Improving Human-Robot Collaborative Reinforcement Learning through Probabilistic Policy Reuse

by

Athanasios C. Tsitos

Submitted
in partial fulfilment of the requirements for the degree of
Master of Artificial Intelligence
at the
UNIVERSITY OF PIRAEUS

Supervisor Dr. Dagioglou
Second Examiner Prof. Vouros
Third Examiner Dr. Giannakopoulos

June 21, 2022

©University of Piraeus, NCSR "Demokritos". All rights reserved.

Abstract

Socially aware robots should be able, among others, to support fluent human-robot collaboration (HRC) in tasks that require interdependent actions in order to be solved. Similar to human-human collaboration, during HRC the actions of each agent affect the actions of its partner. Towards enhancing mutual performance, collaborative robots (cobots) should be equipped with adaptation and learning capabilities. Overall, mutual learning can be a time consuming procedure that depends on the computational complexity of the task, the motor and cognitive load demanded, as well as the skills of the human partner. Nevertheless, cobots should be able to integrate in their actions the capabilities of their human partner and adapt to their strengths and weaknesses. In the current thesis, we focused on HRC settings where a human and a Deep Reinforcement Learning (DRL) agent need to learn in real-time how to solve a shared task through efficient collaboration. In such scenarios, the performance of the team depends on one hand on the ability of the DRL agent to learn how to solve the task while adapting to its human partner and on the other hand on the ability of the human to understand the strengths and weaknesses of the agent and adapt accordingly. The goal of the thesis was to observe how the mutual performance could be improved when the agent needs to collaborate with different humans. The method used was a transfer learning technique called Probabilistic Policy Reuse, which allows DRL agents to take actions based on other pre-trained policies. In order to access this method, we developed a human-agent game where the human and a DRL agent controlled by the Soft Actor-Critic algorithm needed to jointly control the motion of the end-effector of a robotic manipulator and bring it to a goal position. For the experiments, we asked 16 different people to participate. Half of them played the game with a naive agent, meaning that the agent started to play without having any knowledge about the game, while the other half played the game with an agent, which had access to the actions of an expert agent that was trained beforehand by the author. In the second group, the agent took actions based on his current policy with a probability ψ and actions based on the expert policy with a probability 1ψ. The performance of the teams was evaluated through the travelled distance of the end-effector and the results showed that there was a significant difference between the performance of the teams which played without transfer learning and the teams that played with. This result indicates that applying transfer learning in HRC scenarios where the agent needs to collaborate with different humans might improve the mutual performance of the team.

Acknowledgements

First of all I would like to thank Dr. Maria Dagioglou for guiding me and helping me during the thesis as well as the RoboSKEL laboratory for giving me the opportunity to work on a real robot. Furthermore, I would like to thank my family for supporting me throughout my studies.

Any opinions, findings, conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the University of Piraeus and Institute of Informatics and Telecommunications of NCSR "Demokritos".

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1 From Human-Robot Interaction to Collaboration

Human-Robot Interaction (HRI) is an important field of robotics and has grown greatly in the last decades. Robots have the ability to execute fast, accurately and repeatedly tasks without decreasing performance. However, it is hard for them to reason and infer appropriate actions that will allow them to solve complex tasks. On the other hand, humans have increased cognitive skills which help them construct strategies for solving problems but lack at being consistent when executing repetitive actions mainly due to physical fatigue. HRI tries to combine the traits of each agent aiming at providing solutions to problems that are difficult to be solved by humans or robots alone.

There are different ways that an interaction between a human and a robot can be established. In [1], HRI is characterized as either *Instruction*, *Co-operation* or *Collaboration*. With the term *Instruction* the authors define HRI scenarios where the decision making is purely governed by the human. An example is Learning from Demonstration [3], where the human tries to teach the robot skills by means of kinesthetic teaching [4, 5], teleoperation [6, 7] or passive observation [8, 9]. In *Co-operation* on the other hand, the robot is not directly controlled by the human. Instead, each agent acts independently towards a final shared goal. The term "independetly" means that the actions of each agent do not affect the actions of its partner. A simple scenario of such an interaction is preparing a dinner where one might prepare the food while the other cleans the table. The subtasks are independent of one another and the actions of each agent are not affected by its partner. Lastly, in *Collaboration* there is a sequence of interdependent actions, meaning that each agent can potentially affect the actions of its partner indirectly through its own actions. In Fig. 1.1 there is a visual representation of this categorization.

Out of the three categories, the most beneficial is the Human-Robot Collaboration (HRC) because many tasks require interdependent actions in order to be solved. An example is tasks where physical collaboration is required. In these scenarios the human and the robot work in close proximity and the communication is established through interaction forces. This type of interaction is established in manufacturing [10, 11], where a shift from traditional robotic systems, which were operating inside protective cages, to collaborative robots (cobots) capable of operating in close proximity with the humans has been observed. Applications where cobots have been used in physical HRC include pick-and-place scenarios [12], wood polishing and heavy object carrying [13],

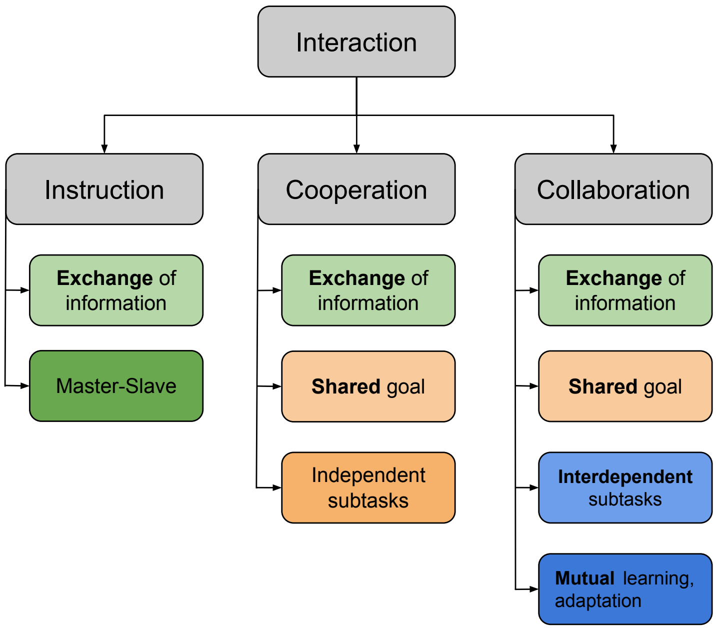


Figure 1.1: Categories of HRI [1]

assembly [14] and others.

As outlined in [1], HRC requires a pipeline for exchanging information between the human and the robot. From the robot's perspective, human information can be exploited either for establishing safety throughout the collaboration [10, 15] or for accessing the human actions and therefore making inferences about the human intentions [13]. The robot's information is important to the human in order to assess the quality of the collaboration and to observe whether the robot is executing its task correctly or needs assistance. This exchange of information can be achieved through different devices. Generally in HRI, devices such as joysticks [16], haptic devices [17], gloves [18], wearable inertial measurement units [19], exoskeletons [20] and motion capture systems [21] have been used. In HRC specifically, in [22] the human is monitored by an RGB camera and is communicating his intentions through hand gestures while in [23] a leap motion capture device is used. In the case of physical HRC such as in [12, 13] the robot is equipped with force/torque sensors, which enables the interaction to be established through forces.

Once the human information has been captured, the robot needs to make inferences about his intentions. In works where the humans motion is observed, their intentions are predicted by estimating their future movements. In [24], the authors propose a method for adaptation of robot trajectories, where humans and robots interact through physical forces and the robot adapts to the human intention by predicting his motion using the Fitts' law. A different approach for predicting the human motion is presented in [25], where the prediction lies on using a fifth order polynomial to fit on the recorded human motion. The experiment used to evaluate this method is a pick-and-place scenario, where the robot needs to predict the human motion and proactively choose which object to approach in order to complete the task as fast as possible. While predicting the human motions can lead to increased performance of the human-robot team, it also has a major drawback, which arises in situations where the human is uncertain about his next actions. Such uncertainty may yield predictions which are not in line with the human intentions. To cope with this issue, the authors in [26] take into account the uncertainty of a human prediction model within the interaction control framework. Specifically, they consider a Hidden Markov Model for the representation of the human motion behavior interpret any other human behavior deviating from this model as process noise. In [13], the authors present a framework for robot adaptation in task-level and apply it in physical HRI tasks. The human controls the motion of a robotic manipulator through interaction forces and after a short period of time the robot recognizes the task the human demonstrated task from a set of available tasks so that it can execute it on its own. In [12], the aforementioned idea is extended to enable the robot to converge to any task without having a predefined set. This is achieved by learning the parameters of a dynamical system which can produce the human demonstrated motion.

The aforementioned papers consist of state-of-the-art works in the field of HRC. However, they focus primarily on the robot learning and do not address the interaction from the human perspective. When agents collaborate it is important to observe the change in the behaviour and the actions of all partners [27], since in collaboration settings both the human and the robot need to adapt to each other's skills in order to cooperate efficiently. This mutual learning in HRC settings is referred to as **Human-Robot Co-Learning** in the literature [27, 28]. An introduction can be found in [27] where the definition of co-learning is established and the differences between co-learning, co-adaptation and co-evolution are presented. Furthermore, the authors define the requirements for studying how bi-lateral adaptation emerges from interactions between humans and robots and present the theoretical framework and methodological approach through an experimental study using a virtual robot for rescuing people trapped in debris. Another important study of co-learning can be found in [28], where the authors set the principles and challenges for successful human-AI co-learning in order to acquire the strengths of human-human teams while exploiting the benefits of intelligent technology. One aspect that the authors underline is the importance of focusing on the personal skills, strengths and weaknesses of the human partners. A relevant field in which research focuses on the personal capabilities of each participant in HRI is the *personalized tutoring* [29, 30]. This field studies if and how the learning procedure of humans can be improved by interacting with a robot-tutor. The main results indicate that assisting humans by focusing on their strengths and weaknesses, instead of not providing any aid at all or providing general aid, leads to increased learning performance of the humans. Therefore, it is important to observe if HRC can be improved by assisting the humans.

Many works around Human-Robot Co-learning research focus on how the performance of a human-robot team can be improved by mutual adaptation. In [31] the authors present an algorithm which allows robots to recognise human adaptive behaviours and guide them properly in order to achieve shared goals. The framework is evaluated through experiments where the team needs to collaborate for transferring objects. A similar work is presented in [32], where a framework which enables the robot to either adapt to the human intentions or guide the human to efficient actions is presented. The results show that enabling mutual adaptation leads to increased team performance.

While these works present impressive and important results in the content of Human-Robot Co-learning, they do not address the issue of real-time robot learning. A recent work which focuses on this aspect of the learning process is presented in [33]. The authors present a HRC setup between a robotic manipulator and a human user and develop a collaborative game which enables them to study real-time mutual learning in HRC settings. The results show that their method leads to successful training of the team. At the same time, it is observed that there exist significant differences in the overall human-robot team performance among different participants. This observation, along with the results from works regarding personalized tutoring, indicate the importance of focusing on the personal skills of each human in order to improve the performance of the team. However, in the context of Human-Robot Co-learning, focusing on the personal skills of each human has not been addressed in the literature yet.

2 Thesis Organization

In Chapter 2 we provide an overview of Machine Learning, Artificial Neural Networks, Reinforcement Learning and Deep Reinforcement Learning and discuss how Transfer Learning can be applied to Reinforcement Learning. In Chapter 3 we present works that have used Deep Reinforcement Learning in Robotics, discuss about the limitations of the current algorithms when applied specifically in robotic applications and present the motivation of the thesis. In Chapter 4 we present the methods used for evaluating the use of PPR in a HRC scenario and describe the experimental setup. The results are presented in Chapter 5. Lastly in Chapter 6 we summarize the findings of the thesis and discuss about the limitations and potential future work.

Chapter 2

Background

In this chapter we present the theoretical background for Deep Reinforcement Learning (DRL) and Transfer Learning (TL) in DRL. We first provide an overview of Machine Learning (ML) and discuss about the limitations which led to the development of Deep Learning. We then present a category of ML called Reinforcement Learning (RL), which in conjunction with Artificial Neural Networks (ANN), are the foundation blocks for DRL. Afterwards we present the DRL framework and focus on the Soft Actor-Critic algorithm (SAC). Finally, we discuss about the concept of Transfer Leaning (TL) and the techniques used for applying TL to DRL, while focusing on a technique called Probabilistic Policy Reuse (PPR).

1 Machine Learning

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on enabling systems to solve problems without being explicitly programmed. ML can be divided into three main parts [34]:

- "A decision process": A set of calculations based on input data that returns a "guess" or an action depending on the algorithm used.
- "An error function": A method of measuring how good the output of the decision process was.
- "An updating or optimization process": A procedure for ensuring that future guesses or actions will be better than the previous ones.

There are four main categories of ML algorithms, which depend on the type of the problem and the availability of the data:

- Supervised learning: Algorithms which aim at *prediction* or *classification* based on labeled data. Examples include the prediction of the human motion from 2D visual data [35].
- Unsupervised learning: Algorithms which aim at *prediction* or *classification* based on unlabeled data. An example is the exploration of a goal space for robots [36].

- Semi-supervised learning: It uses both labeled and unlabeled data and is mainly utilized in situations where the acquisition of a sufficient amount of labeled data is not possible. Semi-supervised learning has been used for modelling the human behaviour in HRC scenarios [37].
- Reinforcement learning: It is used in situations where an agents operates in an environment and needs to take actions. An example is real-time mutual learning between a human and a robotic manipulator [33].

The selection of the ML algorithm depends on many factors, the most important of which is the type of problem we are trying to solve. However, choosing the right algorithm does not guarantee a successful outcome on its own. An equally important aspect is the mathematical representation of the problem. For example, predicting the future motion of a mobile vehicle could be done based on its current velocity and acceleration. Other factors that affect the performance of the ML algorithms include the size of the dataset and the dimensionality of the representation, often called *feature space.*

Traditional ML showed promising results in problems where it was possible to extract useful features and the size of the feature space was relatively small. In problems however where the feature extraction is impossible or the dimensionality of the features is large, ML is unable to provide solutions.

Deep Learning (DL) and artificial neural networks (ANNs) are an extension of Machine Learning approaches which overcome the aforementioned limitations of the traditional ML. ANNs are designed to work similarly to the biological brain. The human brain consists of billions of neurons used to process input from the environment, such as vision, hearing, smelling e.t.c.. Every neuron processes the information it receives and propagates it to many other neurons which process the information they get.

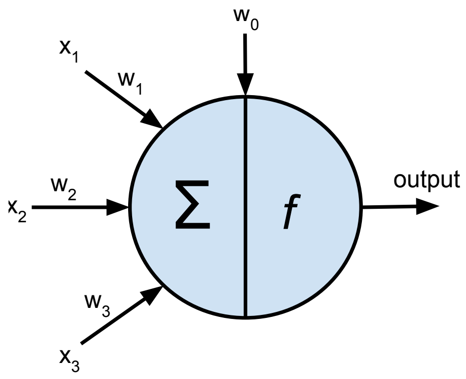


Figure 2.1: Artificial neuron. A weighted sum of the input is passed through an activation function to produce the output of the neuron. The x_i are the outputs of the previous neurons or the initial features and the w_i are the parameters of the neuron.

