \\ 0 & \text{else.} \end{cases} \tag{6.3}$$

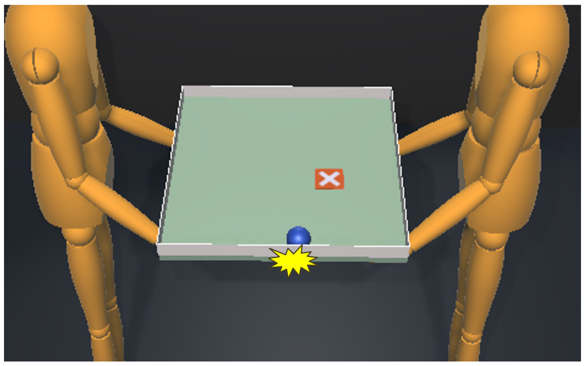


Figure 6.8: The blue ball hits the edge of the tray.

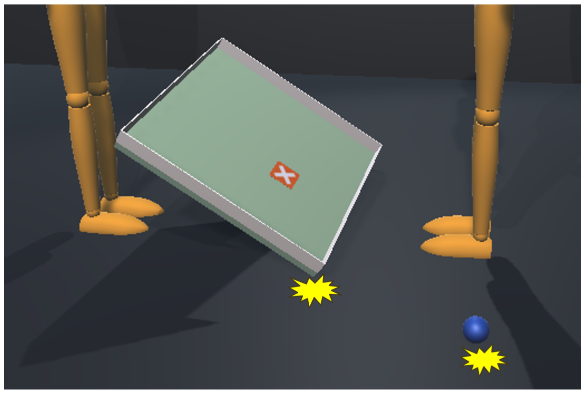


Figure 6.9: The tray hits the ground or the blue ball hits the ground.

6.5 Learning Robust Agent

6.5.1 System Overview

For learning the robust agent, we refer to agent 0 as the noisy agent and agent 1 as the robust agent as described in 6.1.2. We describe our framework in three parts: the physics simulator, the MARL controller, and the robust RL controller. Figure 6.10 shows an overview of the framework. The physics simulator is still used to simulate the two joint-level controlled agents finishing the tray balance task in the Unity game engine. The MARL controller is used to generate the pre-trained policies π_0 and π_1.

The robust RL controller includes two agents: the robust agent and the noisy agent. The noisy agent acquires π_0 as its control policy, and produces noisy action a_μ, while the robust agent acquires π_1 as its initial policy to start the learning. The learning process is described in the C-RSAC algorithm.

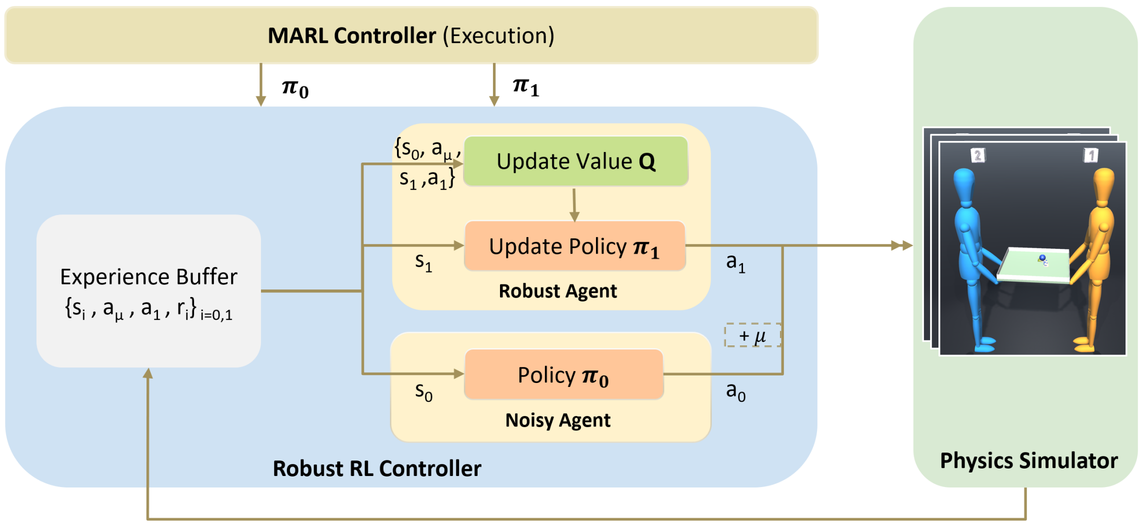


Figure 6.10: The framework overview for learning a robust agent.

6.5.2 Algorithm

Algorithm 2 summarizes the procedure used in our proposed approach C-RSAC during the training process. On lines 1-3 is the policy initialization process for both robust agent and noisy agent, the policy network θ_0 and θ_1 are the pre-trained policies from C-MSAC framework. Lines 5-11 show the process of collecting samples of the noisy agent and robust agent in different ways, and lines 7-8 add a randomized noise to the noisy agent. Lines 12-14 are the same policy updating procedure as in the C-MSAC algorithm (123).

6.6 Results

In this section, we show the results of all the experiments. We refer to the learned multi-agent policies with joint-level controlled agents as agents under physics, and the learned multi-agent policies with IK controlled agents as agents under inverse kinematics (IK).

6.6.1 Compute Platform

We train all our experiments using the Google Cloud Platform (GCP). Overall 8 machines with Ubuntu 20 were used for training. Each machine is a c2-standard-4

Algorithm 2 Constrained Robust Soft Actor Critic Algorithm (C-RSAC)

```
1: Import policy network $\theta_0$ for noisy agent;
2: Initialize policy network $\theta_1$ for robust agent, and value estimation networks $\phi_1$,
   $\phi_2$ for each phase;
3: Initialize replay buffer $\mathcal{D}$ as empty dictionary ;
4: for each step do
5:     Sample action $a_1^t$ from policy $\pi_{\theta_1}(a_1^t \big| s_1^t)$ for robust agent;
6:     Output mean action $a_0^t$ from policy $\pi_{\theta_0}(a_0^t \big| s_0^t)$ for noisy agent;
7:     Generate noise $\mu$ from a uniform distribution $\sim U[0, \omega]$;
8:     Compute noisy action $a_\mu^t$ for the noisy agent;
9:     Execute the joint actions($a_1^t$, $a_\mu^t$) in task environment;
10:    Collect reward $r^t$ and next state $s^{t+1}$ from environment;
11:    Send tuple ( $s_i^t$, $a_\mu^t$, $a_1^t$, $r_i^t$, $s_i^{t+1}$ )$_{i=0,1}$ to replay buffer $\mathcal{D}$;
12:    if step > batch_size then
13:        Select phase $k$ as constraint phase;
14:        Update policy $\theta_1$;
15:    end if
16: end for
17: Output optimal policy $\theta_1$ for the robust agent.
```

instance, with 4 CPU cores and 16 GB memory. The episodic length for the tray balance task is 260 control steps, and 1300 simulation steps, to allow the moving target to form a complete ellipse shape trajectory. For each experiment, we train 500,0000 episodes, that is 130 million control steps. For training multi-agent models under physics, each experiment takes approximately 10 days. For training the robust agent, it takes about 3 days since a pre-trained initial policy is employed for both the robust agent.

6.6.2 Parameters

Table 6.1 shows all the parameters used for the robust learning and multiagent multiphase learning for the tray balance task. The network we use is a three-layer fully connected network, and the activate function is ReLU.

6.6.3 Learning Agent under Physics

In this section, we present the performance and learning outcomes of agents under physics and compare those with the corresponding outcomes for the agents under IK.

Comparing Agent under IK and Agent under Physics

Comparing the arm movements of the agent trained under IK with the agent trained under physics, we observe that the arm movements of the agent under physics are a

Physics engine simulation rate(Hz)	50
Learning policy control rate(Hz)	10
Policy learning rate(η)	0.0009
Discount factor(γ)	0.99
Soft update parameter τ	0.005
Explore steps	1000
Batch size	256
Neural network hidden state dim	256
Iterations for policy update	100
Number to update	4
Reward scale	1.0
Threshold step reward	0.9

Table 6.1: Training parameters.

lot more natural, with reduced oscillations and glitches. As shown in Figure 6.11, we also observe that the ball trajectory for the agent under physics is smoother compared to that for the agent under IK.