The artificial neuron, which consists of the building block of ANNs, is roughly simulating the biological neuron. Mathematically speaking, each neuron receives input from several neurons. This input is processed using a weighted sum and adding a bias term. This outcome then passes through an activation function. The output of the activation function is also the output of the neuron and is propagated to other neurons. A visual representation of a neuron is shown in Fig. 2.1. The output of a neuron can be written as $f(w_0 + \sum_{i=1}^{N} w_i \cdot x_i)$. An ANN is a collection of neurons as shown in Fig. 2.2. The neurons are structured in teams called *layers*. The first layers is called *input layer*, the last *output layer* while the rest are called *hidden layers*.

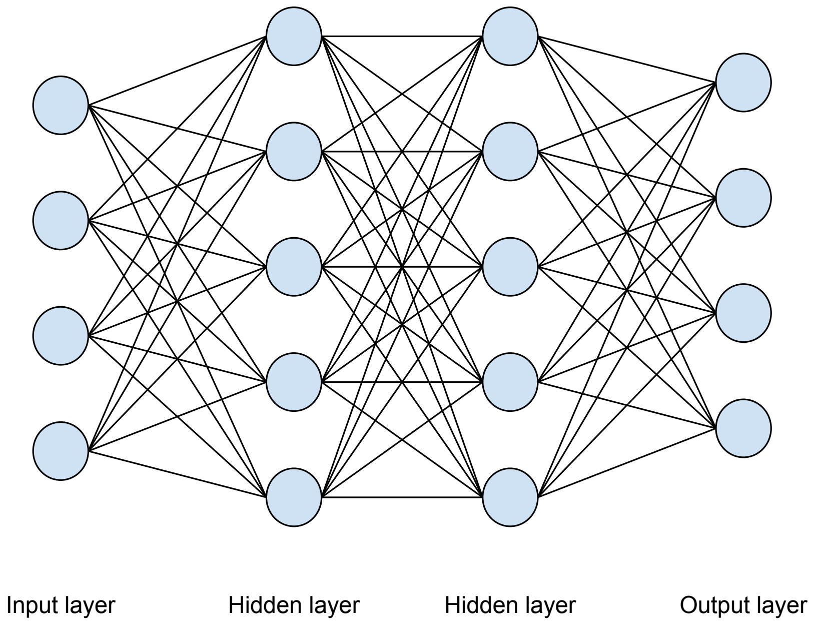


Figure 2.2: Artificial Neural Network. Each neuron receives input from all the neurons of the previous layers and propagates its output to all the neurons of the next layer.

Learning Procedure

The goal of the learning procedure is to learn the parameters w_i for each neuron. This is achieved by defining an objective function parameterized by w_i and trying to minimize it. The main steps of the learning procedure of a ANN are as follows:

- **Input**: The input of the ANN is a feature vector.
- **Feed-forward**: The feed-forward procedure produces the output of the ANN. Each neuron computes its output as explained in the previous section and propagates this output to the following neurons. The final result is the prediction $\hat{Y}$ of the ANN for the current input $\hat{Y} = f(x, w)$.

- **Loss**: The loss $L(w)$ is computed by comparing the predicted value $\hat{Y}$ with the groundtruth one Y[1]. The type of the loss function differs from task to task. However, a very common one is the Mean Square Error (MSE) and is defined as follows:
$$MSE(Y, \hat{Y}) = \frac{\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}{N}$$
where N is the total number of samples.
- **Backpropagation**: The next step is to update the parameters of the network in order to minimize the loss. The backpropagation algorithm uses the chain rule to update the parameters starting from the end towards the start of the network. The idea is to update the parameters towards the direction of the gradient of the loss [38]. A common way to update the parameters is to use the Stochastic Gradient Decent algorithm:
$$w_{i+1} = w_i + \alpha \cdot \nabla_{w_i} L(w_i)$$
where α is the learning rate and determines how much the algorithm alters the parameters.

Another important component of the ANNs are the activation functions. The activation functions are used to add a non-linear attribute to the neural network. Without the activation function, the output of each neuron (and therefore the output of the entire ANN) is just a linear regression of the input, which in most problems does not lead to a solution. The three most commonly used activation functions are the *Sigmoid*, *Tanh* and *ReLU*.

The sigmoid function is defined as follows:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

A neuron with a sigmoid activation function is basically a logistic regression model. Small and gradual changes in the input x produce small and gradual changes in the neuron's activation $f(x)$, but very small or very large values of x are mapped to 0 or 1, meaning that no significant change in the output is observed.

The tanh function is similar to the sigmoid with the difference that the output range in $[-1, 1]$ and is defined as follows:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.2}$$

Unlike the sigmoid function, tanh avoids the saturation issue is usually adopted in hidden nodes.

The third common activation function is the ReLU and is defined as follows:

$$f(x) = max(0, x) \tag{2.3}$$

[1]Note that the groundtruth value is not always known. In these cases (such as in DRL0 an estimated groundtruth value is used

ReLU functions only have positive firing mode and in practice it has been observed that they help ANNs learn faster than sigmoid or tanh.

Universal Approximation Theorem The Universal Approximation Theorem states that an ANN with a single hidden layer and with finite number of neurons can approximate any function. The practical use of the Universal Approximation Theorem is that it guarantees that any complex function can be estimated by a neural network.

2 Reinforcement Learning

2 .1 Markov Decision Processes

Reinforcement Learning is an area of Machine Learning where an agent learns how to interact with an environment in order to achieve specific goals [39]. It is widely used in problems which can be modelled as a **Markov Decision Process** (MDP). MDP is a discrete-time stochastic control process and is applied to situations where outcomes are partly random and is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$[2], where:

- $\mathcal{S}$: set of states which correspond to a representation of the environment.
- $\mathcal{A} = A(s)$: set of actions the agent can take at any given state.
- $\mathcal{T} = T(s, \alpha, s')$: probability that an action α at a state s will yield the state s'.
- $\mathcal{R} = R(s, \alpha, s')$: immediate reward for transitioning to state s' from state s by taking action α.

At each timestep $t = 0, 1, 2, ...$ the agent receives a state $s_t \in \mathcal{S}$ and then selects an action $\alpha_t \in \mathcal{A}$ which changes the state of the environment to $s'_{t+1} \in \mathcal{S}$. Furthermore, it receives a reward $r_{t+1} \in \mathcal{R}$ for taking the action α_t in the state s_t and transitioning to the state s_{t+1}. The goal of RL is to find a solution to problems which can be defined as a MDP. The solutions are mappings from states to probabilities of selecting possible actions, are called *policies* and are denoted as $\pi_t(\alpha|s)$. At this point, let us note that the policy can be either deterministic, meaning that for each state s, $\pi_t(a|s) \in \{0, 1\}$, or stochastic, meaning that $\pi_t(a|s)$ is a probability distribution.

The first step towards solving RL problems is to define them as a MDP, namely define the state space $\mathcal{S}$, the action space $\mathcal{A}$, the reward function $\mathcal{R}$ and the transition function $\mathcal{T}$. The second step is to define a way for the agent to construct a policy $\pi_t(a|s)$ which will enable him to solve the problem. The general idea is that actions which bring the agent closer to solving the task are considered to be good actions. From a mathematical perspective, finding such actions is achieved by trying to maximize the future expected reward. This maximization guarantees that the agent will converge to a policy and specifically to an optimal policy $\pi^*_t(\alpha|s)$, meaning that the agent will solve the problem in the most efficient way. The future expected cumulative reward is defined as

$$G_t = \sum_{i=t}^{\infty} \gamma^i \cdot R_{t+k+1} \tag{2.4}$$

[2]The notation used for the rest of the section is the one used in [39].

where $R_{t+k+1} = r_{t+k+1}$ is the reward that the agent received at the timestep $t+k+1$ and $0 < \gamma < 1$ is a discounting factor. The discounting factor is mandatory because otherwise the optimization might not be possible. This is because in the case where no discounting factor is used, the cumulative reward function would be $G_t = \sum_0^\infty R_{t+k+1}$ and in many settings this might tend to infinity ($G_t \to \infty$), resulting in trying to optimize using an unbounded function. The discount factor solves this issue since $G_t = \sum_{i=t}^{\infty} \gamma^i \cdot R_{t+k+1} \geq G_t = \sum_{i=t}^{\infty} \gamma^i \cdot R_{max} = R_{max} \cdot \sum_{i=t}^{\infty} \gamma^i = R_{max}/(1-\gamma)$. Furthermore, the discount factor implements the idea of focusing more on immediate rewards than future ones.

2 .2 Basic Concepts

Based on the knowledge of the agent about the environment, RL problems can be divided into two main categories. The first one is called *model-based* and it pertains to problems where the agent has a model of the environment. Based on this model, the agent can predict the outcome of his actions at any given state. In other words, in *model-based* RL the agent can construct the transition function $\mathcal{T}$. The action selection depends on the accuracy of the model and the more accurate the model is, the better the agent actions are. The second one is call *model-free* RL. In *model-free* RL the agent can not predict the effect of its actions to the environment and therefore can not construct a transition function $\mathcal{T}$. Instead, it follows a try-and-error procedure, meaning that it takes actions and observes how they alter the environment. By repeating this procedure, the agent distinguishes good from bad actions depending on whether they enable it to solve the task.

In order for this repetitive try-and-error process to be successful, the agent needs at first to collect enough information about the environment. This is typically achieved through a strategy named *Exploration-Exploitation.* During the learning process, the agent can follow two approaches for choosing which action to take at each state:

- Exploration: The agent chooses randomly an action.
- Exploitation: The agent chooses an action based on its current policy.

At the beginning of learning, the agent is encouraged to take random actions in order to collect information about the environment. As it gathers information and learns more, the agent decides to start taking actions based on its knowledge, namely based on its current policy. This procedure is also referred to as the *ϵ-greedy* algorithm. The exploration-exploitation trade-off is rendered by a parameter ϵ which is initialized to 1 (meaning that the agent will explore the environment) and slowly decays (meaning that the agent gradually starts to exploit his knowledge).

In the present thesis, we focus on the *model-free* RL. The algorithms used for solving *model-free* RL problems can be divided into three main categories: *Value-Function based algorithms* and *Policy based algorithms* and *Actor-Critic algorithms.*

2 .3 Model-free algorithms

Value-Function based algorithms estimate how good it is for an agent to be in any state s or how good it is to perform a given action a in a state s. The term "how good" is defined in terms of future expected rewards. Using the Bellman equations, the value of a state s under a policy π is defined as follows:

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}[G_t|s] \\ &= \sum_{a \in A} \pi(a|s) \sum_{s' \in S} T(s'|s,a)[R(s,a,s') + \gamma V^{\pi}(s')] \end{aligned} \tag{2.5}$$

Similarly, the value of taking an action a at a given state s is defined as follows:

$$\begin{aligned} Q^{\pi}(s,a) &= \mathbb{E}[G_t|s,a] \\ &= \sum_{s' \in S} T(s'|s,a)[R(s,a,s') + \gamma \sum_{a' \in A} \pi(a'|s') Q^{\pi}(s',a')] \end{aligned} \tag{2.6}$$

Most value-function based algorithms try to maximize the function $Q(s, a)$, namely to find the policy π^* which provides the best action a at any given state s in terms of maximizing the future expected reward. The most important issue of the maximization process is that $Q(s, a)$ is not differentiable and its optimal value is unknown. There exist different techniques which solve this type of optimization problems [40], like Dynamic Programming-based methods, Monte-Carlo methods and Temporal Difference Learning.

Policy based algorithms take a more direct approach than the ones based on the value-functions. Instead of trying to estimate a value in each state or a value of each state-action pair and then derive a policy, they try to directly construct a policy $\pi_t(a|s;\theta)$ based on some parameters θ. The idea is for the algorithm to learn these parameters by maximizing the expected reward. One important difference between the Value and the Policy based algorithms is the amount of memory they require. Value functions need to store the value of each state and the value of each state-action pair. Policy based algorithms, on the other hand, need to store only the values of the parameters θ, which in general requires less memory. For this reason, they are preferred over Value-based algorithms in cases where the state or the actions space have high dimensionality.

As mentioned earlier, Value-based algorithms fall short in cases where the state space is large or the action space is continuous. This is because exploring all possible actions using an ϵ-greedy strategy might require a prohibitive amount of time and the algorithm might converge to a local maximum instead of the optimal policy. The second issue arises in cases where the action space is a continuous space e.g. when training a self-driving car where the action is the angle that the agent needs to turn the wheel. The third and final issue is that the exploration strategy might not be sufficient. Suppose that in an environment the agent can take 10 different actions using an ϵ-greedy policy with $\epsilon = 0.1$. Assuming that the action 1 a_1 has the highest Q-value means that the agent has a chance of 91% to choose this action (90% using the ϵ-greedy policy and 1% to choose it randomly), while the rest actions have 1% chance of being selected. Therefore, in cases where $Q(s, a_1) = 3.01, Q(s, a_2) = 3, Q(s, a_{10}) = 0.01$, the probability of selecting the action a_1 is much larger than the probability of selecting action a_2 even though the respective Q-values are similar. Furthermore, the action a_2 has the same probability of being selected as the action a_{10} even though $Q(s, a_2) >> Q(s, a_{10})$.

One way to solve overcome the aforementioned problems is by using Policy-based algorithms. These algorithms skip the part of evaluating all the state-action pairs and instead learn directly a policy $\pi_t(a|s)$, thus avoiding all three problems described in the previous paragraph. The most important issue is the selection of the objective function that will be used in the training and which will estimate the policy π. A common choice is the total cumulative reward over an entire episode:

$$J = \sum_{\alpha} [Q(s_0, \alpha) \cdot \pi(a)] \tag{2.7}$$

which is actually the value function $V(s_0)$ at the initial state s_0.

Even though this approach solves indeed the issues of the Value-based algorithms, it introduces another problem which is related to the objective function. Consider the case where each step yields a reward of +10 and the 36^{th} step yields a reward of -100. The issue is that the agent has access only to the overall reward has no knowledge that the step 36 yield such a negative reward, meaning that the agent might not change its action at step 36 in future episodes.

A way to overcome this issue while maintaining the advantages of the policy-based algorithms is by combining the ideas of direct policy learning and evaluating the state-action pairs. Specifically, a network is used to predict the best action for each state (namely learn the policy) named *Actor* and a second network to evaluate this action selection (namely learn the Q-function) named *Critic*. The algorithms which fit this architecture are called **Actor-Critic algorithms**. The optimization techniques used for the *Actor-Critic* algorithms are the same as the ones listed in the presentation of the value functions. Out of these, the most common one is the Temporal Difference Learning (TDL).

TDL is an iterative procedure which updates the value of a state-action pair by taking a weighted average of the old value and the new information:

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot [r_t + \gamma \cdot max_a \cdot Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{2.8}$$

The “new information” is the immediate reward for taking the action a in the state s and the discounted future reward assuming that for any future state the agent acts optimally, namely he takes the best action. The parameter α is the learning rate and it determines how much we will take into account the new information. It has also been proven that this procedure converges to the optimal $Q^*(s, a)$ function even though the agent acts suboptimal in the meantime.