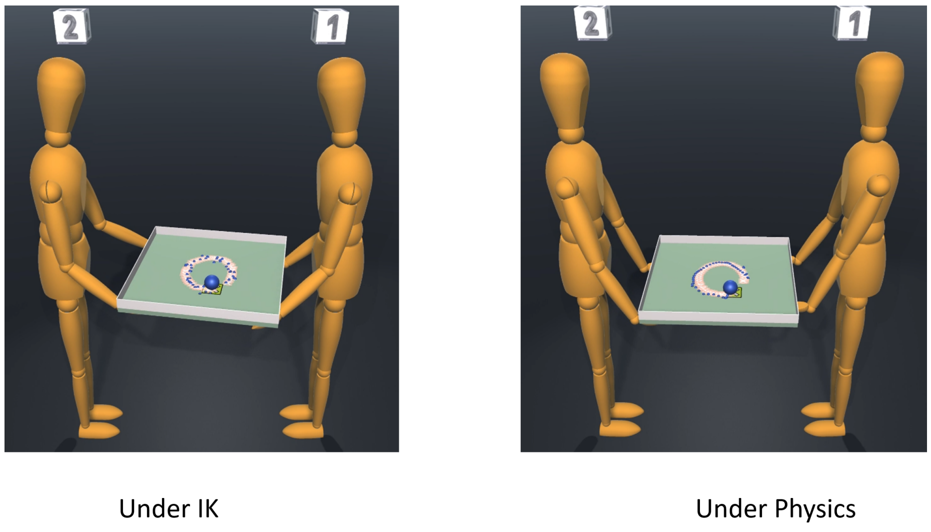


Figure 6.11: The comparison of two agents collaboratively completing the trajectory following tray balance task under IK(left figure) and physics(right figure).

Grasping Behavior

When designing the reward function for phase 1 (hand reaches the tray), we introduce two additional terms to encourage the hand to reach the tray anchor point. One of the terms focuses on grasping position and the other term focuses on the grasping angle. We also introduce two parameters a, b that correspond to the weight of the grasping position term and the grasping angle term respectively. a and b are ratios with values within $[0, 1]$ such that $a + b = 1$. The reward function is expressed as $r_{hand} = a * r_{dist} + b * r_{ang}$. If a is greater than b, the reward term encourages hands to reach the exact grasping position on the tray. Otherwise, the palms are more encouraged to face up to the tray to have a more realistic grasping behavior. Figure 6.12 shows the grasping behavior we observed from the learned agent while adjusting these reward ratio parameters. The first figure shows the result from IK-based agents, and the remaining three figures show the result from physics-based agents. For the IK-based setup, the anchor point is used as the IK target and so the hands of both agents are incapable of moving away from the anchor point. As a result oscillation of the arms and penetration of arms to the tray can occur when the tray moves away from the reach range of the agent. For the physics-based agent, the arm is controlled by joint-level movement. Although the reward terms incentivize grasping the tray at the anchor point, the agents have more flexibility to grasp around the anchor point. The three figures under physics show examples of this flexibility with agents grasping far away from the anchor point. The third figure ($a = 0.5, b = 0.5$) is actually a snapshot of an interesting instance where the agent exhibits a sliding grasping behavior on the tray (which can be seen in the simulation video).

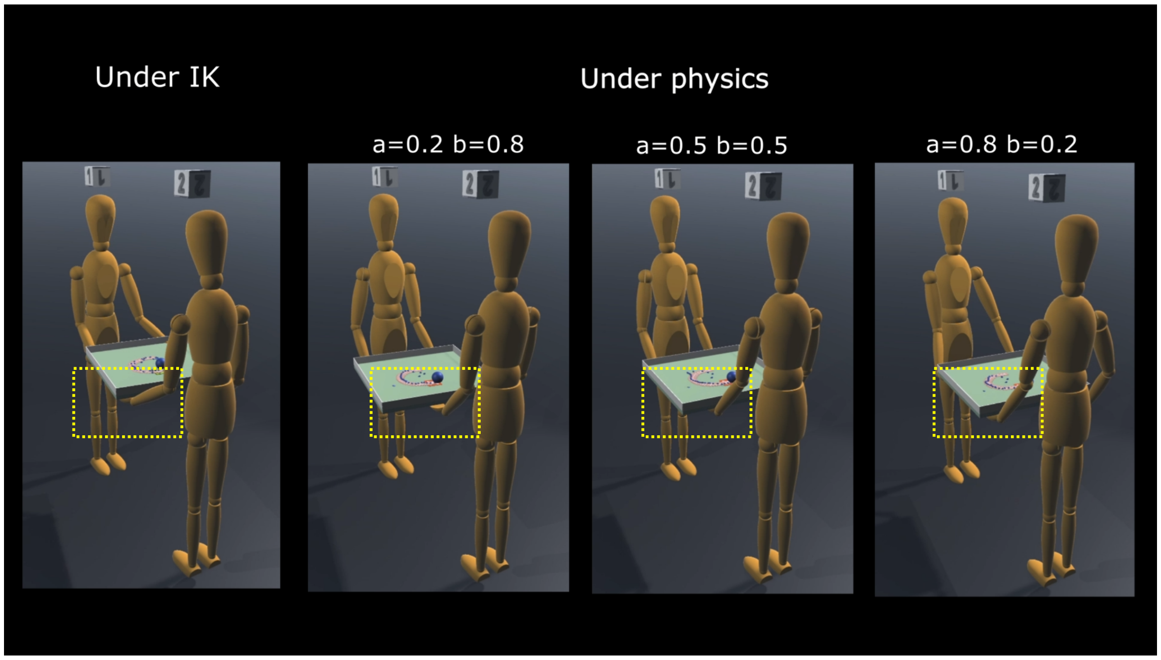


Figure 6.12: The different grasping behaviors of agents under IK and physics.

Robustness Evaluation

We analyzed the robustness of the learned policies to changes in the physical properties of the environment by varying the ball mass. We run evaluations where the mass of the ball is multiplied by a fixed ratio k. $k = 1$ corresponds to the case where the mass of the ball during evaluation matches the mass of the ball used during the training process. $k > 1$ corresponds to cases where the ball mass used in the evaluation is higher than the mass used during the training process. We vary the ball mass ratio k from 1 to 25 and measure the mean episode reward as well as the corresponding standard deviation for the agent under IK and the agent under physics.

Figure 6.13 shows the results of this experiment. We can see that the agent under IK achieves a higher reward in the matched condition for $k = 1$. However, the agent under IK shows unnatural arm movements as we have noted earlier, and this effect is not captured in the reward statistics. As we increase the ball mass ratio k beyond 1, the agent under physics outperforms the agent under IK even in terms of the mean episode reward. As the mass ratio approaches 25, both agent under IK and agent under physics are unable to cope with the increased ball mass and the mean reward approaches zero.

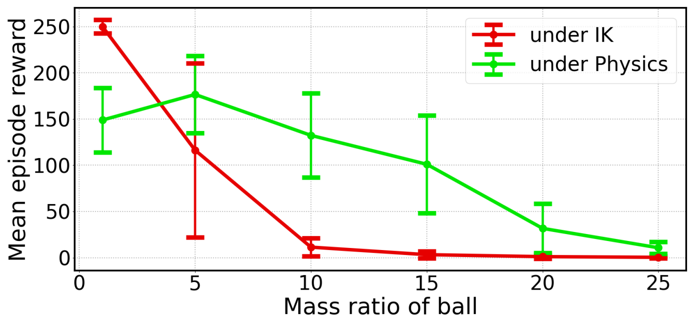


Figure 6.13: The mean episode reward performance against the ball mass ratio for agents under IK (blue) and under Physics (orange).

Parameter Selection and Reward Averaging

Parameter selection plays a critical role in reinforcement learning as different types of tasks can have different optimum values of learning parameters such as learning rate. Besides that, additional parameters can be introduced used when designing the task specific reward functions and those need to be adjusted to optimize the task performance. Improving task performance is even more critical in our case since the results of C-MSAC training are later used as initial policies for C-RSAC. We studied three parameters that are important for our tray balance task: the learning rate and the ratio parameters a, b described in 6.6.3 (Grasping behavior). In addition, we also studied optimizing the individual agent rewards by optimizing a single averaged reward during the training process. We periodically run evaluations during the training process and compare the results in terms of the mean episode reward on testing episodes for the three task phases. For all the figures, we use Gaussian smoothing to average out the noisy fluctuations while highlighting the overall trend. We also show the standard deviation for the episode rewards by shading the region around the mean, where a scaling factor of 0.5 is applied to the standard deviation in order to avoid overcrowding the figures.