A widely used algorithm which belongs to TDL is the Q-learning algorithm. The main steps are summarized in Algorithm 1 [41]. In Q-learning, we first initialize the learning rate α, the discount factor γ and set the exploration ration ϵ to 1. Furthermore, we set the values of each state-action pair to 0 and define a decay factor E used for altering the exploration ratio (line 1). At the beginning of each episode, we set the current state to the initial state of the environment (line 3). Then, as long as the game has not ended (line 4), the agent first selects an action a at the state s either by exploring the environment (random action) or by exploiting his knowledge (the action is derived by the current policy) (lines 5-10). Then, he observes the new state s' and receives the reward r for taking the action a in the state s and transitioning to the new state s' (line 11). The next step is to update its current estimate of the state-action s, a pair based on the TDL update mechanism 2.8 (lines 12-13). The new state s' is then set as the current state (line 14). When the episode ends, the exploration ratio is decreased by the factor E which encourages the agent to depend on his own policy for the action selection as he moves to further episodes.

Algorithm 1 Q-learning

```
 1: Initialize:
        $\alpha - learning\,rate, \gamma - discount\,factor, \epsilon - exploration\,ratio$
        $Q(s,a) \leftarrow 0 \forall s \in S, a \in A$
        $0 < E << 1$
 2: for episode = 1 to max_episodes do
 3:     $s \leftarrow s_0$
 4:     while $s \neq s_{terminal}$ do
 5:         $random\_action = rand([0,1])$
 6:         if $random\_action > \epsilon$ then
 7:             $a \leftarrow choose\_random\_action$
 8:         else
 9:             $a \leftarrow \pi(s)$
10:         end if
11:         Take action a, receive reward r and next state s'
12:         $a'^* \leftarrow \max_{\tilde{a}}(Q(s', \tilde{a}))$
13:         $Q(s,a) \leftarrow Q(s,a) + \alpha \cdot (r + \gamma \cdot Q(s', a'^*) - Q(s,a))$
14:         $s \leftarrow s'$
15:     end while
16:     $\epsilon \leftarrow \epsilon - E$
17: end for
```

3 Deep Reinforcement Learning

Reinforcement Learning is used in problems where the state space $\mathcal{S}$ or the state-action space $\mathcal{S} \times \mathcal{A}$ is relatively small, e.g. a game where an agents moves in a finite grid and his actions are up, down, left, right. In this case, the $Q(s,a)$ function is stored in a tabular data structure, where each cell stores the value of a state-action pair. For problems, however, where the state-action space is large, it is impossible to store all the values $Q(s,a)$ in the memory. Suppose for example that we want to train an agent to learn how to play atari games as shown in Fig. 2.3. In the case where the input to the agent is the image of the game and the size of the image is 200×200 pixels, the dimension of the state space is $(256^3)^{200\times200}$, which is an insanely large number. It is therefore clear that storing information about so many states in a computer memory is impossible.

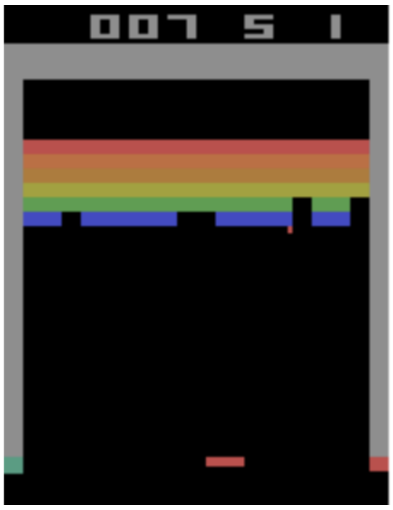


Figure 2.3: Atari game. The image is 200×200 RGB pixels where each channel takes a value in the interval $\{0, 1, 2, ..., 256\}$. The dimension of the state space is $(256^3)^{200\times200}$.

The solution to this problem was first introduced in [42]. The main idea is that instead of computing and storing in memory the values $Q(s, a)$, we try to learn an estimate $\tilde{Q}(s, a)$ of $Q(s, a)$. As the agent explores the environment, his estimation $\tilde{Q}(s, a)$ improves. Furthermore, the only thing that is stored in memory are the parameters of the state-action function and not the values for every state-action pair. The estimation $\tilde{Q}(s, a)$ is modeled as an artificial neural network. Furthermore, it is guaranteed by the Universal Approximation Theorem that an ANN is able to estimate the $Q(s, a)$ function.

3 .1 Deep Q-Network

The first Deep Reinforcement Learning algorithm was presented by a group of researchers at DeepMind in [43]. Its name is Deep Q-Network (DQN) and it is the first algorithm to combine Q-learning with neural networks. DQN was tested in Atari games and the results showed that the agent managed to learn how to win these games while having as input the raw 2D image and no other hand-crafted features. A neural network was used for estimating the $Q(s, a)$ function. Furthermore, the algorithm uses an ϵ-greedy policy for selecting the agent's action, like in the traditional RL. Lastly, the authors presented two novel ideas for improving the stability of their algorithm: *experience replay* and *frozen target network.*

Experience Replay The idea of *experience replay* is to store the agent's experience, namely the states s_t that it visited, the actions α_t that it took, the rewards r_t that it received and the states s_{t+1} that it transitioned in a buffer. During the training, a mini-batch of the experience is randomly sampled from the buffer and fed to the network. The main advantage of the *experience replay* is that the the training does not take place between consecutive inputs (i.e. in the atari games consecutive frames might be highly correlated) and therefore the network can learn without overfitting. Another advantage is that old experiences might be reused, which makes the learning smoother and more efficient.

Frozen Target Network The algorithm uses two networks with an identical architecture, but different parameter values: θ for the Q-network and θ^- for the target network. While the Q-network is updated every updated cycle, the target network stays frozen and is updated every C cycles by copying the parameters of the Q-network: $\theta^- = \theta$. This method induces a smoothing of oscillating policies and leads to more stabilized learning.

Soft Actor-Critic

Even though DQN led to success in cases where traditional RL could not provide solutions, it still presents the limitations of poor exploration of the Value-based algorithms. As described in 2 .3, the Actor-Critic algorithms overcome the limitations of Value-based and Policy-based methods. Therefore, the idea was to develop an Actor-Critic algorithm designed for the DRL framework. This lead to the creation of the **Soft Actor-Critic** algorithm. The architecture of the SAC is shown in Fig. 2.4. SAC is an algorithm that uses the ideas of *experience replay* and *frozen target network* introduced in DQN and trying to estimate not only the policy $\pi_t(a|s)$ directly but also the Q-function. As far as the optimization procedure is concerned, SAC seeks to maximize the entropy of the policy instead of just the future expected reward. The term"entropy" can be interpreted as how unpredictable a variable is; the higher its entropy, the more unpredictable it is. We want the SAC model to have high entropy in order to encourage exploration and to encourage the policy to assign equal probabilities to actions that have same or nearly equal Q-values.

The SAC uses three networks: a state value function V, a state-action function Q (Critic) and a policy function π (Actor).

- **Value Network**: The value network is trained by minimizing the following error:

$$J_V(\psi) = \mathbb{E}_{s_t \sim D}\left[\frac{1}{2}\Big(V_\psi(s_t) - \mathbb{E}_{a_t \sim \psi_\phi}\left[Q_\theta(s_t, a_t) - log\pi_\phi(a_t|s_t)\right]\Big)^2\right] \tag{2.9}$$

  The idea is that across all the states that we sample from the experience replay buffer, we want to minimize the squared difference between the prediction of our value network and the expected prediction of the Q function plus the entropy of the policy function π.

- **Critic (Q Network)**: The Critic is trying to estimate the state-action function

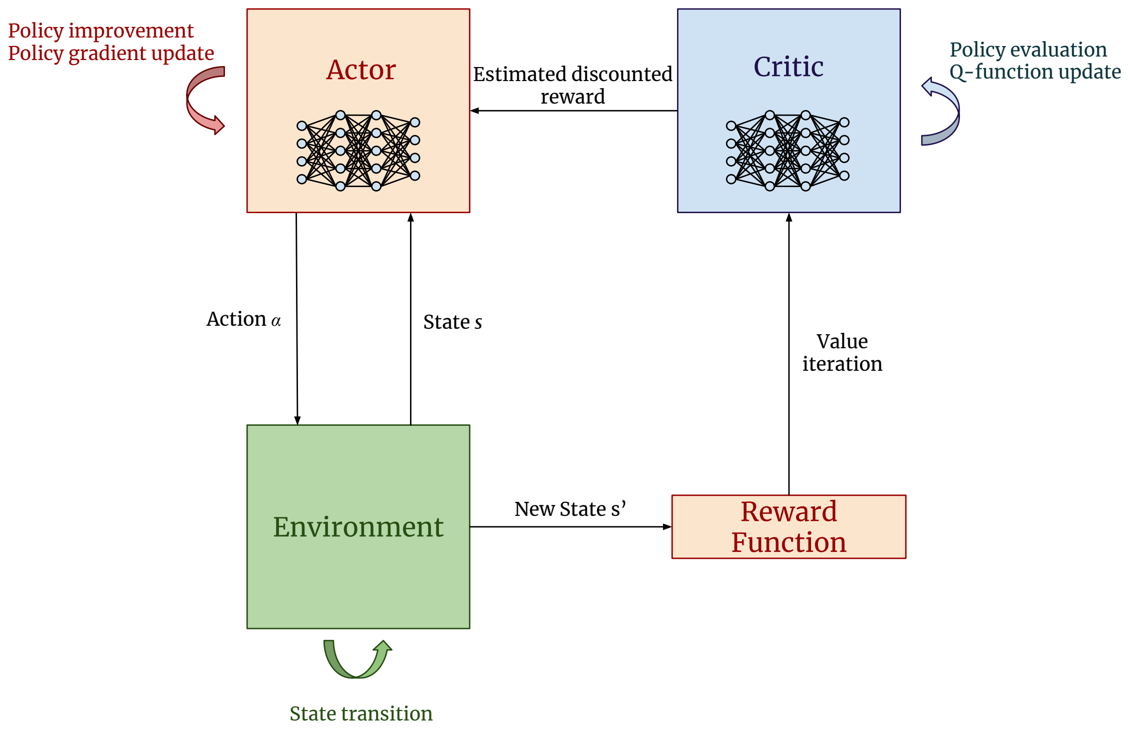


Figure 2.4: Architecture of the Soft Actor-Critic algorithm. The Actor decides which action a to take in each state s, namely it estimates the policy $\pi(a|s)$. The Critic evaluates the action that the Actor chose, namely it estimates the function $Q(s, a)$ [2]

$Q(s, a)$ and is trained by minimizing the following error:

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim D}\left[\frac{1}{2}\Big(Q_\theta(s_t, a_t) - \hat{Q}(s_t, a_t)\Big)\right] \tag{2.10}$$

where

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma \cdot \mathbb{E}_{s_{t+1}\sim p}\Big[V_{\bar{\psi}}(s_{t+1})\Big]$$

The first equation states that for all state-action pairs sampled from the experience replay buffer, we want to minimize the squared difference between the prediction of the the Q-function (Q_θ) and the "actual" Q-function which is the immediate reward granted for the state-action pair s_t, a_t plus the discounted expected value of the next state. Note that the value function $V_{\bar{\psi}}$ differs from the function approximated by the value network. By comparing the equations 2.9 and 2.10 we observe that the training of the value network depends on the Q-function and vice versa. This inner-dependency makes the training very unstable. The solution to this problem is the use of a *frozen target network* as in the DQN; We construct a second value function which parameters are the same as the original value function, but with a time delay. It has been shown that this approach overcomes the issue of the stability in the training.

- **Actor (Policy Network)**: The Actor is responsible for selecting which action the agent will take at any given state. In other words, the actor estimates the policy $\pi(a|s) \forall s \in \mathbf{S}$. The policy network is trained by minimizing the error:

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim D}\left[D_{KL}\left(\pi_\phi(\cdot|s_t) \Big|\Big| \frac{e^{Q_\theta(s_t,\cdot)}}{Z_\theta(s_t)}\right)\right] \tag{2.11}$$

  The D_{KL} function is called the Kullback-Leibler Divergence and it shows how different two distributions are. In order to minimize the error of two distributions, the authors in [44] propose a reparameterization trick to make sure that sampling from the policy is a differentiable process. The parameterization procedure of the action a_t is $a_t = f_\phi(\epsilon_t; s_t)$, where the ϵ is a noise factor. Basically, we model the action probability distibution to be a Gaussian distribution with a mean $\mu_\theta(s)$ and standard deviation $\sigma_\theta(s)$ and we try to learn these parameters. Using the reparameterization technique, the objective function is expressed as follows:

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim D, \epsilon_t \sim N}\Big[log\pi_\phi(f_\phi(\epsilon_t; s_t)|s_t) - Q_\theta(s_t, f_\phi(\epsilon_t; s_t))\Big]$$

4 Transfer Learning

Transfer Learning is an area in Machine Learning where knowledge regarding one problem (referred to as source problem) is utilized in order to solve a similar problem (referred to as target problem) [45]. For example, if we have trained a DRL agent to control the motion of a vehicle in the 2D space, then this model may be exploited to train another agent for navigating a vehicle in the 3D space [46].

When applying TL in Deep ANNs, the learning procedure is the same as the one discussed in Section 1 . The difference is that instead of starting from a model with random parameters, we use a pretrained model hoping that its parameters will be closer to the optimal parameters for the target problem. In this way, we accelerate the training procedure. In DRL, however, applying TL is not so straightforward because similar problems may differ in fundamental aspects such as the dimension of the state or action space. An example is the case where we have trained a 6-DoF robotic manipulator to solve a task and the goal is to teach a 7-DoF manipulator to solve the same task. Even though the task is the same, the difference in the robot embodiments (6-DoF to 7-DoF) alters the state space and therefore the trained model can not be utilized directly.

As mentioned in [45], there are five techniques for transferring knowledge in RL frameworks:

- Reward Shaping: The idea of reward shaping is to utilize knowledge in order to alter the reward function of the target task in order to accelerate training. An important characteristic of reward shaping is that it might change the convergence of the policy because it alters the reward function. However, in [47], the authors show how any function can be used to reshape the reward function while maintaining policy invariance.

- Learning from Demonstrations: The general idea is that provided demonstrations encourage the agent to explore states which will help him converge to a efficient policy faster. This can be achieved either “offline”, by learning the value function [48] or the model transition dynamics [49] or “online” aiming at direct efficient exploration [50].

- Policy Transfer: Previously learned policies are used to construct the new policy. One way to achieve this is by policy distillation [51], which means that the agent will select an action by minimizing the divergence of action distributions between the “teacher”(source) policies and the “student”(target) policy. Another approach is direct policy reuse [52], where the agent can select an action based on the pre-learned policy instead of his own policy.

- Inter-task Mapping: This method is not a standalone technique for transferring knowledge but rather enables other techniques to be applied when the source and target domains differ. Specifically, inter-task mapping methods define functions that map the source state space to the target state space, or the source action space to the target action space or any other RL component [53]. This way, pre-learned policies can be exploited for solving to a target problem that might share conceptual similarities with the source problem but differ in the MDP definition.

- Representation Learning: Representation learning aims at extracting features of the source problem which exist in the target problem as well. This is achieved by disentangling the state space, the action space or the reward space into task-invariant sub-spaces which are shared by both source and target domains.

In this thesis, we will focus on *Probabilistic Policy Reuse*. The reason why we chose to use this method is because it is the simplest approach to apply TL in DRL frameworks and because conceptually it makes sense to use it to our problem. The last point is made clear in Section 3 , where the motivation of our work and the conducted experiments are presented.

Probabilistic Policy Reuse: As described in Section 2 .2, a RL agent will probabilisticaly choose to either exploit the knowledge it has learned, or explore a random action (the exploration-exploitation trade-off). *Probabilistic Policy Reuse* adds a third option, which is the exploitation of a previously learned policy, often referred to as expert policy. Specifically, with probability ψ, an action is selected according to the old policy; with probability $1 - \psi$, a standard action selection mechanism, such as ϵ-greedy is used. ψ is decayed over time to allow the learner to emphasize new knowledge more as it learns more in the new task. This adds a bias to the exploration of the agent, intended to guide it towards good policies in the new task.

Chapter 3

Deep Reinforcement Learning in Robotics

In this chapter we present works that have used Deep Reinforcement Learning in robotics. We first present some works that have utilised DRL in order to solve robotic tasks. Then we discuss the limitations of DRL when applied to robotics and possible approaches that have been developed in order to overcome these issues. Finally, we address the use of DRL in HRC specifically.

1 Brief Overview

An overview of RL algorithms used in robotics is presented in [40]. However, the authors focus on RL algorithms because by the time of the writing, DRL was not yet applied to robotics. This gap is addressed in [54], where a survey of works that have applied DRL in robotic manipulation tasks is presented.

Deep Reinforcement Learning has been applied to many fields of robotics, such as mobile platforms [55, 56, 57], robotic arm control [58, 59, 60], robotic grasping [61, 62, 63], humanoids [64, 65, 66], drones [67, 68, 69] quadruples [70] and others. The reason behind this extensive use in different robot embodiments is because DRL policies can produce motions and behaviours that are extremely difficult to be generated by hard-coded control laws due to their complexity.

As far as mobile platforms are concerned, in [55] the authors use the Asynchronous Advantage Actor-Critic algorithm to enable a mobile robot to navigate without a map or a path planner and using data from a 2D laser scan and a RGB-D camera. The goal is for the robot to get to a predefined goal pose while avoiding static obstacles and is achieved by training the robot to a simulated environment and then deploying it to the real world. In [70], the authors present a DRL algorithm based on maximum entropy RL in order to teach a quadruple how to walk. An additionally interesting result is that the robot learns without having access to his dynamic model. In [63] the authors propose a Q-learning based network architecture for improving the grasping capabilities of a robotic manipulator using visual-based input from a multi-camera setup. For robot arm motion control, a Deep Q network has been used in [58] to enable a 3 DoF robotic arm to reach target configurations without prior knowledge of the goal and using only raw visual pixels as input to the network.

2 Limitations of DRL in Robotics

Even though DRL has gained success in the field of robotics, there are still many challenges that make its use impractical to some extent. The three main issues are *Sample insufficiency*, the *Exploration-Exploitation* trade-off and the *Generalization*.

Sample insufficiency means that DRL algorithms need large datasets in order to converge to a near-optimal policy. A possible solution to this issue is by using multiple robots simultaneously for the data collection [61]. In this work, the authors used 14 robots to collect data in order to train a grasp prediction model. This approach, however, is costly and in most cases infeasible due to the lack of hardware. Another way of solving the sample insufficiency problem is by training robots in simulation, which is faster and less costly, and transferring the learned policies to the real world. In [71], the authors present a survey of methods for transferring learned policies from simulated environments to the real world. The first method is called "Zero-shot Transfer" and refers to situations where the the simulation is very similar to the real world and the policies can be transferred directly. The authors in [72] evaluate this procedure through a reaching, a pushing and a sliding task executed by a robotic manipulator. However, in most cases, policies learned in simulation are not always efficient in real-world application due to the differences between the two worlds.

One of the first approaches to address the aforementioned issue is called "System Identification" [73] and aims at deriving a precise mathematical model of real-world systems and building simulations based on this model. This way, the simulation will be very similar to the physical system and approaches like "Zero-shot Transfer" might be applicable. However, in many cases this approach is not realistic due to the complexity of the system or the uncertainties of the real world. A third method used for sim-to-real transfer, which overcomes the limitations of the "System Identification", is called "Domain Randomization". In this approach, parameters of the simulation are parameterized randomly in order to cover the distribution of the real-world data. An example is [74], where an object detector is trained in a simulated environment under several different conditions like camera positioning or illumination and then tested in the real world without additional training for pick-and-place applications. Other applications where domain randomization has been applied include pose estimation [75] and semantic segmentation [76].

The second issue that DRL algorithms face is the Exploration-Exploitation trade-off. This concept has been introduced in Section 2 .2. The reason why this poses a challenge for applying DRL in robotics specifically is because completely random actions might lead to mechanical damage. A recent work which addresses this issue is [77], where the authors present methods so that safety principles can be incorporated in reinforcement learning. Safety has also been taken into account in real-world applications, like [78], where a neural network is used to predict the outcome of an action in terms of safety. This way, only safe actions are executed, leading to safe exploration. Another limitation of random exploration is that it can be a time consuming procedure due to the high dimensionality of the action space in robotic applications (e.g. an agent controlling the rotation of the wheel of a self-driving car). An example work which addresses this issue is [79], where demonstrated trajectories have been utilised as a bias that governs the learning procedure in early stages.

The final issue of DRL in robotics is the ability of the robots to generalize knowledge in order to operate to new, unknown circumstances and environments. Most works try to solve a RL task by training the robot from scratch. However, this approach is non-optimal mainly because it is time consuming. One way to overcome this issue is by transferring knowledge among conceptually similar tasks, just as humans do. The methods for applying transfer learning in DRL have been presented in Section 4 . In [80], the authors present an implementation of reward shaping in robotics in order to improve the training of an RL agent used for mobile navigation by altering its reward function based on the knowledge about the map provided by the SLAM algorithm. Learning from demonstration has been applied in [79] where the issue of sparse reward function is a pick-and-place scenario is addressed. The use of demonstrated trajectories for efficient exploration enables the agent to solve the task, which might have been infeasible with random exploration.

Pre-learned policies have also been used for training DRL agents. For example, policy distillation has been applied in robotics in a continual learning problem [81], where the goal is for a single RL agent to learn three different policies for three different navigation tasks and learn which policy to use by identifying in real time the task to be solved. The second approach is direct policy reuse [52] and the agent can select an action based on the pre-learned policy instead of his own policy. This idea has been used in [66], where the authors teach a humanoid robot how to walk fast by exploiting a policy that allows the robot to walk in a normal speed. Knowledge has also been transferred between morphologically different robots, like in [82], where the authors train a 3-link robotic manipulator in three different tasks (target reaching, peg insertion, and block moving) and exploit the policies in order to train a 4-link one. Finally, in the context of representation learning, some works such as [83] show how to reuse the extracted features for transferring knowledge, while other such as [84] focus on the feature extraction. An application of representation learning in robotics is presented in [85], where the authors show how extracting important features from the environment can accelerate the learning procedure of an RL agent in the case of slot car racing and mobile robot navigation.

3 Motivation

Although there have been works which address the problem of collaboration in Human-AI teams such as [86, 87, 88], the work around DRL in Human-Robot teams in still sparse. A recent work which focuses on this aspect of the learning process is presented in [33]. The authors present a HRC setup between a robotic manipulator and a human user and develop a collaborative game which enables them to study real-time mutual learning in HRC settings. The results show that their method leads to successful training of the team. At the same time, it is observed that there exist significant differences in the overall human-robot team performance among different participants. This observation, along with the results from works regarding personalized tutoring, indicate the importance of focusing on the personal skills of each human in order to improve the performance of the team.

As explained earlier, an important aspect of the learning procedure is the overall training time. In the context of mutual learning in HRC scenarios, different humans might need different training times in order to achieve a certain performance or might achieve different levels of performance in the same training time [33]. In this work we focus on real-world HRC settings which depend on real-time mutual learning and adaptation and study how they can be extended in order to improve the learning performance of the team among different humans. Specifically, we suppose that a human needs to collaborate with a robot in order to learn in real-time how to solve a task. The robot is controlled by a Deep Reinforcement Learning agent, who at the beginning of the collaboration does not have any knowledge about the task. The main idea is to observe whether transferring knowledge by a pre-trained expert agent can improve the overall performance of the team. In the RL framework, one way to achieve this is by using Probabilistic Policy Reuse (PPR), which enables the agent to have access to the actions of an expert agent. The method is evaluated by conducting experiments with different humans. The experiments are divided into two groups. In the first one, no Transfer Learning (TL) is applied while in the second one the PPR technique is used. The results show that the second group managed to outperform the first group in terms of efficient collaboration and training time. The main contributions are as follows:

- Design of a HRC task between a human and a DRL agent.
- Integration of the PPR method.
- Integration of the system into ROS
- Study with eighteen human-robot teams.

Chapter 4

Research Method

In this chapter we present a human-robot collaborative task that was developed in order to conduct experiments and try to answer the research questions stated in the Introduction. First we describe the robotic setup and the formulation of the MDP which will be tackled by a SAC DRL agent. Then we provide some technical implementation details. A detailed presentation of the implementation can be found in the Appendix. The game is integrated into ROS [89], which is an open-source middleware widely used for the development of robotic applications and the code can be found online[1]. Finally, we provide details about the experimental setup and the user study.

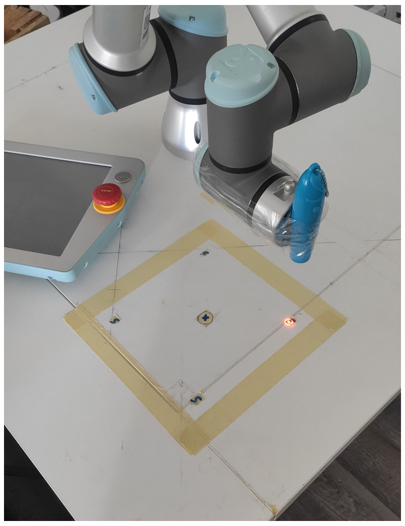


Figure 4.1: Robotic setup. The robot is placed in the middle of a 1m x 1m table. A laser is attached to the EE of the robot in order to provide to the human visual feedback about the position of the EE. The feedback is the red dot on the table.

[1]https://github.com/ThanasisTs/human_robot_collaborative_learning

1 Human-Robot Collaborative Game

In the HRC game, the team consists of a human and a Universal Robot UR3, which is a non-redundant 6-DoF robotic arm. The robot is placed in the middle of a 1m x 1m table and its EE is placed perpendicular to the table and can move parallel to it at a certain height (Fig. 4.1). Furthermore, a laser is attached to the EE and points towards the table (red laser dot). The human is responsible for controlling the motion of the robot in one axis (y-axis), while a DRL agent controls the motion of the robot in the perpendicular axis (x-axis). By combining the motions of the two partners, the EE can move in the xy plane (plane of the table's surface). The motion of the robot in the xy plane is confined inside a 20cm x 20cm square. The goal of the team is to jointly control the position of the laser dot inside the square and specifically to bring the the dot to the goal position, which is in the middle of the square (Fig. 4.2). The team wins the game if it manages to bring the laser dot inside the circle of the goal position with a relative low speed within a time window of 30 secs, otherwise the team loses. The coordinates of the goal position were $[-0.264, 0.242]$ with respect to a reference frame at the base of the robot, while the position and velocity tolerances (maximum distance from the goal position and maximum speed in order for the team to win) were 0.01m and 0.05m/s respectively.

The human-robot team played 150 games in total. At the beginning of each game, the robot chose randomly an initial position out of the four possible ones and the EE was automatically placed on top of it. The distance between the starting positions and the goal position was $0.12m$. Once it reached the starting position, a sequence of three short and one long ”beeps” indicated the start of the game. In both cases, a different sound was generated which informed the human about the outcome of the game. Furthermore, the outcome as well as the score of the team was visualized in a computer monitor (Fig. 4.3).

2 Reinforcement Learning agent

The motion of the robot in the x- axis is controlled by a SAC agent. As explained in Section 2 .1, RL (and SAC essentially) is modelled as a MDP. In our case, the MDP is defined as follows:

- $\mathbf{S} = \{ee\_pos\_x, ee\_pos\_y, ee\_vel\_x, ee\_vel\_y\}$: The observation space is the EE position and velocity.
- $\mathbf{A} = \{-1, 0, 1\}$. The actions the agent can take correspond to the acceleration in the x- axis.
- $\mathbf{R}$: At each timestep, the agent receives a reward of -1 if the EE transitioned to a non-goal state and 10 if it reached the goal.

The parameters used in the SAC algorithm are presented in Table 4.1 and were based on [33]. Furthermore, the implementation was based on [90], where the author presents some which allow SAC to handle discrete action spaces.

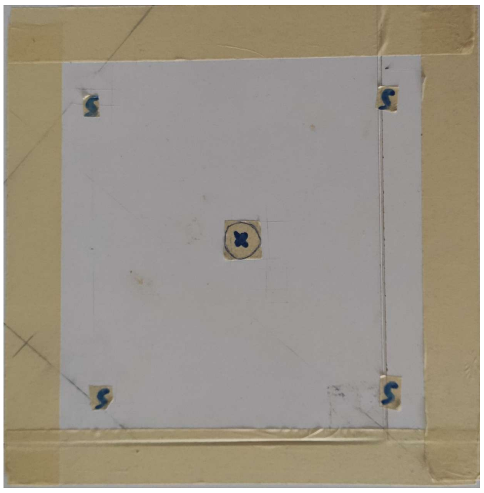


Figure 4.2: Rectangle inside of which the red dot can move. The initial positions are in the four corners denoted with the letter ”S” and the goal position is in the center of the rectangle denoted with the symbol ”X”. The team wins if the red dot gets inside the circle around the goal position with a relatively slow speed.

SAC Parameters	Values
size of 1^{st} hidden layer	32
size of 2^{nd} hidden layer	32
γ	0.99
τ	0.005
α	0.0003
β	0.0003
target_entropy_ratio	0.4
buffer_max_size	1000000
batch_size	256

Table 4.1: Parameters of SAC algorithm

No Transfer Learning

In case the SAC algorithm is used without TL, the selected action can be either a random action or an action derived by the current policy. The action selection procedure is

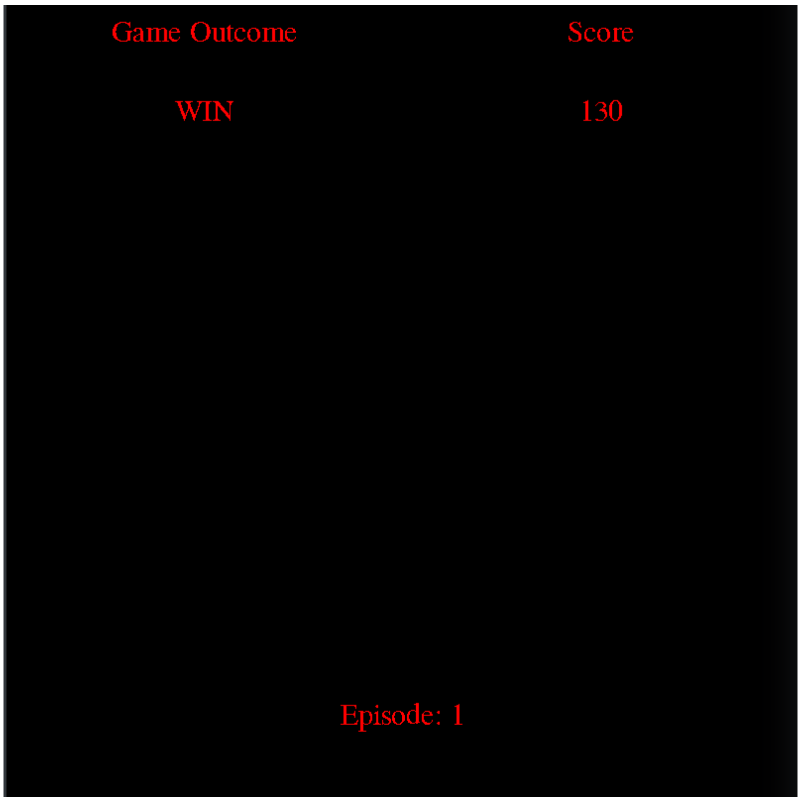


Figure 4.3: Visualization of the score of a game. The outcome of the game (”Win” or ”Lose”) is visualized in the top left corner. The score is shown in the top right corner while the number of the game is shown in the bottom.

summarized as follows:

$$No\_TL\_\alpha(i, i_N) = \begin{cases} random([-1, 0, 1]) & i \leq N \\ \arg\max_\alpha \pi(\alpha|s) & i > N \end{cases} \tag{4.1}$$

where i is the number of the current game, α is the selected action for the i^{th} game, $random[-1, 0, 1]$ means that a random action out of the possible actions is selected, $\arg\max_\alpha \pi(\alpha|s)$ means that the action derived by the policy is selected and N is the game where the action selection strategy changes.