- **Learning Rate**, Figure 6.14 shows the results from varying the learning rate used during our training. We chose the value of 0.0009 to train our final policy as it has the best mean episode reward performance on all three phases.
- **Hand Reach Reward Ratio**, Figure 6.16 shows the result of varying the reward ratios a, b related to the grasping behavior in our tray balance task. Both the hand and the tray are separately covered with a rectangular collision bounding box. So the more the palm touches the tray, the better support and more realistic grasp the hand can give to the tray, which relates to the reward ratio b. The result shows that $(a, b) = (0.2, 0.8)$ as the pair of reward ratios achieves the best task performance. However, $(a, b) = (0.5, 0.5)$ also achieves comparable performance, and videos of the simulations show that it results in a better grasping behavior as there is an equal encouragement from the reward function to grasp both at the proper position and the proper angle.
- **Reward Averaging**, Figure 6.15 compares the task performance between two cases: one where each agent's reward is calculated separately for optimization and one where we average their rewards. In our collaborative task environment, when training the agents under IK we choose to average the sum reward of both agents and assign it back to each agent during the training process to achieve a Nash equilibrium (100). However, while training the agents under physics this can potentially cause issues due to the flexibility in the way agents choose to grab the tray. In particular, agents are penalized when their lower arm touches the tray. If we average the reward, even if just one agent's lower arm touches the tray both agents are penalized. To avoid this scenario from happening, we chose not to average the rewards while training in physics. The comparison in Figure 6.15 confirms our intuition showing that not averaging the rewards leads to a better performance, especially in the final phase of reaching the target.

6.6.4 Learning Robust Agent

Performance Based on Mean Episode Reward

While training the robust agent with C-RSAC we use the pre-trained policies from C-MSAC under physics as the execution policy for the noisy agent, and the initial policy for the robust agent. During training, we periodically run evaluations and calculate the mean episode reward for the robust agent while collaborating with the noisy agent. Figure 6.17 shows the mean episode reward on the evaluation episodes for

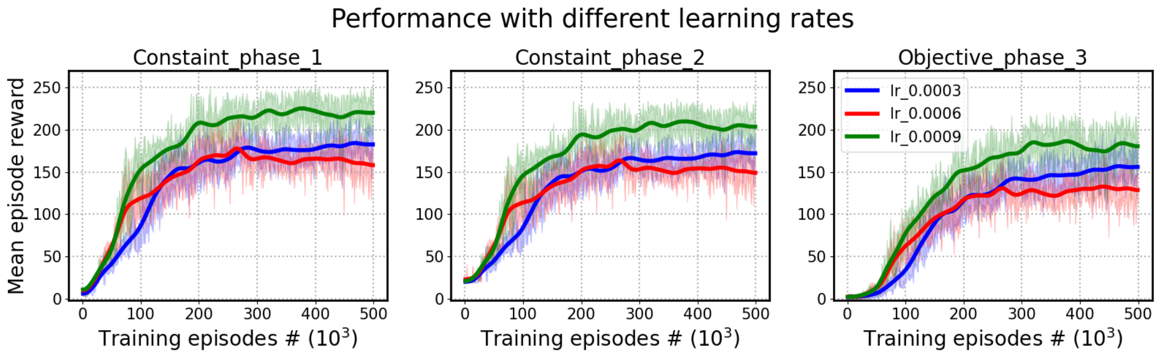


Figure 6.14: The mean episode reward performance of different learning rates.

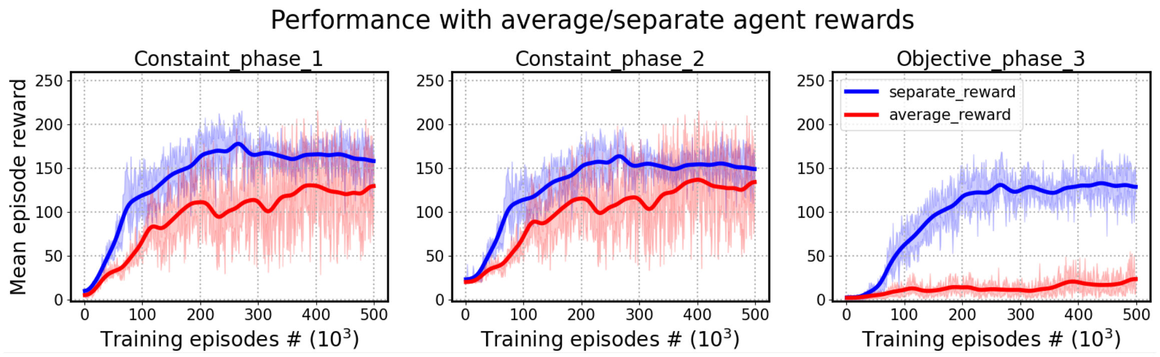


Figure 6.15: The mean episode reward performance of averaging and separating agents rewards.

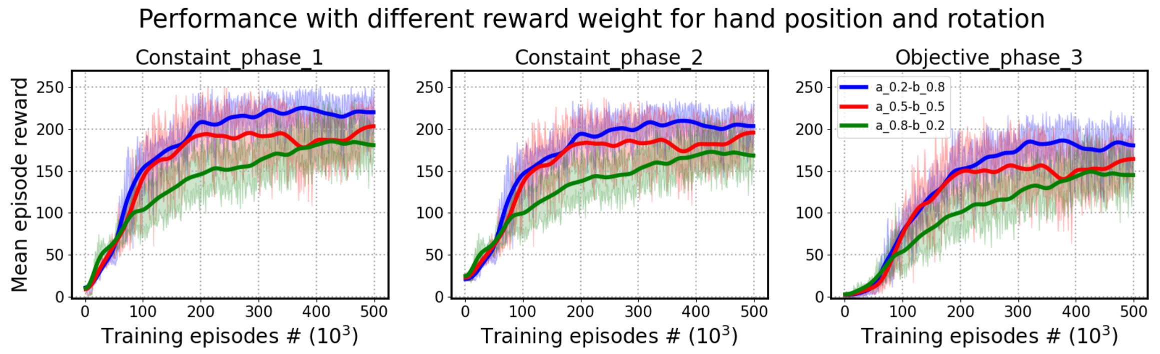


Figure 6.16: The mean episode reward performance of designing hand reward with different reward ratio parameters.

all three phases as training progresses. We run evaluation on 100 test episodes after every 2000 steps of training. The leftmost value for 0 steps of training corresponds to the initial policy obtained from C-MSAC. The graph shows that the mean episode reward increases for all three phases as the training progresses, indicating that the C-RSAC leads to improved robustness compared to C-MSAC.

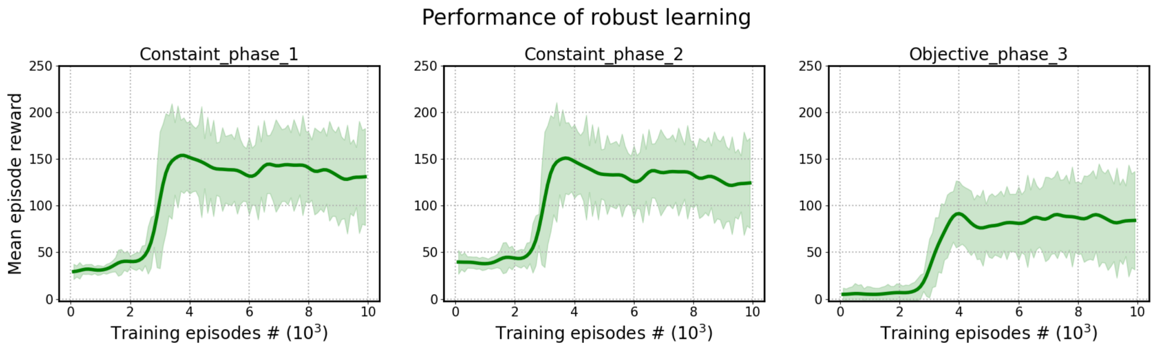


Figure 6.17: The robust agent performance.

Comparing Mean Episode Reward Across Different Noise Scales

We measure the mean episode reward by varying the noise scale ω which determines the intensity of the noise applied to the noisy agent's actions. $\omega = 0$ corresponds to the case where there is no noise applied to the actions, and as ω increases more and more noise is added to the actions. The results are shown in Figure 6.18.

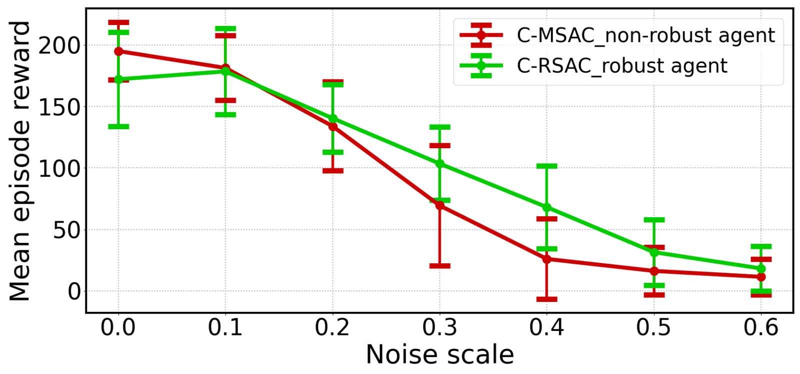


Figure 6.18: Comparison of mean episode reward for C-MSAC agent and C-RSAC agent for different values of noise scale ω

We can see that in the absence of noise ($\omega = 0$), C-MSAC performs slightly better than C-RSAC but the difference of mean episode reward is smaller than the corresponding standard deviations. For $\omega = 0.1$ and $\omega = 0.2$, the mean reward is roughly equal. However, as ω increases above 0.2, we see that C-RSAC achieves a significantly better mean episode reward compared to C-MSAC. This shows that the agent trained with C-RSAC exhibits a higher robustness to noise.

Comparing On-target Performance Across Different Noise Scales

Similar to Section 5.6.3, we obtain histograms for the number of on-target steps to evaluate the on-target performance for C-MSAC and C-RSAC. We run evaluation for 2000 episodes, and define the ball to be on-target at a given step if the target reward at any step is above 0.7. The histogram for $\omega = 0.3$ is shown in Figure 6.19 and for $\omega = 0.4$ is shown in Figure 6.20.

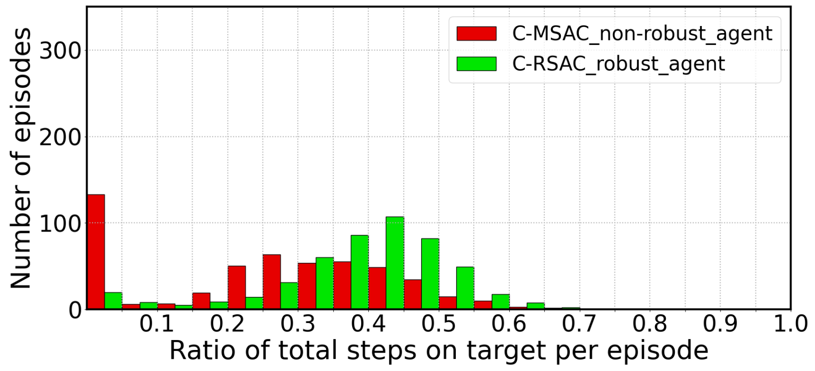


Figure 6.19: Histogram of on-target steps for noise scale $\omega = 0.3$.

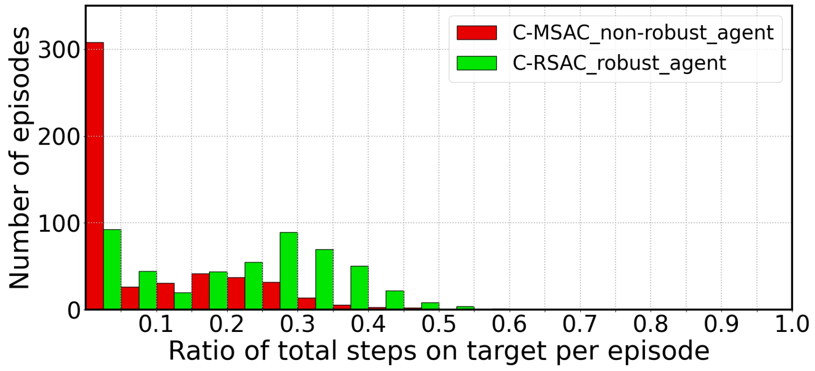


Figure 6.20: Histogram of on-target steps for noise scale $\omega = 0.4$.

The figures show that the histogram for C-RSAC is shifted more towards the right compared to C-MSAC. This demonstrates that under noisy conditions the agent trained with C-RSAC is able to maintain the ball on target more frequently compared to the agent trained with C-MSAC.

User Control

For interactive visualization, we also add a user interface for users to interact with the robust agent by adding a user-controlled noise signal to the noisy agent, as shown in Figure 6.21. The blue agent is the robust agent, and the yellow agent is the noisy agent. The agent action space includes orientation in x, y, z direction for upper arms, y, z direction for lower arms, and z direction for hands. We chose to add noise in the z direction for upper arms, the y and z direction for lower arms, and the z direction for hands for ease of user control and also based on our observation of the arm movement while performing the task. For each arm, the users can adjust the scroll bar to control the noise signal that is added to the yellow noisy agent.

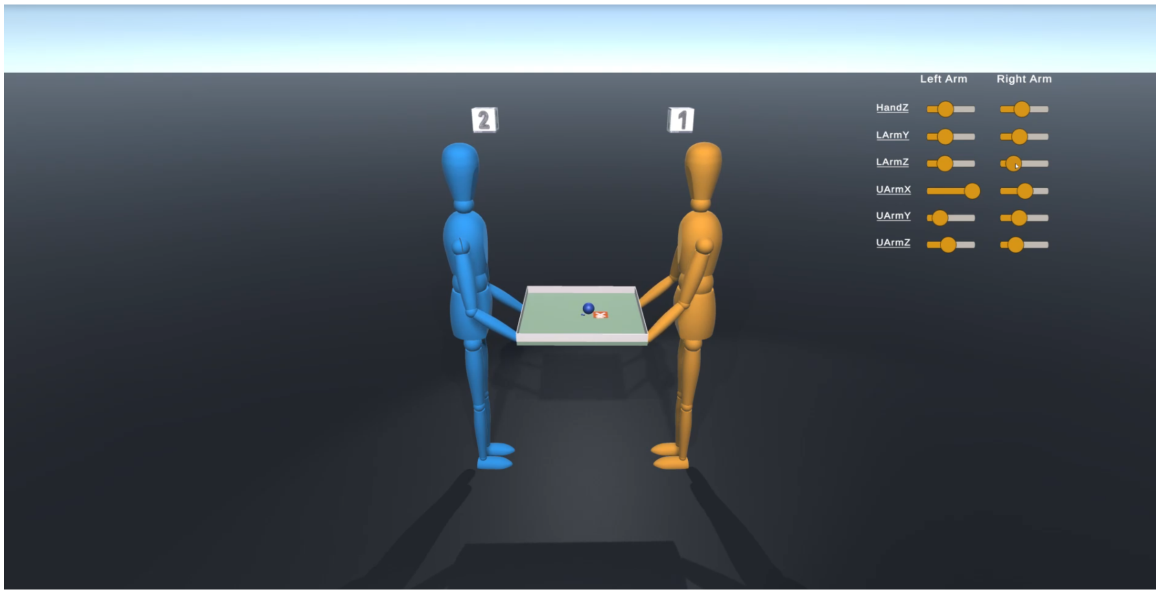


Figure 6.21: The environment for robust agent reacts to user-controlled noisy agent.

6.7 Conclusions

In this chapter, we address the challenges of mitigating unnatural oscillatory movements arising from IK-based control and the robustness of the agents against noise. We propose physics-based joint-level full-arm control for C-MSAC which leads to more natural movements and also improves robustness to changes in the physical properties of the environment such as the mass of the ball. We propose the C-RSAC approach which uses the pre-trained policy from C-MSAC for the noisy agent and add noise in the action space in order to simulate the effect of errors from another agent or human. We show that training the robust agent to collaborate with such a noisy agent leads to an improved robustness to noise, as demonstrated by the superior performance of C-RSAC resulting in higher mean test episode reward for all three phases, improved performance for higher noise scales and better on-target performance under noise.