Probabilistic Policy Reuse

In the case of TL, a pre-trained “expert” agent is used. According to a probability ψ_{ppr}, the agent can either select an action based on the “No Transfer Learning” procedure or select the action derived by the policy of the “expert” agent. This idea is summarized below:

$$TL\_\alpha(i, N, \psi, \psi_{ppr}) = \begin{cases} No\_TL\_\alpha(i, N) & \psi \leq \psi_{ppr} \\ \arg\max_{\alpha} \pi_{expert}(\alpha|s) & \psi > \psi_{ppr} \end{cases} \tag{4.2}$$

where ψ is a random number in the interval [0, 1], ψ_{ppr} the PPR threshold and π_{expert} the policy of the expert agent.

3 Experimental Setup

3 .1 User Study

In order to validate the use of PPR in the HRC game, we asked 16 different people to participate in the experiments. Out of the participants, 8 played without TL, while the rest played with PPR. The expert agent used for applying PPR was trained beforehand by an expert user who had been trained on the game for almost 10 hours. The percentage of the participants regarding the gender, the experience with Artificial Intelligence and with Robotics are shown in Table 4.2 while the average age was 29.2 with standard deviation 4.8.

Male	Female	AI	No AI	Robotics	No Robotics
43.8%	56.2%	43.8%	56.2%	25.0%	75.0%

Table 4.2: Gender, experience with Artificial Intelligence and with Robotics of the participants

At the beginning of each experiment, a letter was given to the participants which informed them about aspects of the experiment, such as the fact that their involvement was voluntary and that no information about the experiments will be used against their will. After that, they provided their written consent. The information letter as well as the consent form are reported in the Appendices B and C . The study protocol was approved by the Research Ethics Committee (REC) of NSCR "Demokritos".

Instructions

The instructions for the experiments were given by the author of the thesis, who was present throughout the entire experiment. The participants were informed about the nature of the collaboration with the robot, the axis they were controlling (y- axis), the total number of games (150), the two possible outcomes of each game (”win” or ”lose” as well as the respective sounds), the visualization module and the fact that the games were divided into groups of 20. Furthermore, the participants were told that the robot movements were confined in the rectangle (Fig. 4.2) and that it was safe to operate with the robot in close proximity not only due to the kinematic constraints imposed by the design of the experiment but also because of the safety button in the polyscope of the UR3, which shuts down the robot. The participants were not informed that a DRL

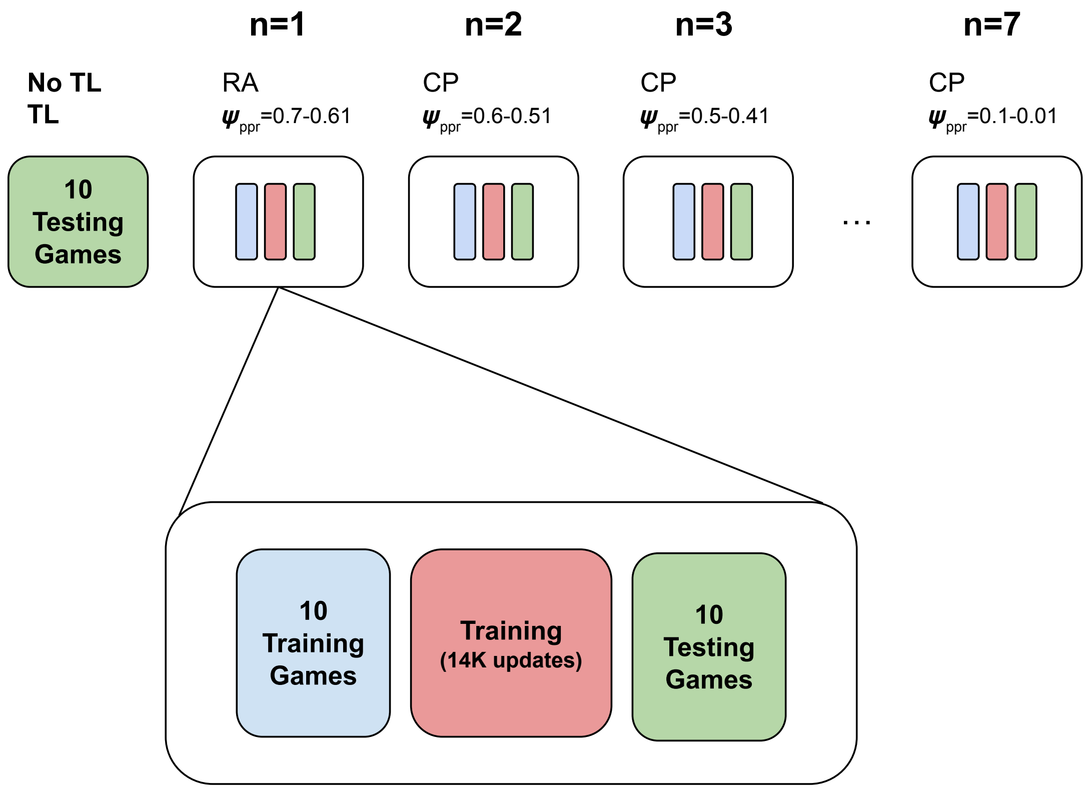


Figure 4.4: Pipeline of the experiments. The first 10 test games are played with the RL agent picking random actions (RA). In the first 10 train games, the agent selects either random actions if no TL is applied or he uses the action selection procedure (4.2) with $\psi_{ppr} = (0.7 - 0.61)$ depending on the game. For the rest train games, the agent selects an action based on his current policy (CP). In each training there are a total of 14000 updates.

agent was responsible for the EE motion in the perpendicular axis nor that after each 20-group of games a training was occurred. Regarding the latter, they were told that a 3-minute break existed after 20 games so that they could rest.

As far as the instruction about the robot control are concerned, the participants were told that they could control the motion of the robot through the keyboard and specifically using the keys “i”, “k”, “,”. The instruction for each key was given as follows:

- “i”: The participant can move the EE away from him.
- “,”: The participant can move the EE towards him.
- “k”: By pressing the “k” button, the participant commands the EE to continue moving the exact same way as it was moving the moment he pressed the button.

3 .2 Experimental Setup

The 150 games were divided into groups of 20 games. At the end of each group, the RL agent was trained with 14000 gradient updates. Each group was then divided into two subgroups of 10 games each (Fig. 4.4). The first 10 games were used to test the last trained model while the second 10 were used to store data for the upcoming training. Therefore, there were 80 testing and 70 training games in total. In the testing games, the agent was sampling actions from its current policy whether TL was used or not. In the training games, however, the action selection procedure depended on whether TL was applied or not and was formulated based on 4.1 and 4.2. In 4.1, the parameter N was set equal to 10. This means that for the first 10 training games the agent was selecting random actions while for the rest 60 he was sampling from his current policy. The parameter ψ_{ppr} in 4.2 was set initially to 0.7 and after each training game it was decaying with a ratio of $\psi_{\epsilon} = 0.01$. This training-testing procedure is shown in Fig 4.4, while the parameters of the game are shown in Tables 4.3-4.4.

Game Parameters	Values
training games	70
testing games	80
maximum game duration	30 secs
duration of RL agent action	0.2 secs
start training after N training games	10
train the agent every N training games	10
games with random actions	10
total training cycles	98000
win reward	10
time penalty	-1
initial ppr probability	0.7
ppr probability decay ratio	0.01

Table 4.3: Parameters of game

Robot motion Parameters	Values
goal position tolerance	0.01 m
goal velocity tolerance	0.05 m/s
maximum velocity in x- axis	0.2 m/s
maximum velocity in y- axis	0.2 m/s

Table 4.4: Parameters of robot motion

Familiarization

One important feature which might affect the performance of the teams was the difference in the inherent abilities of the participants to control the motion robot in their axis. During the collaboration, it is very difficult to conduct this assessment because the motion of the robot in the agent's axis might affect the actions of the humans. For example, playing the game with a naive agent and with an expert one might result in different human behaviours. To overcome this limitation, a total of 7 games were played before the 150 games described above. In these 7 games, the EE was able to move only to the axis controlled by the human. At the beginning of each game, the EE was automatically placed in the initial position (Fig. 4.5). Once the game started, the participant had a time window of 10 secs to achieve the goal which was to bring the red dot to the goal position with a relatively slow speed. This way, we were able to evaluate the ability of the human to control the robot motion. The goal position and the positional tolerance were the same as in the HRC game but the velocity tolerance was set to 0.02m/s.

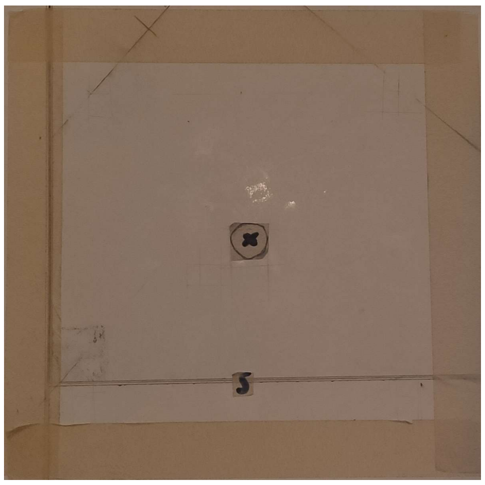


Figure 4.5: Rectangle for the familiarization games. The "S" symbol denotes the initial position of the EE while the "X" symbol the goal position. The EE can move in the line connecting the two symbols.

4 Robot Control

4 .1 Human and RL control

Both the human and the RL agent control the motion of the EE in their respective axis by providing commanded accelerations. The commanded accelerations are then numerically integrated to commanded velocities. Therefore, the control design consists of a feedforward term on the acceleration as follows:

$$\ddot{\mathbf{x}}_{com} = u \tag{4.3}$$

$$\dot{\mathbf{x}}_{com} = \dot{\mathbf{x}}_{com} + T_c * \ddot{\mathbf{x}}_{com} \tag{4.4}$$

where u is the desired acceleration imposed by the human or the RL agent, $\ddot{\mathbf{x}}_{com}$ is the commanded acceleration, $\dot{\mathbf{x}}_{com}$ is the commanded velocity and T_c is the control cycle. In our case $T_c = 0.008s$ because the robot controllers operate at 125Hz.

As shown by the control laws, both partners control the robot through commanded accelerations. In the HRC game, they can apply three discrete accelerations:

- an acceleration of $+\alpha\, m/s^2$
- an acceleration of $-\alpha\, m/s^2$
- an acceleration of $0\, m/s^2$

where $\alpha > 0$ is a positive constant. This kind of control imposes an inherent difficulty to the game for the human. In this way, we enforce the concept of mutual learning, since the human needs to learn how to control the motion of the robot while collaborating and potentially adapting to the DRL agent.

4 .2 Reset

A feedback control law on the position of the EE in used before the start of the game in order for the EE to get to an initial position. The control law is defined as follows:

$$\dot{\mathbf{x}}_{com} = K_P * (\mathbf{x}_{des} - \mathbf{x}_{curr}) \tag{4.5}$$

where $\mathbf{x}_{des}$ is the desired position of the EE, $\mathbf{x}_{curr}$ its current position, K_P a 2x1 gain matrix and $\dot{\mathbf{x}}_{com}$ is the commanded velocity. In our case, $K_P = [1 \quad 1]^T$.

The motion of the robot is regulated by commanded EE velocities. The EE velocities are then mapped to joint velocities using the Inverse Kinematics algorithm, which are then passed to the robot controllers for executions.

5 Technical Implementation

The entire system is implemented in ROS and tested on Melodic and Ubuntu 18.04. ROS is a middleware which provides the necessary functionalities for developing robotic applications and is widely used mainly due to its large community. The main building block of ROS is the *node*, which is an entity that provides a certain functionality. The nodes communicate with each other by exchanging information, which is passed as ROS messages. The communication was achieved through *topics*. Topics are named buses. A node (called Publisher) sends information to a topic and multiple nodes (called Subscribers) have access to this information. Topics are mainly used for exchanging streams of data. Their main drawback is that they do not deal with the issue of data loss (published data that did not reach the subscriber). Note that a node can both publish and receive data.

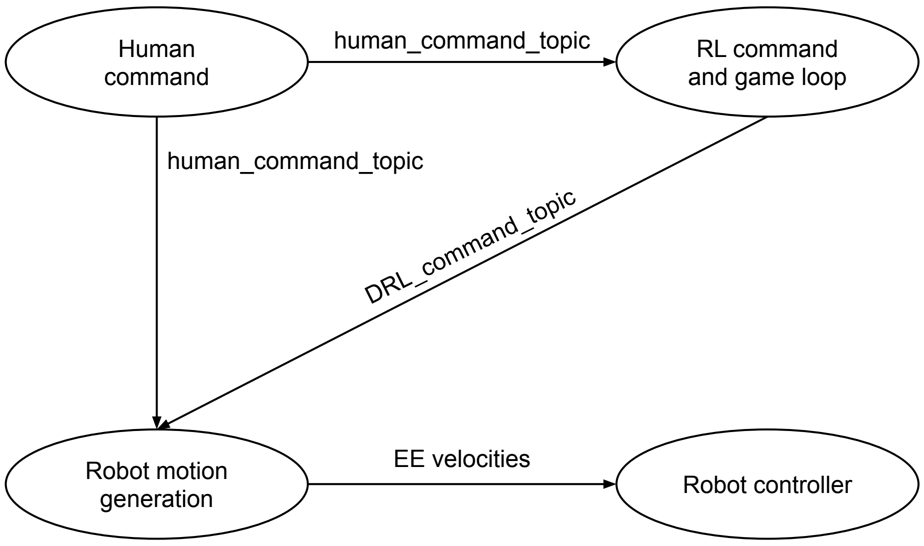


Figure 4.6: Simplified ROS graph

A simplified graph of the ROS system which shows the nodes and the communication pipeline of the HRC game is presented in Fig. 4.6. The entire ROS graph (called rqt graph) as well as a detailed presentation of the ROS implementation is presented in the Appendix A . The components of the graph are presented as follows. Note that all the nodes except for the *Robot controller* node were developed during the thesis.