Chapter 7

Conclusion

This thesis addressed challenges related to the design and training of virtual human agents on complex collaborative tasks. Numerous aspects were addressed such as the impact of agent gender and feedback strategies on user performance and preference. Various approaches for training agents to collaborate on multi-phase tasks were also proposed, achieving natural movements and robustness to noise while avoiding the need for relying on expensive human interaction data.

In chapters 3 and 4 user studies were performed to study the effect of design variables like agent gender and feedback strategies on user performance and preference. It was presented that while agent gender had no significant impact on user performance or preference, female agents were considered more attractive by the users. A comparison between correctness and suggestive feedback strategies was also presented and the results were that users preferred suggestive feedback which also led to a 65% reduction in task completion time.

In chapter 5 the C-MSAC approach was proposed for training agents on multi-phase tasks using constraints. The proposed approach was evaluated on a tray balancing task and showed that the proposed approach leads to a better performance in terms of mean episode reward compared to an unconstrained baseline. C-MSAC also lead to a better generalization performance on unseen trajectories and exhibited better robustness to environmental disturbances.

In chapter 6 a joint-based control method was proposed for training the agents with C-MSAC under physics which led to more natural movements and better robustness against changes to physical parameters. The C-RSAC approach was also introduced for improving the robustness of an agent by training it to collaborate with a noisy agent.

Overall, the findings from this thesis provide valuable insights into the design and modeling choices required for getting closer to the end goal of developing virtual human agents that can collaborate effectively with humans in challenging collaborative tasks in immersive virtual reality settings.

7.1 Future Directions

One interesting future direction for this work could be to use human interaction data to fine-tune the policies learned using C-RSAC. The obtained findings could also be validated on more complex tasks involving more phases. Recent advances in Machine Learning (ML) literature could be incorporated to improve the neural network architectures used for controlling the agents.

Another promising direction is to evaluate the proposed learning methods in an application interacting directly with real users in an immersive virtual reality environment.

Bibliography

[1] Abdul-Kader, S. A. and Woods, J. C. (2015). Survey on chatbot design techniques in speech conversation systems. *International Journal of Advanced Computer Science and Applications*, 6(7).

[2] Abulrub, A.-H. G., Attridge, A. N., and Williams, M. A. (2011). Virtual reality in engineering education: The future of creative learning. In *2011 IEEE Global Engineering Education Conference (EDUCON)*, pages 751–757. IEEE.

[3] Accessories, O. R. (2018). *https://www.oculus.com/rift/accessories.*

[4] Achiam, J., Held, D., Tamar, A., and Abbeel, P. (2017). Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR.

[5] Agrawal, S. and van de Panne, M. (2016). Task-based locomotion. *ACM Transactions on Graphics (TOG)*, 35(4):1–11.

[6] Allen, J. and Ferguson, G. (2002). Human-machine collaborative planning. In *Proceedings of the Third International NASA Workshop on Planning and Scheduling for Space*, pages 27–29.

[7] Almutairi, B. and Rigas, D. (2014). The role of avatars in e-government interfaces. In *International Conference of Design, User Experience, and Usability*, pages 28–37.

[8] Andrist, S., Pejsa, T., Mutlu, B., and Gleicher, M. (2012). *A head-eye coordination model for animating gaze shifts of virtual characters.* In: 4th Workshop on Eye Gaze in Intelligent Human Machine Interaction.

[9] Andrychowicz, O. M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al. (2020). Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20.

[10] Aristidou, A., Lasenby, J., Chrysanthou, Y., and Shamir, A. (2018). Inverse kinematics techniques in computer graphics: A survey. In *Computer graphics forum*, volume 37, pages 35–58. Wiley Online Library.

[11] Attali, Y. (2015). Effects of multiple-try feedback and question type during mathematics problem solving on performance in similar problems. *Computers & Education*, 86:260–267.

[12] Babadi, A., Naderi, K., and Hämäläinen, P. (2019). Self-imitation learning of locomotion movements through termination curriculum. In *Motion, Interaction and Games*, pages 1–7.

[13] Bagnell, J. A., Ng, A. Y., and Schneider, J. G. (2001). Solving uncertain markov decision processes.

[14] Bailenson, J. N., Blascovich, J., Beall, A. C., and Loomis, J. M. (2003). Interpersonal distance in immersive virtual environments. *Personality and Social Psychology Bulletin*, 29(7):819–833.

[15] Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., and Mordatch, I. (2019). Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*.

[16] Banakou, D. and Chorianopoulos, K. (2010). The effects of avatars' gender and appearance on social behavior in online 3D virtual worlds. *Journal for Virtual Worlds Research*, 2(5).

[17] Bangert-Drowns, R. L., Kulik, J. A., and Kulik, C.-L. C. (1991). Effects of frequent classroom testing. *The Journal of Educational Research*, 85(2):89–99.

[18] Barto, A. G. and Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1):41–77.

[19] Baylor, A. L. (2009). Promoting motivation with virtual agents and avatars: role of visual presence and appearance. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1535):3559–3565.

[20] Baylor, A. L. and Kim, Y. (2004). Pedagogical Agent Design: The Impact of Agent Realism, Gender, Ethnicity, and Instructional Role. In *International Conference on Intelligent Tutoring Systems*, pages 592–603. Springer, Berlin, Heidelberg.

[21] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279.

[22] Ben-Tal, A., El Ghaoui, L., and Nemirovski, A. (2009). *Robust optimization*, volume 28. Princeton university press.

[23] Bente, G., Dratsch, T., Rehbach, S., Reyl, M., and Lushaj, B. (2014). *Do you trust my avatar? Effects of photo-realistic seller avatars and reputation scores on trust in online transactions.* In: International Conference on HCI in Business.

[24] Biljanovic, P., for Information, C. S., Communication Technology, E., and Microelectronics-MIPRO. (2010). Intelligent Pedagogical Agents in immersive virtual learning environments: A review. In *MIPRO 2010 : 33rd International Convention on Information and Communication Technology, Electronics and Microelectronics*, Opatija, Croatia. Croatian Society for Information and Communication Technology, Electronics and Microelectronics.

[25] Brady, A. T. and Walker, M. B. (1978). Interpersonal distance as a function of situationally induced anxiety. *British Journal of Social and Clinical Psychology*, 17(2):127–133.

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

[27] Chaminade, T., Hodgins, J., and Kawato, M. (2007). Anthropomorphism influences perception of computer-animated characters's actions. In *Social Cognitive and Affective Neuroscience*. Books (MIT Press.

[28] Chen, J. Y. C. and Barnes, M. J. (2014). Human–agent teaming for multirobot control: A review of human factors issues. *IEEE Transactions on Human-Machine Systems*, 44(1):13–29.

[29] Chen, Z. and Liu, B. (2018). Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207.

[30] Cheng, Z., Liwang, M., Chen, N., Huang, L., Du, X., and Guizani, M. (2022). Deep reinforcement learning-based joint task and energy offloading in uav-aided 6g intelligent edge networks. *Computer Communications*, 192:234–244.

[31] Cherubini, A., Passama, R., Crosnier, A., Lasnier, A., and Fraisse, P. (2016). Collaborative manufacturing with physical human–robot interaction. *Robotics and Computer-Integrated Manufacturing*, 40:1–13.

[32] Clegg, A., Yu, W., Tan, J., Liu, C. K., and Turk, G. (2018). Learning to dress: Synthesizing human dressing motion via deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 37(6):1–10.

[33] Dede, C. (2009). Immersive interfaces for engagement and learning. *Science*, 323(5910):66–69.

[34] Doerner, R., Tesch, A., Hildebrand, A., Leenders, S., Tropper, T., Wilke, W., Winkler, C., Hillig, J., Pestov, A., Walsh, J. A., et al. (2022). Vr/ar case studies. In *Virtual and Augmented Reality (VR/AR) Foundations and Methods of Extended Realities (XR)*, pages 331–369. Springer.

[35] Dörner, R., Kallmann, M., and Huang, Y. (2015). *Content Creation and Authoring Challenges for Virtual Environments: From User Interfaces to Autonomous Virtual Characters*, pages 187–212. Springer International Publishing, Cham.

[36] falcão, T. P. and Pontual, T. (2018). Feedback and Guidance to Support Children with Intellectual Disabilities in Discovery Learning with a Tangible Interactive Tabletop. *ACM Transactions on Accessible Computing*, 11(3):1–28.

[Fidelity] Fidelity, H. *https://highfidelity.com*. last accessed 2018/08/03.

[38] Florensa, C., Duan, Y., and Abbeel, P. (2017). Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*.

[39] Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. (2018). Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

[Framework] Framework, O. S. *https://www.oculus.com/experiences/rift/1776111379163747*. last accessed 2018/08/04.

[41] Freina, L. and Ott, M. (2015). A Literature Review on Immersive Virtual Reality in Education: State Of The Art and Perspectives. Technical report.

[42] Garcıa, J. and Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480.

[Generator] Generator, A. C. *https://charactergenerator.autodesk.com*. last accessed 2018/08/03.

[44] Gleave, A., Dennis, M., Wild, C., Kant, N., Levine, S., and Russell, S. (2019). Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*.

[45] Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

[46] Graesser, A., Lu, S., Jackson, G., Mitchell, H., Ventura, M., Olney, A., and Louwerse, M. (2004). Autotutor: a tutor with dialogue in natural language. 36:180–192.

[47] Granitz, N. A., Koernig, S. K., and Harich, K. R. (2009). Now It's Personal: Antecedents and Outcomes of Rapport Between Business Faculty and Their Students. *Journal of Marketing Education*, 31(1):52–65.

[48] Gris, I. and Novick, D. (2018). Virtual Agent Interaction Framework (VAIF): A Tool for Rapid Development of Social Agents. In *AAMAS '18 Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2230–2232, Stockholm, Sweden.

[49] Grochow, K., Martin, S. L., Hertzmann, A., and Popović, Z. (2004). Style-based inverse kinematics. In *ACM SIGGRAPH 2004 Papers*, pages 522–531.

[50] Guadagno, R. E., Blascovich, J., Bailenson, J. N., and Mccall, C. (2007). Virtual humans and persuasion: The effects of agency and behavioral realism. *Media Psychology*, 10(1):1–22.

[51] Guénette, D. (2007). Is feedback pedagogically correct?: Research design issues in studies of feedback on writing. *Journal of Second Language Writing*, 16(1):40–53.

[52] Ha, S. and Liu, C. K. (2014). Iterative training of dynamic skills inspired by human coaching techniques. *ACM Transactions on Graphics (TOG)*, 34(1):1–11.

[53] Haake, M. and Gulz, A. (2008). Visual stereotypes and virtual pedagogical agents. *Journal of Educational Technology Society*, 11(4).

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

[55] Hanus, M. D. and Fox, J. (2015). Assessing the effects of gamification in the classroom: A longitudinal study on intrinsic motivation, social comparison, satisfaction, effort, and academic performance. *Computers & education*, 80:152–161.

[56] Hartholt, A., Traum, D., Marsella, S., Shapiro, A., Stratou, G., Leuski, A., Morency, L.-P., and Gratch, J. (2013). All Together Now Introducing the Virtual Human Toolkit. In *International Workshop on Intelligent Virtual Agents*, pages 368–381. Springer, Berlin, Heidelberg.

[57] Haworth, B., Berseth, G., Moon, S., Faloutsos, P., and Kapadia, M. (2020). Deep integration of physical humanoid control and crowd navigation. In *Motion, Interaction and Games*, pages 1–10.

[58] Hebert, J. S. (2014). : Normative data for modified Box and Blocks test measuring upper-limb function via motion capture. *Journal of rehabilitation research and development*, 51(6).

[59] Hernandez-Leal, P., Kartal, B., and Taylor, M. E. (2019). A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797.

[60] Holden, D., Komura, T., and Saito, J. (2017). Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)*, 36(4):1–13.

[61] Huang, Y. and Kallmann, M. (2016). Planning motions and placements for virtual demonstrators. *IEEE Transactions on Visualization & Computer Graphics*, 22(5):1568–1579.

[62] Hundt, A., Killeen, B., Greene, N., Wu, H., Kwon, H., Paxton, C., and Hager, G. D. (2020). "good robot!": Efficient reinforcement learning for multi-step visual

tasks with sim to real transfer. *IEEE Robotics and Automation Letters*, 5(4):6724–6731.

[63] Inc, A. (2018). *https://altvr.com.* last accessed 2018/08/03.

[64] Iqbal, S. and Sha, F. (2019). Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 2961–2970. PMLR.

[65] Jo, D., Kim, K., Welch, G. F., Jeon, W., Kim, Y., Kim, K. H., and Kim, G. J. (2017). *The impact of avatar-owner visual similarity on body ownership in immersive virtual reality.* In: 23rd ACM Symposium on Virtual Reality Software and Technology.

[66] Johannsmeier, L. and Haddadin, S. (2016). A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes. *IEEE Robotics and Automation Letters*, 2(1):41–48.

[67] Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., et al. (2018). Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*.

[68] Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., and Lange, D. (2020). Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*.

[69] Kartoun, U., Stern, H., and Edan, Y. (2010). A human-robot collaborative reinforcement learning algorithm. *Journal of Intelligent & Robotic Systems*, 60(2):217–239.

[70] Kennedy, R. S., Lane, N. E., Berbaum, K. S., and Lilienthal, M. G. (1993). Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. *The International Journal of Aviation Psychology*, 3(3):203–220.

[71] Kilteni, K., Bergstrom, I., and Slater, M. (2013). Drumming in immersive virtual reality: The body shapes the way we play. *IEEE Transactions on Visualization Computer Graphics*, pages 597–605.

[72] Kirk, D. and Stanton Fraser, D. (2006). Comparing remote gesture technologies for supporting collaborative physical tasks. In *Proceedings of the SIGCHI conference on Human Factors in computing systems - CHI '06*, page 1191, New York, New York, USA. ACM Press.

[73] Krämer, N. C., Karacora, B., Lucas, G., Dehghani, M., Rüther, G., and Gratch, J. (2016). Closing the gender gap in STEM with friendly male instructors? On the effects of rapport behavior and gender of a virtual agent in an instructional interaction. *Computers Education*, 99:1–13.

[74] Kucuk, S. and Bingul, Z. (2006). *Robot kinematics: Forward and inverse kinematics*. INTECH Open Access Publisher London, UK.

[75] Kulhavy, R. W., White, M. T., Topp, B. W., Chan, A. L., and Adams, J. (1985). Feedback complexity and corrective efficiency. *Contemporary Educational Psychology*, 10(3):285–291.

[76] Kurach, K., Raichuk, A., Stańczyk, P., Zajac, M., Bachem, O., Espeholt, L., Riquelme, C., Vincent, D., Michalski, M., Bousquet, O., et al. (2019). Google research football: A novel reinforcement learning environment. *arXiv preprint arXiv:1907.11180*.

[77] Lee, K., Lee, S., and Lee, J. (2018). Interactive character animation by learning multi-objective control. *ACM Transactions on Graphics (TOG)*, 37(6):1–10.

[78] Lester, J. C., Converse, S. A., Kahler, S. E., Barlow, S. T., Stone, B. A., and Bhogal, R. S. (1997). The persona effect: affective impact of animated pedagogical agents. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '97*, pages 359–366, New York, New York, USA. ACM Press.

[79] Li, J., Kizilcec, R., Bailenson, J., and Ju, W. (2016). Social robots and virtual agents as lecturers for video instruction. *Computers in Human Behavior*, 55:1222–1230.

[80] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

[81] Lim, S. and Reeves, B. (2010). Computer agents versus avatars: Responses to interactive game characters controlled by a computer or other player. *International Journal of Human-Computer Studies*, pages 57–68.

[82] Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier.

[83] Liu, L. and Hodgins, J. (2018). Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 37(4):1–14.

[84] Liu, L., Yin, K., van de Panne, M., Shao, T., and Xu, W. (2010). Sampling-based contact-rich motion control. In *ACM SIGGRAPH 2010 papers*, pages 1–10.

[85] Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Pieter Abbeel, O., and Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30.

[86] Lu, H., Gu, C., Luo, F., Ding, W., Zheng, S., and Shen, Y. (2020). Optimization of task offloading strategy for mobile edge computing based on multi-agent deep reinforcement learning. *IEEE Access*, 8:202573–202584.

[87] Mandlekar, A., Zhu, Y., Garg, A., Fei-Fei, L., and Savarese, S. (2017). Adversarially robust policy learning: Active construction of physically-plausible perturbations. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3932–3939. IEEE.

[88] Martey, R. M. and Consalvo, M. (2011). Performing the looking-glass self: Avatar appearance and group identity in Second Life. *Popular Communication*, 9(3):165–180.