- *Human command*: The human controls the motion of the EE in the y- axis through a keyboard. A node listens to keyboard input and publishes ROS messages which correspond to the human desired acceleration. Specifically the human can press the buttons “i”, “k” or “,”. The “i” button applies an acceleration of $+0.4\, m/s^2$, the “,” button an acceleration of $-0.4\, m/s^2$ and the “k” button applies zero acceleration. These values were chosen experimentally. In Fig. 4.6, this functionality is encapsulated in the node named “Human command”.

- *RL command and game loop*: The node “RL command and game loop” provides the action of the agent and the loop of the game. The agent action is sampled from the policy provided by the SAC algorithm. The implementation of the algorithm for discrete action settings was found online[2] and was integrated into ROS.

- *Robot motion generation*: The node "Robot motion generation" accepts the human and the agent commanded accelerations and implements the feedback control law described in Section 4 . Furthermore, it provides the functionality for resetting the robot described in Section 4 .2. Its output are the EE commanded velocities, which are sent to the "Robot controller" node. The EE velocities are mapped to joint velocities through Inverse Kinematics, which are then executed by the robot.
- *Robot controller*: The robot controller is the node which implements the ROS interface for controlling the UR3.

[2]https://github.com/Roboskel-Manipulation/maze3d_collaborative

Chapter 5

Results

In this chapter we present the results of the familiarization and the game experiments described in Chapter. 4.

1 Familiarization

In Fig. 5.1 we present the number of wins in the familiarization games. We observe that all of the participants manages to win at least 2 times while most of them managed to win at least 4.

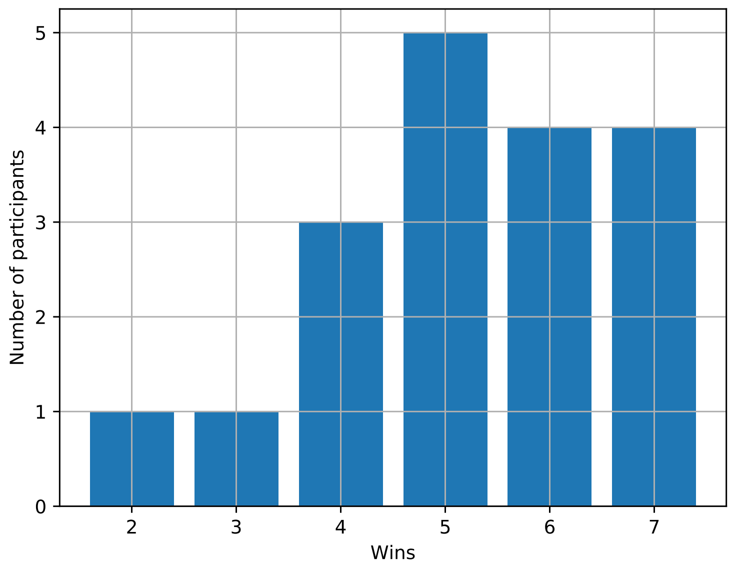
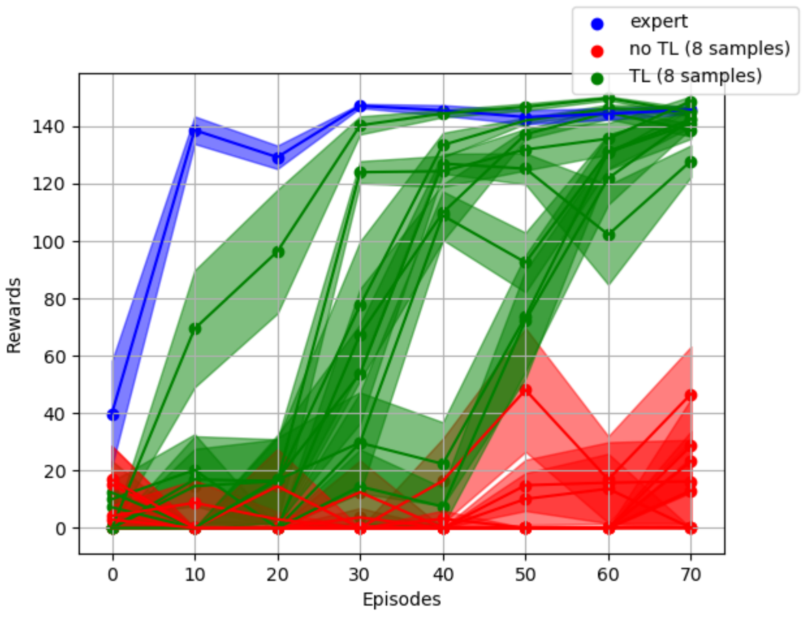


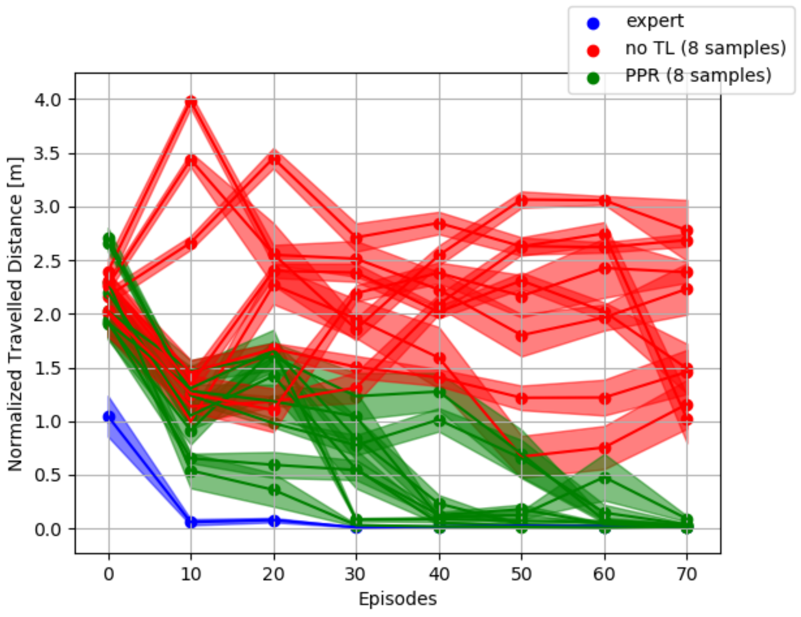
Figure 5.1: Number of wins in the familiarization games

2 Games

In Fig. 5.2a we present the learning curves of the participants and the expert over the 80 test games, while in Fig. 5.2b we present the normalized travelled distance, which is



(a)



(b)

Figure 5.2: Rewards (a) and normalized travelled distance (b) over the testing episodes. At the end of each batch (10, 20, ... episodes) we compute the average over the last 10 games. The transparent regions are the standard error of the mean over the last 10 games.

the travelled distance multiplied by the percentage of the total time spent in a game. Both figures show the same qualitative result. In the games where TL was applied, the performance of the human-robot teams was higher than the performance of the teams in the case where no TL was used. Furthermore, the teams which played without TL needed on average **73** minutes (stdev=2) to complete the experiment while for the teams in the TL group needed on average **34** minutes (stdev=9).

In Figs. 5.3-5.5 and 5.6-5.8 we present the paths during three episodes of a team without TL and a team with PPR from the first, fourth and eighth test batch respectively. These figures provide a spatial representation of the performance of the teams and indicate that using PPR enables the team to win while travelling less distance compared to not using TL at all.

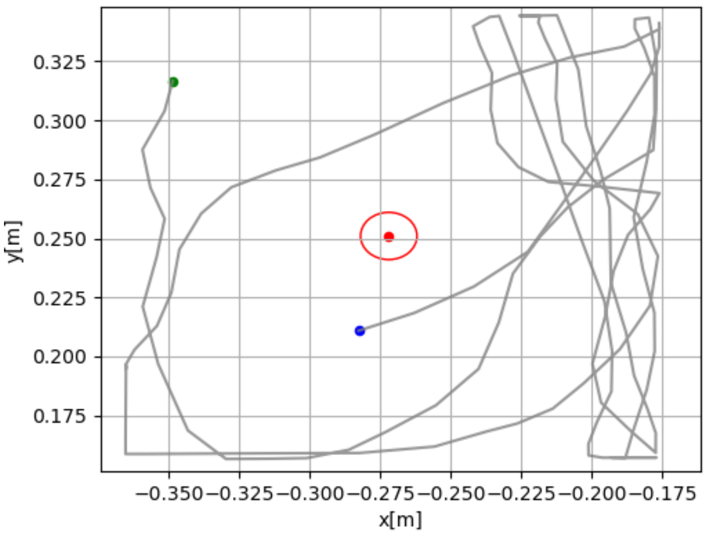


Figure 5.3: Sample runs of a team **without** TL from the **first** batch batch. The green dot is the initial position, the blue the final position, the red the goal position, the red circle is the goal area and the grey line is the path.

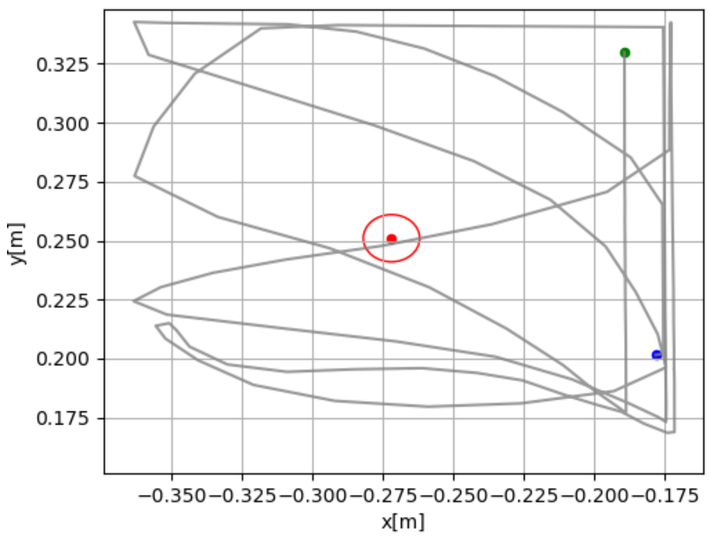


Figure 5.4: Sample runs of a team **without** TL from the **fourth** batch batch. The green dot is the initial position, the blue the final position, the red the goal position, the red circle is the goal area and the grey line is the path.

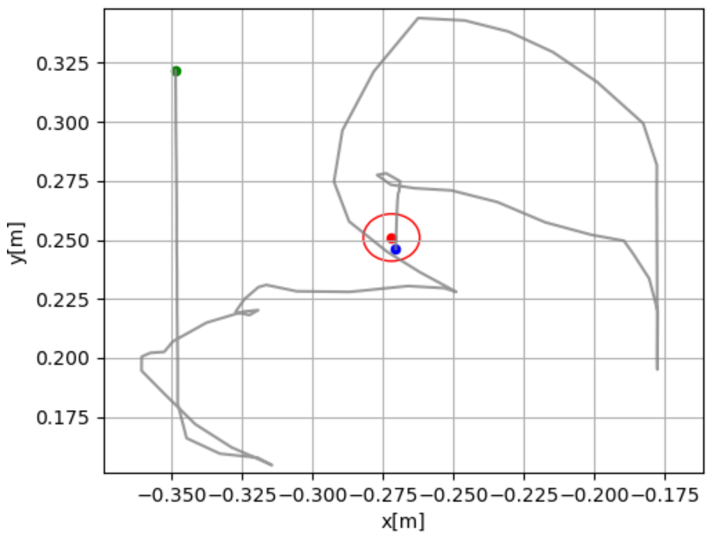


Figure 5.5: Sample runs of a team **without** TL from the **eighth** batch batch. The green dot is the initial position, the blue the final position, the red the goal position, the red circle is the goal area and the grey line is the path.

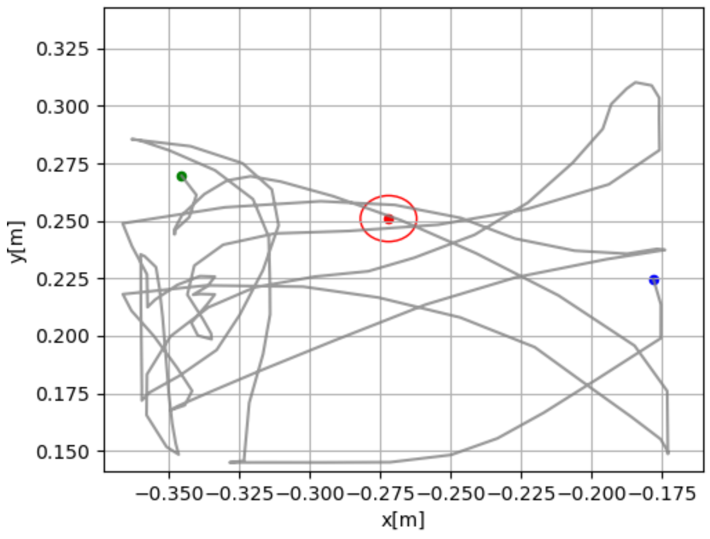


Figure 5.6: Sample runs of a team **with** TL from the **first** batch. The green dot is the initial position, the blue the final position, the red the goal position, the red circle is the goal area and the grey line is the path.

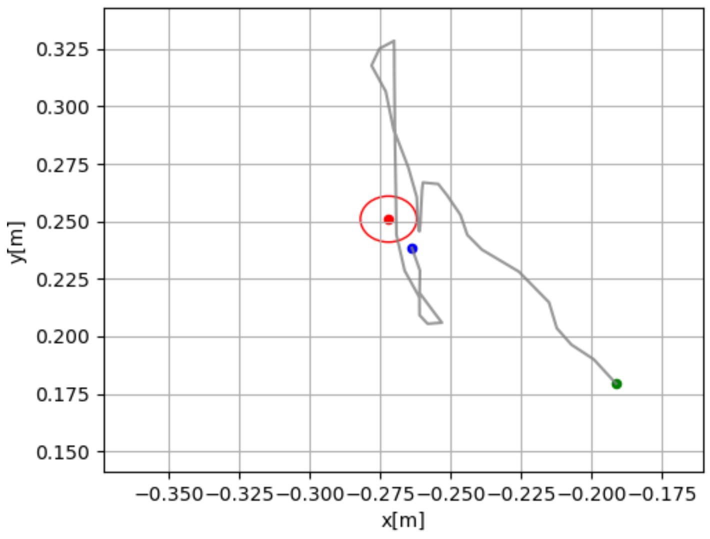


Figure 5.7: Sample runs of a team **with** TL from the **fourth** batch. The green dot is the initial position, the blue the final position, the red the goal position, the red circle is the goal area and the grey line is the path.