[89] Mayer, R. E. and DaPra, C. S. (2012). An embodiment effect in computer-based learning with animated pedagogical agents. *Journal of Experimental Psychology: Applied*, 18(3).

[90] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

[91] Morimoto, J. and Doya, K. (2005). Robust reinforcement learning. *Neural computation*, 17(2):335–359.

[92] Nachum, O., Ahn, M., Ponte, H., Gu, S., and Kumar, V. (2019). Multi-agent manipulation via locomotion using hierarchical sim2real. *arXiv preprint arXiv:1908.05224*.

[93] Nachum, O., Gu, S. S., Lee, H., and Levine, S. (2018). Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31.

[94] Naderi, K., Babadi, A., and Hämäläinen, P. (2018). Learning physically based humanoid climbing movements. In *Computer Graphics Forum*, volume 37, pages 69–80. Wiley Online Library.

[95] Nelson, M. M. and Schunn, C. D. (2009). The nature of feedback: how different types of peer feedback affect writing performance. *Instructional Science*, 37(4):375–401.

[96] Neuron, P. (2018). *https://neuronmocap.com.* last accessed 2018/08/03.

[97] Nguyen, T. T., Nguyen, N. D., and Nahavandi, S. (2020). Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics*, 50(9):3826–3839.

[98] Nilim, A. and El Ghaoui, L. (2005). Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798.

[99] OpenAI (2020). Robogym. `https://github.com/openai/robogym`.

[100] Osborne, M. J. and Rubinstein, A. (1994). *A course in game theory*. MIT press.

[101] Pan, X., You, Y., Wang, Z., and Lu, C. (2017). Virtual to real reinforcement learning for autonomous driving. *arXiv preprint arXiv:1704.03952*.

[102] Park, C. and Manocha, D. (2014). Fast and dynamically stable optimization-based planning for high-dof human-like robots. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 309–315. IEEE.

[103] Pattanaik, A., Tang, Z., Liu, S., Bommannan, G., and Chowdhary, G. (2017). Robust deep reinforcement learning with adversarial attacks. *arXiv preprint arXiv:1712.03632*.

[104] Peng, X. B., Abbeel, P., Levine, S., and van de Panne, M. (2018). Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4):1–14.

[105] Peng, X. B., Berseth, G., Yin, K., and Van De Panne, M. (2017). Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 36(4):1–13.

[106] Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. (2017). Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826. PMLR.

[107] Plant, E. A., Baylor, A. L., Doerr, C. E., and Rosenberg-Kima, R. B. (2009). Changing middle-school students' attitudes and performance regarding engineering with computer-based social models. *Computers Education*, 53(2):209–215.

[108] Pridemore, D. R. and Klein, J. D. (1995). Control of Practice and Level of Feedback in Computer-Based Instruction. *Contemporary Educational Psychology*, 20(4):444–450.

[109] Puterman, M. L. (1990). Markov decision processes. *Handbooks in operations research and management science*, 2:331–434.

[110] Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., and Whiteson, S. (2018). Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4295–4304. PMLR.

[111] Rich, C. and Sidner, C. L. (1996). Adding a collaborative agent to graphical user interfaces. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 21–30.

[112] Rickel, J. and Johnson, W. L. (1997). Steve: An animated pedagogical agent for procedural training in virtual environments (extended abstract). *SIGART Bulletin*, 8:16–21.

[113] Ring, L., Utami, D., and Bickmore, T. W. (2014). The right agent for the job? - the effects of agent visual appearance on task domain. In *IVA*.

[114] Rosenberg-Kima, R. B., Baylor, A. L., Plant, E. A., and Doerr, C. E. (2008). Interface agents as social models for female students: The effects of agent visual presence and appearance on female students' attitudes and beliefs. *Computers in Human Behavior*, 24(6):2741–2756.

[115] Roth, D., Lugrin, J. L., Galakhov, D., Hofmann, A., Bente, G., Latoschik, M. E., and Fuhrmann, A. (2016). Avatar realism and social interaction quality in virtual reality. *In: Virtual Reality (VR)*, pages 277–278.

[116] Rozo, L., Calinon, S., Caldwell, D. G., Jimenez, P., and Torras, C. (2016). Learning physical collaborative robot behaviors from human demonstrations. *IEEE Transactions on Robotics*, 32(3):513–527.

[117] Ruhland, K., Peters, C. E., Andrist, S., Badler, J. B., Badler, N. I., and Gleicher, M., . (2015). Mcdonnell. *R. A review of eye gaze in virtual agents, social robotics and HCI: Behaviour generation, user interaction and perception. Computer Graphics Forum*, 34(6):299–326.

[118] Sansar (2017). *https://www.sansar.com.* last accessed 2018/08/03.

[119] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[120] Schwind, V., Knierim, P., Tasci, C., Franczak, P., Haas, N., and Henze, N. (2017). *These are not my hands!: Effect of gender on the perception of avatar hands in virtual reality.* In: 2017 CHI Conference on Human Factors in Computing Systems.

[121] Shang, X., Kallmann, M., and Arif, A. S. (2019a). Effects of correctness and suggestive feedback on learning with an autonomous virtual trainer. In *Proceedings of the 24th International Conference on Intelligent User Interfaces: Companion*, pages 93–94.

[122] Shang, X., Kallmann, M., and Arif, A. S. (2019b). Effects of virtual agent gender on user performance and preference in a vr training program. In *Proceedings of the Future of Information and Communication Conference (FICC).*

[123] Shang, X., Xu, T., Karamouzas, I., and Kallmann, M. (2023). Constraint-based multi-agent reinforcement learning for collaborative tasks. *Computer Animation and Virtual Worlds*, page e2182.

[124] Shiban, Y., Schelhorn, I., Jobst, V., Hörnlein, A., Puppe, F., Pauli, P., and Mühlberger, A. (2015). The appearance effect: Influences of virtual agent features on performance and motivation. *Computers in Human Behavior*, 49:5–11.

[125] Shute, V. J. (2008). Focus on Formative Feedback. *Review of Educational Research*, 78(1):153–189.

[126] Sklar, E. and Elizabeth (2003). Agents for education: when too much intelligence is a bad thing. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems - AAMAS '03*, page 1118, New York, New York, USA. ACM Press.

[127] Smith, H. J. and Neff, M. (2018). Communication Behavior in Embodied Virtual Reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*, pages 1–12, New York, New York, USA. ACM Press.

[128] Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., et al. (2017). Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*.

[129] Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J., and Vicente, R. (2017). Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395.

[130] Tessler, C., Efroni, Y., and Mannor, S. (2019). Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, pages 6215–6224. PMLR.

[131] Tickle-Degnen, L. and Rosenthal, R. (1990). The Nature of Rapport and Its Nonverbal Correlates. *Source: Psychological Inquiry*, 1(4):285–293.

[132] Tsai, W.-C., Lee, Y.-H., Chang, T.-H., Ho, C.-J., and Hsu, J. Y.-j. (2008). Designing human-computer multi-agent collaboration in productive multi-player games. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3*, AAMAS '08, page 1441–1444, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

[133] Van der Kleij, F. M., Feskens, R. C. W., and Eggen, T. J. H. M. (2015). Effects of Feedback in a Computer-Based Learning Environment on Students' Learning Outcomes. *Review of Educational Research*, 85(4):475–511.

[134] van Wissen, A., van Diggelen, J., and Dignum, V. (2009). The effects of cooperative agent behavior on human cooperativeness. In *AAMAS (2)*, pages 1179–1180.

[135] Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354.

[136] Vrancx, P., Verbeeck, K., and Nowé, A. (2008). Decentralized learning in markov games. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(4):976–981.

[137] Wang, X. V., Kemény, Z., Váncza, J., and Wang, L. (2017). Human–robot collaborative assembly in cyber-physical production: Classification framework and implementation. *CIRP annals*, 66(1):5–8.

[138] Wang, Y., Liu, H., Zheng, W., Xia, Y., Li, Y., Chen, P., Guo, K., and Xie, H. (2019). Multi-objective workflow scheduling with deep-q-network-based multi-agent reinforcement learning. *IEEE Access*, 7:39974–39982.

[139] Won, J., Gopinath, D., and Hodgins, J. (2021). Control strategies for physically simulated characters performing two-player competitive sports. *ACM Transactions on Graphics (TOG)*, 40(4):1–11.

[140] Wu, X., Li, X., Li, J., Ching, P., Leung, V. C., and Poor, H. V. (2021). Caching transient content for iot sensing: Multi-agent soft actor-critic. *IEEE Transactions on Communications*, 69(9):5886–5901.