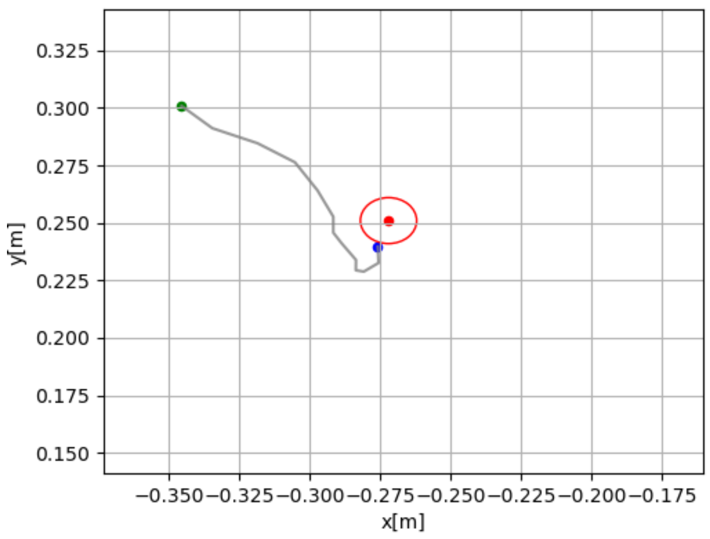


Figure 5.8: Sample runs of a team **with** TL from the **eighth** batch. The green dot is the initial position, the blue the final position, the red the goal position, the red circle is the goal area and the grey line is the path.

The same information extracted from all the test games of one participant is presented in Figs. 5.9-5.11 and 5.12-5.14 for no TL and PPR respectively. For both conditions, the situation in the first batch is the same; the motion of the dot was random and no particular pattern can be observed. In the fourth and eighth test batches, however, we observe a difference in the position of the dot between the two conditions. Specifically, in the fourth batch, the motion of the dot in the case of no TL is still random while in the case of PPR a concentration around the goal position is observed. The same result for the PPR condition can be observed in the eighth batch as well. For the case of no TL, however, we observe an incline towards the left side of the space. This means that the RL agent had learned to lead the dot towards left.

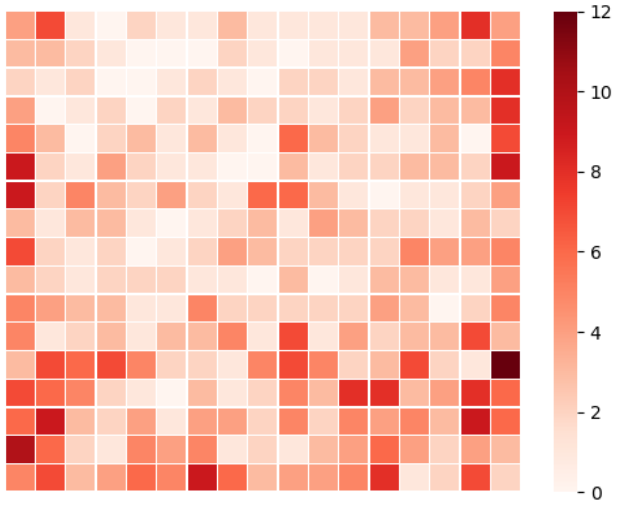


Figure 5.9: Heatmaps of the dot position throughout all the test episodes of a team **without** TL from the **first** batch. The numbers indicate the frequency that the dot was in the respective rectangular region.

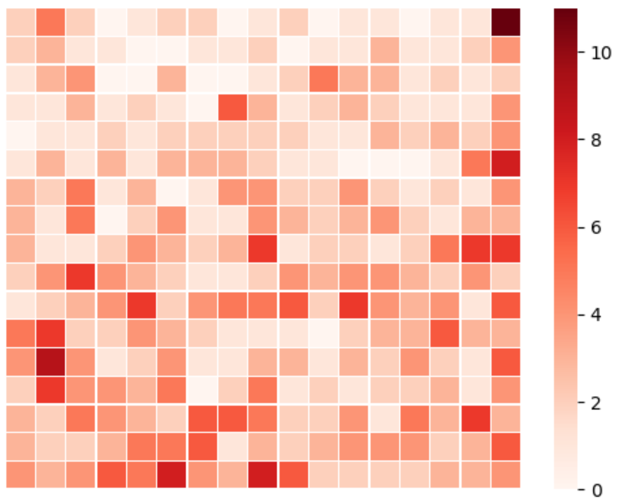


Figure 5.10: Heatmaps of the dot position throughout all the test episodes of a team **without** TL from the **fourth** batch. The numbers indicate the frequency that the dot was in the respective rectangular region.

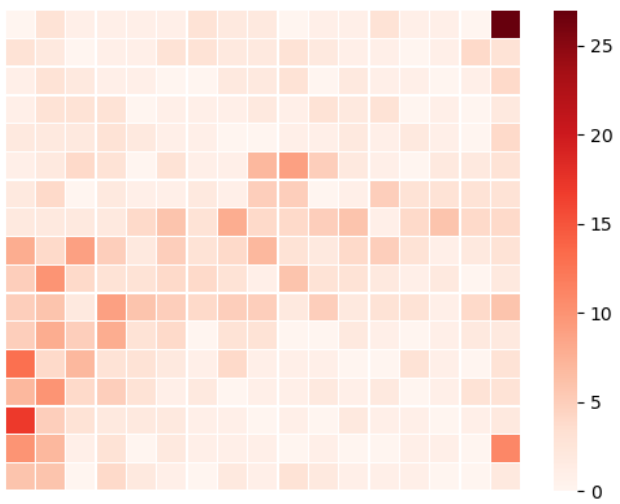


Figure 5.11: Heatmaps of the dot position throughout all the test episodes of a team **without** TL from the **eighth** batch. The numbers indicate the frequency that the dot was in the respective rectangular region.

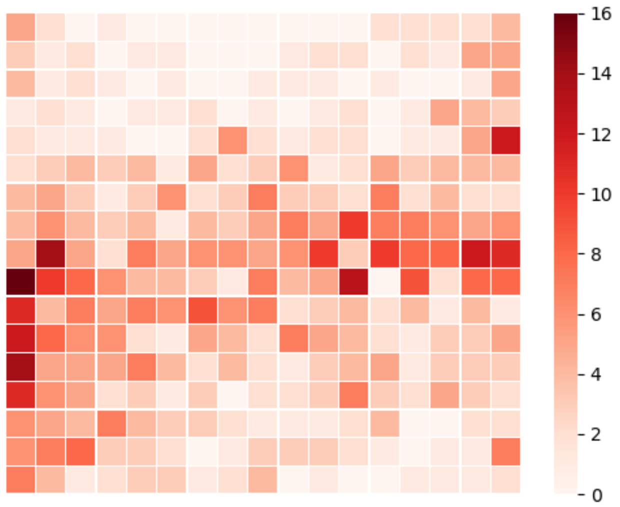


Figure 5.12: Heatmaps of the dot position throughout all the test episodes of a team **with** TL from the **first** batch . The numbers indicate the frequency that the dot was in the respective rectangular region.

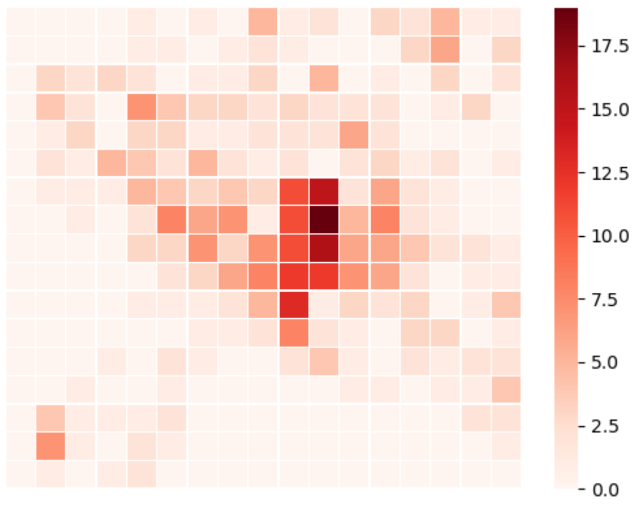


Figure 5.13: Heatmaps of the dot position throughout all the test episodes of a team **with** TL from the **fourth** batch . The numbers indicate the frequency that the dot was in the respective rectangular region.

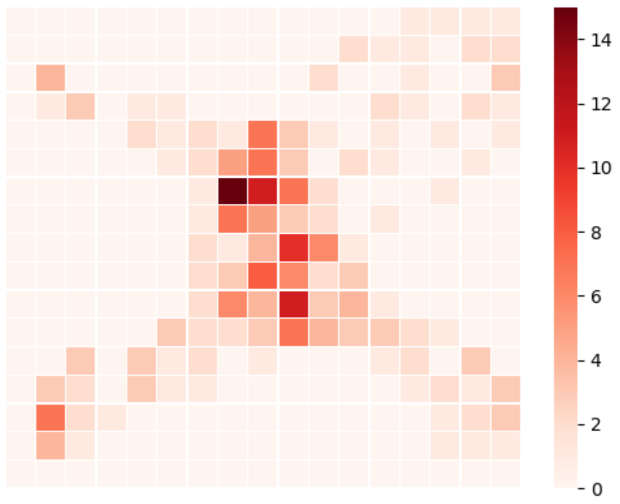


Figure 5.14: Heatmaps of the dot position throughout all the test episodes of a team **with** TL from the **eighth** batch . The numbers indicate the frequency that the dot was in the respective rectangular region.

This qualitative representation of the results, however, does not indicate whether the difference in the learning procedure is significant. In order to answer this question, we applied a two-way mixed ANOVA, which is a statistical test used to compare the means of groups cross-classified by two independent categorical variables, including one between-subjects (condition: No TL or PPR) and one within-subjects (batch: 1, 2, ..., 8). The dependent variable that we applied the ANOVA test on was the average travelled distance per participant and per batch, multiplied by the time percetage of the each game $distance \times (game\_timesteps/max\_timesteps)$. The maximum number of timesteps was 150.

batch	**condition**	**statistic**	**p**
1	No TL	0.904	0.314
1	PPR	0.911	0.362
2	No TL	0.827	0.056
2	PPR	0.904	0.312
3	No TL	0.924	0.466
3	PPR	0.881	0.190
4	No TL	0.958	0.793
4	PPR	0.945	0.664
5	No TL	0.954	0.755
5	PPR	0.693	0.002
6	No TL	0.932	0.530
6	PPR	0.732	0.005
7	No TL	0.941	0.617
7	PPR	0.669	0.001
8	No TL	0.866	0.139
8	PPR	0.843	0.081

Table 5.1: Normality of the dataset

In Table. 5.1 we observe that three datasets do not present normal distribution. For this reason, we analyzed the data using the package WRS2 of R [91]. The results of the ANOVA test are presented in Table. 5.2.

Effect	**F**	**p**	**p<.05**
condition	42.516	0.0003	☑
batch	6.0243	0.0175	☑
condition:batch	5.3243	0.0241	☑

Table 5.2: Mixed ANOVA results

Lastly, the pairwise condition/batch/condition:batch comparison confirmed a significant main effect in all three situations ($p = 0$(*sppba* function)$, p = 0.02$(*sppbb* function), $p = 0.004$(*sppbi* function)).

Excluded Participant One more participant, who played without TL, took part in the experiments but was excluded from the analysis. The reason is that in the first 3 training batches she performed much better than the rest of the human-robot teams. This increased performance might be the result of the inherent abilities of the human participant or the "luck" of the random agent. Since, however, we can not differentiate between the two factors, we decided to exclude her from the analysis.

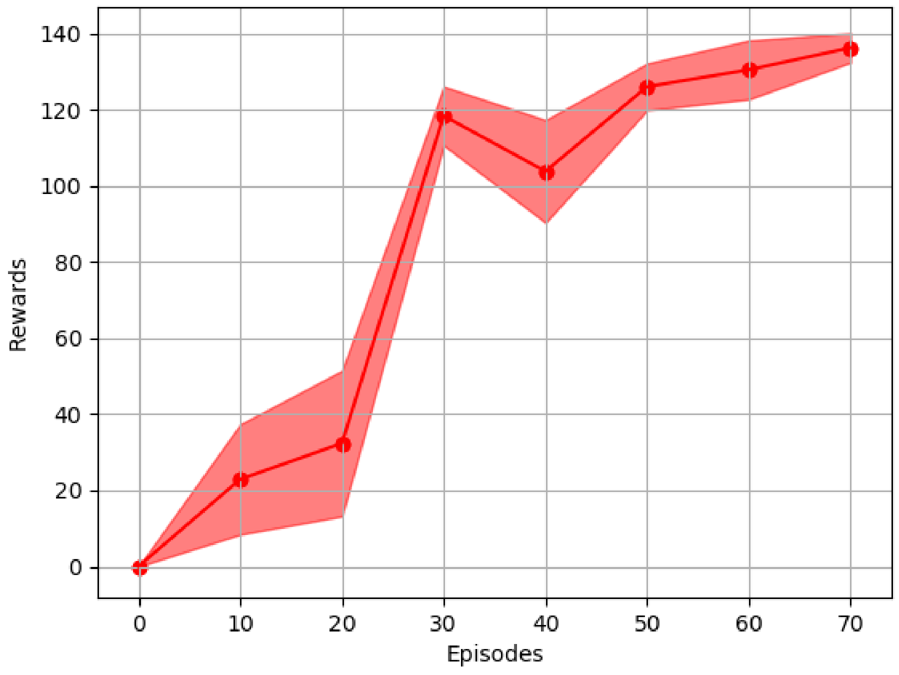


Figure 5.15: Learning curves of participants not included in the analysis

Her learning curve is presented in Fig. 5.15. It is clear that the team managed to collaborate efficiently and win the game consistently even though they played without TL. This observation firstly proves that it is possible to win the game without using TL even without training the human beforehand, like the expert did. Another important observation about the team is presented in Table. 5.3, where the number of wins in the second, third and fourth training batch for teams who played without Tl is presented. The bold data correspond to the team who reached high performance and are much higher than the numbers of wins of the rest teams. This observation might indicate that the performance of the teams in the early training games, where the DRL agent has not converged to a policy yet but still acts a bit randomly, might be crucial for the final performance.

Nr. of participant	**Wins till batch**		
	2nd	3rd	4th
1	2	2	2
2	0	0	0
3	0	0	1
4	0	0	0
5	2	5	5
6	0	2	2
7	2	2	3
8	0	1	1
9	**5**	**7**	**16**

Table 5.3: Number of wins for the teams who played without TL in the second, third and fourth training batches. The bold data correspond to the participant who reached high performance but was excluded by the analysis.

Chapter 6

Discussion

In this thesis, we focused on how the collaboration between a Deep Reinforcement Learning agent and different humans can be enhanced in scenarios where they both need to learn in real-time how solve a shared task while adapting to their partner's behavior. The idea was to exploit the knowledge of an expert agent trained by an expert human, during the collaboration between a naive agent and a different non-expert human. In order to assess this method, we developed a human-agent game in which a human and a Soft Actor-Critic agent needed to collaborate in order to jointly control the motion of the end-effector of a robotic manipulator in the xy-plane. Specifically, the human was controlling the motion of the end-effector in one axis (y-axis), while the agent was controlling the motion of the end-effector in the perpendicular axis (x-axis). The goal of the team was to bring the end-effector to a goal position with a relative low speed. The requirement of the low speed encouraged the team to have more control over the motion of the end-effector.

In general, the results showed that the use of PPR can increase the performance of the human-robot team in a statistically significant way. First of all, the average time for completing the experiment for the No TL group was **73** minutes (stdev=2) while for the TL group was **34** minutes (stdev=9). The teams which played with PPR started to show an improved performance after the fourth batch and their performance increased in every following batch. On the other hand, in the case where no TL was applied, it was observed that the teams were not able to develop a policy which allowed them to win the game systematically and that their performance was not increased significantly. The main reason of this result is the differences in the performance of the teams in the training games. In Fig. 6.1 the percentage of the wins in the training and testing games is shown. The graph shows that even from the second batch the teams which played with PPR managed to win in over 80% of the games while the teams without TL did not achieve a percentage of over 20% in any batch. Lastly, the fact that the teams which played with PPR started to collaborate efficiently after the fourth testing batch while the teams with no TL did not manage to perform well even after eight batches might indicate that the use of PPR is able to decrease the learning time more than in half in simple scenarios such as the one examined in these experiments.