[141] Xiao, C., Li, B., Zhu, J.-Y., He, W., Liu, M., and Song, D. (2018). Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*.

[142] Xie, Z., Ling, H. Y., Kim, N. H., and van de Panne, M. (2020). Allsteps: Curriculum-driven learning of stepping stone skills. In *Computer Graphics Forum*, volume 39, pages 213–224. Wiley Online Library.

[143] Xu, P. and Karamouzas, I. (2021). A gan-like approach for physics-based imitation learning and interactive character control. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 4(3):1–22.

[144] Xu, P., Shang, X., Zordan, V., and Karamouzas, I. (2023). Composite motion learning with task control. *Siggraph TOG*.

[145] Xu, T., Liang, Y., and Lan, G. (2021). Crpo: A new approach for safe reinforcement learning with convergence guarantee. In *International Conference on Machine Learning*, pages 11480–11491. PMLR.

[146] Yamane, K., Kuffner, J. J., and Hodgins, J. K. (2004). Synthesizing animations of human manipulation tasks. In *ACM SIGGRAPH 2004 Papers*, pages 532–539.

[147] Yang, H.-Y. and Wong, S.-K. (2019). Agent-based cooperative animation for box-manipulation using reinforcement learning. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 2(1):1–18.

[148] Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., and Wang, J. (2018). Mean field multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 5571–5580. PMLR.

[149] Yin, K., Loken, K., and Van de Panne, M. (2007). Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics (TOG)*, 26(3):105–es.

[150] Yin, Z., Yang, Z., Van De Panne, M., and Yin, K. (2021). Discovering diverse athletic jumping strategies. *ACM Transactions on Graphics (TOG)*, 40(4):1–17.

[151] Yin, Z. and Yin, K. (2020). Linear time stable pd controllers for physics-based character animation. In *Computer Graphics Forum*, volume 39, pages 191–200. Wiley Online Library.

[152] Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., and Wu, Y. (2022). The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624.

[153] Zanbaka, C., Goolkasian, P., and Hodges, L. (2006). *Can a virtual cat persuade you?: The role of gender and realism in speaker persuasiveness.* In: SIGCHI conference on Human Factors in computing systems.

[154] Zhang, H., Starke, S., Komura, T., and Saito, J. (2018). Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)*, 37(4):1–11.

[155] Zhang, K., Sun, T., Tao, Y., Genc, S., Mallya, S., and Basar, T. (2020). Robust multi-agent reinforcement learning with model uncertainty. *Advances in neural information processing systems*, 33:10571–10583.

[156] Zhou, K. and Doyle, J. C. (1998). *Essentials of robust control*, volume 104. Prentice hall Upper Saddle River, NJ.

[157] Zuo, Z., Han, Q.-L., Ning, B., Ge, X., and Zhang, X.-M. (2018). An overview of recent advances in fixed-time cooperative control of multiagent systems. *IEEE Transactions on Industrial Informatics*, 14(6):2322–2334.

Appendix A

Collaborative Virtual Humans Including Locomotion

This appendix presents a summary of an initial work performed (34) to enable two virtual humans to perform a collaborative manipulation task during locomotion. This work was not continued given the focus given in this dissertation on upper-arm movements instead of locomotion.

A.1 Summary

Using virtual trainers to assist users during direct manipulation tasks, in either simulated environments or physical environments, requires using some specific approach for achieving adaptive motion control. Previous work has demonstrated the effectiveness of using DRL for virtual trainers or robotic agents. In our work, we focus on applying the Deep Reinforcement Learning (DRL) methodology to two virtual trainers to collaboratively transport objects in a VR environment. We designed a task involving two virtual trainers collaboratively moving a tray from a random position to a target position in a dynamic environment with an object on top of the tray. The goal for the two virtual trainers is to carry the object to reach the target location while avoiding collisions with obstacles in the dynamic environment and keeping the object on the tray. Based on this design, we trained an efficient initial policy in this virtual environment, as illustrated in Figure A.1. This learned policy is expected to serve as an initial policy for this virtual trainer to collaborate with human users in an immersed VR environment as a human-agent interaction use case.

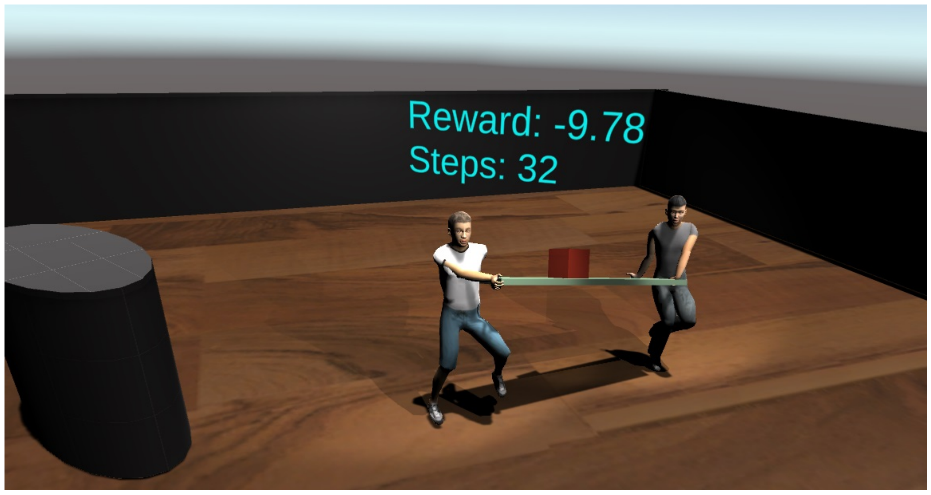


Figure A.1: Two virtual trainers collaboratively carry an object with a tray in a dynamic environment.

Appendix B

Composite Motion Learning with Task Control

This appendix presents the abstract of an additional work performed, as second author with external collaborators, on learning virtual human physics-based composite motions with deep learning methods (144).

B.1 Abstract

We present a deep learning method for composite and task-driven motion control for physically simulated characters. In contrast to existing data-driven approaches using reinforcement learning that imitate full-body motions, we learn decoupled motions for specific body parts from multiple reference motions simultaneously and directly by leveraging the use of multiple discriminators in a GAN-like setup. In this process, there is no need of any manual work to produce composite reference motions for learning. Instead, the control policy explores by itself how the composite motions can be combined automatically. We further account for multiple task-specific rewards and train a single, multi-objective control policy. To this end, we propose a novel framework for multi-objective learning that adaptively balances the learning of disparate motions from multiple sources and multiple goal-directed control objectives. In addition, as composite motions are typically augmentations of simpler behaviors, we introduce a sample-efficient method for training composite control policies in an incremental manner, where we reuse a pre-trained policy as the meta policy and train a cooperative policy that adapts the meta one for new composite tasks. We show the applicability of our approach on a variety of challenging multi-objective tasks involving both composite motion imitation and multiple goal-directed control.

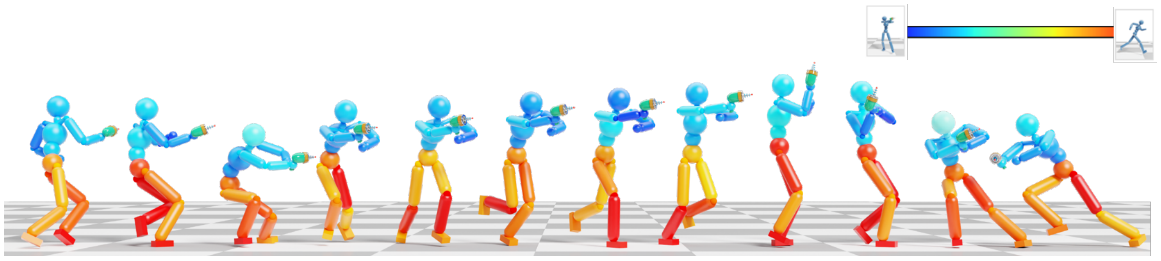


Figure B.1: Example of a physically simulated character performing composite motion with locomotion and aiming a weapon. The colors show the automatic mixing of the combined inputs that change dynamically over time based on the state. As indicated in the inset, red denotes body parts that are vital for locomotion while blue for aiming respectively. Our multi-objective approach learns this mixture along with imitation from two disparate reference motions and two goal-directed task rewards for each action.