An important parameter that might had an influence on the outcome of the analysis might have been the experience of the participants with playing video games using a keyboard. A common characteristic between the developed human-robot collaborative task and video games is that in both situations the human needs to quickly respond

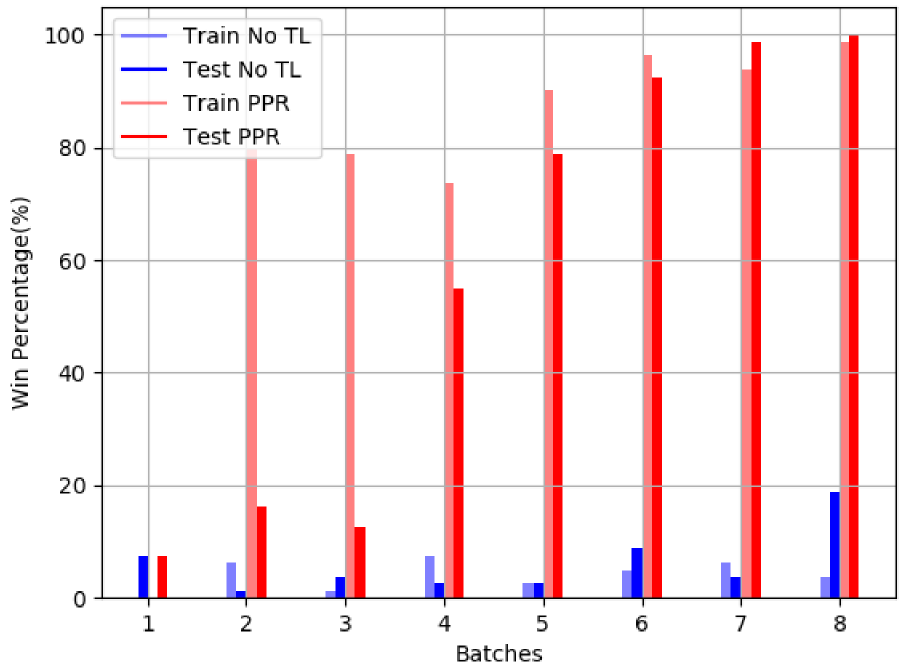


Figure 6.1: Percentage of wins in the training and testing games for each condition and each batch

to visual stimulus by taking appropriate actions using a keyboard. Experience that enhances the human reactions in pressing buttons on a keyboard might help the team to complete the task more frequently. One way to answer this question is by evaluating the experience of each participant in gaming. For example, this is possible by asking the participants to complete a questionnaire, in which relative questions are stated.

A second limitation of our work is that we did not experiment with the parameters of the SAC algorithm. As mentioned earlier, the values of the parameters were based on [33]. Different values might have resulted in different behaviors of the human-robot teams and therefore different results of the analysis. For example, the parameter α of the entropy, which indicates if the agent will choose a random action or an action based on his policy (Exploration vs Exploitation) was 0.0003. This small value means that most of the time the agent will choose to sample an action from its policy. On one hand, acting mostly based on its current policy eliminates the factor of "chance", which makes the results of the analysis more stable. This means that the teams' performance was not boosted or hindered by the probability of the agent taking random actions. On the other hand, not taking random actions might not permit the teams in which the agent had started to converge to a poor policy, to recover and manage to win the game. We believe that experimenting with other values of the SAC algorithm is an important future step.

A different direction for future work would be to test other techniques like Reward Shaping, which in certain scenarios has been proven to be more effective than PPR [46], and observe whether the performance of the team improves even more. Another future direction would be to consider a real-life collaboration scenario, which will obviously be more complex, and where the strengths and weaknesses of the human might have a more significant role on the performance of the team. An example would be situations where a human and a robot need to jointly carry heavy objects. The collaboration is this scenario is twofold. First both parties would need to provide the necessary force so that the object does not fall, while at the same time accounting for the contribution of their partner. The second point is to efficiently move the object. This would require mutual adaptation and potentially prediction of the future motion of each partner. It is obvious that this task is much more complex than the tasks in our work. Furthermore, both the physical and cognitive capabilities of the human would probably have a much bigger influence on the performance of the team compared to our human-robot task.

Bibliography

[1] Judith Bütepage and Danica Kragic. Human-robot collaboration: From psychology to social robotics. *ArXiv*, abs/1705.10146, 2017.

[2] Enrico Anderlini, Salman Husain, Gordon G Parker, Mohammad Abusara, and Giles Thomas. Towards real-time reinforcement learning control of a wave energy converter. *Journal of Marine Science and Engineering*, 8(11):845, 2020.

[3] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:297–330, 2020.

[4] Sarah Elliott, Zhe Xu, and Maya Cakmak. Learning generalizable surface cleaning actions from demonstration. In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 993–999. IEEE, 2017.

[5] S Reza Ahmadzadeh, Roshni Kaushik, and Sonia Chernova. Trajectory learning from demonstration with canal surfaces: A parameter-free approach. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 544–549. IEEE, 2016.

[6] Carlos A Garcia, Jose E Naranjo, Luis A Campana, Maritza Castro, Carmen Beltran, and Marcelo V Garcia. Flexible robotic teleoperation architecture under iec 61499 standard for oil & gas process. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1269–1272. IEEE, 2018.

[7] Lucia Schiatti, Jacopo Tessadori, Giacinto Barresi, Leonardo S Mattos, and Arash Ajoudani. Soft brain-machine interfaces for assistive robotics: A novel control approach. In *2017 International Conference on Rehabilitation Robotics (ICORR)*, pages 863–869. IEEE, 2017.

[8] Jacques Kaiser, Svenja Melbaum, J Camilo Vasquez Tieck, Arne Roennau, Martin V Butz, and Rudiger Dillmann. Learning to reproduce visually similar movements by minimizing event-based prediction error. In *2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob)*, pages 260–267. IEEE, 2018.

[9] Maria Dagioglou, Athanasios C Tsitos, Aristeidis Smarnakis, and Vangelis Karkaletsis. Smoothing of human movements recorded by a single rgb-d camera for robot

demonstrations. In *The 14th PErvasive Technologies Related to Assistive Environments Conference*, pages 496–501, 2021.

[10] Valeria Villani, Fabio Pini, Francesco Leali, and Cristian Secchi. Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, 55:248–266, 2018.

[11] Eloise Matheson, Riccardo Minto, Emanuele GG Zampieri, Maurizio Faccio, and Giulio Rosati. Human–robot collaboration in manufacturing applications: a review. *Robotics*, 8(4):100, 2019.

[12] Mahdi Khoramshahi, Antoine Laurens, Thomas Triquet, and Aude Billard. From human physical interaction to online motion adaptation using parameterized dynamical systems. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1361–1366. IEEE, 2018.

[13] Mahdi Khoramshahi and Aude Billard. A dynamical system approach to task-adaptation in physical human–robot interaction. *Autonomous Robots*, 43(4):927–946, 2019.

[14] Leonardo Sabatino Scimmi, Matteo Melchiorre, Mario Troise, Stefano Mauro, and Stefano Pastorelli. A practical and effective layout for a safe human-robot collaborative assembly task. *Applied Sciences*, 11(4):1763, 2021.

[15] Przemyslaw A Lasota, Terrence Fong, Julie A Shah, et al. *A survey of methods for safe human-robot interaction*, volume 104. Now Publishers Delft, The Netherlands, 2017.

[16] Jair Cornejo, Eddy Denegri, Karina Vasquez, and Oscar E Ramos. Real-time joystick teleoperation of the sawyer robot using a numerical approach. In *2018 IEEE ANDESCON*, pages 1–3. IEEE, 2018.

[17] Mario Selvaggio, Fei Chen, Boyang Gao, Gennaro Notomista, Francesco Trapani, and Darwin Caldwell. Vision based virtual fixture generation for teleoperated robotic manipulation. In *2016 International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 190–195. IEEE, 2016.

[18] Bin Fang, Di Guo, Fuchun Sun, Huaping Liu, and Yupei Wu. A robotic hand-arm teleoperation system using human arm/hand with a novel data glove. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2483–2488. IEEE, 2015.

[19] Shuang Li, Jiaxi Jiang, Philipp Ruppel, Hongzhuo Liang, Xiaojian Ma, Norman Hendrich, Fuchun Sun, and Jianwei Zhang. A mobile robot hand-arm teleoperation system by vision and imu. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10900–10906. IEEE, 2020.

[20] Ioannis Sarakoglou, Anais Brygo, Dario Mazzanti, Nadia Garcia Hernandez, Darwin G Caldwell, and Nikos G Tsagarakis. Hexotrac: A highly under-actuated hand exoskeleton for finger tracking and force feedback. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1033–1040. IEEE, 2016.

[21] D Dajles, F Siles, et al. Teleoperation of a humanoid robot using an optical motion capture system. In *2018 IEEE International Work Conference on Bioinspired Intelligence (IWOBI)*, pages 1–8. IEEE, 2018.

[22] Osama Mazhar, Sofiane Ramdani, Benjamin Navarro, Robin Passama, and Andrea Cherubini. Towards real-time physical human-robot interaction using skeleton information and hand gestures. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–6. IEEE, 2018.

[23] Marek Čorňák, Michal Tölgyessy, and Peter Hubinskỳ. Innovative collaborative method for interaction between a human operator and robotic manipulator using pointing gestures. *Applied Sciences*, 12(1):258, 2021.

[24] Tadej Petrič, Jan Babič, et al. Cooperative human-robot control based on fitts' law. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 345–350. IEEE, 2016.

[25] Ozgur S Oguz, Volker Gabler, Gerold Huber, Zhehua Zhou, and Dirk Wollherr. Hybrid human motion prediction for action selection within human-robot collaboration. In *International Symposium on Experimental Robotics*, pages 289–298. Springer, 2016.

[26] José Ramón Medina, Dongheui Lee, and Sandra Hirche. Risk-sensitive optimal feedback control for haptic assistance. In *2012 IEEE international conference on robotics and automation*, pages 1025–1031. IEEE, 2012.

[27] Emma M Van Zoelen, Karel Van Den Bosch, and Mark Neerincx. Becoming team members: Identifying interaction patterns of mutual adaptation for human-robot co-learning. *Frontiers in Robotics and AI*, 8, 2021.

[28] Karel van den Bosch, Tjeerd Schoonderwoerd, Romy Blankendaal, and Mark Neerincx. Six challenges for human-ai co-learning. In *International Conference on Human-Computer Interaction*, pages 572–589. Springer, 2019.

[29] Paul Baxter, Emily Ashurst, Robin Read, James Kennedy, and Tony Belpaeme. Robot education peers in a situated primary school study: Personalisation promotes child learning. *PloS one*, 12(5):e0178126, 2017.

[30] Thorsten Schodde, Kirsten Bergmann, and Stefan Kopp. Adaptive robot language tutoring based on bayesian knowledge tracing and predictive decision-making. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 128–136, 2017.

[31] Stefanos Nikolaidis, David Hsu, and Siddhartha Srinivasa. Human-robot mutual adaptation in collaborative tasks: Models and experiments. *The International Journal of Robotics Research*, 36(5-7):618–634, 2017.

[32] Stefanos Nikolaidis, Yu Xiang Zhu, David Hsu, and Siddhartha Srinivasa. Human-robot mutual adaptation in shared autonomy. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, pages 294–302. IEEE, 2017.

[33] Ali Shafti, Jonas Tjomsland, William Dudley, and A Aldo Faisal. Real-world human-robot collaborative reinforcement learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11161–11166. IEEE, 2020.

[34] https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/.

[35] Athanasios C Tsitos, Maria Dagioglou, and Theodoros Giannakopoulos. Real-time feasibility of a human intention method evaluated through a competitive human-robot reaching game. In *Proceedings of the 2022 ACM/IEEE International Conference on Human-Robot Interaction*, pages 1080–1084, 2022.

[36] Alexandre Péré, Sébastien Forestier, Olivier Sigaud, and Pierre-Yves Oudeyer. Unsupervised learning of goal spaces for intrinsically motivated goal exploration. *arXiv preprint arXiv:1803.00781*, 2018.

[37] Vaibhav V Unhelkar, Shen Li, and Julie A Shah. Semi-supervised learning of decision-making models for human-robot collaboration. In *Conference on Robot Learning*, pages 192–203. PMLR, 2020.

[38] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[39] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[40] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[41] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3): 279–292, 1992.

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

[43] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[44] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[45] Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *arXiv preprint arXiv:2009.07888*, 2020.

[46] Tim Brys, Anna Harutyunyan, Matthew E Taylor, and Ann Nowé. Policy transfer using reward shaping. In *AAMAS*, pages 181–188, 2015.

[47] Anna Harutyunyan, Sam Devlin, Peter Vrancx, and Ann Nowé. Expressing arbitrary reward functions as potential-based advice. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[48] Xiaoqin Zhang and Huimin Ma. Pretraining deep actor-critic reinforcement learning algorithms with expert demonstrations. *arXiv preprint arXiv:1801.10459*, 2018.

[49] Stefan Schaal. Learning from demonstration. *Advances in neural information processing systems*, 9, 1996.

[50] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[51] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

[52] Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 720–727, 2006.

[53] Matthew E Taylor, Peter Stone, and Yaxin Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(9), 2007.

[54] Hai Nguyen and Hung La. Review of deep reinforcement learning for robot manipulation. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pages 590–595. IEEE, 2019.

[55] Hartmut Surmann, Christian Jestel, Robin Marchel, Franziska Musberg, Houssem Elhadj, and Mahbube Ardani. Deep reinforcement learning for real autonomous mobile robot navigation in indoor environments. *arXiv preprint arXiv:2005.13857*, 2020.

[56] Gregory Kahn, Adam Villaflor, Bosen Ding, Pieter Abbeel, and Sergey Levine. Self-supervised deep reinforcement learning with generalized computation graphs

for robot navigation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5129–5136. IEEE, 2018.

[57] Lucia Liu, Daniel Dugas, Gianluca Cesari, Roland Siegwart, and Renaud Dubé. Robot navigation in crowded environments using deep reinforcement learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5671–5677. IEEE, 2020.

[58] Fangyi Zhang, Jürgen Leitner, Michael Milford, Ben Upcroft, and Peter Corke. Towards vision-based deep reinforcement learning for robotic motion control. *arXiv preprint arXiv:1511.03791*, 2015.

[59] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6023–6029. IEEE, 2019.

[60] Stephen James and Edward Johns. 3d simulation for robot arm control with deep q-learning. *arXiv preprint arXiv:1609.03759*, 2016.

[61] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018.

[62] Marwan Qaid Mohammed, Kwek Lee Chung, and Chua Shing Chyi. Review of deep reinforcement learning-based object grasping: Techniques, open challenges, and recommendations. *IEEE Access*, 8:178450–178481, 2020.

[63] Shirin Joshi, Sulabh Kumra, and Ferat Sahin. Robotic grasping using deep reinforcement learning. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pages 1461–1466. IEEE, 2020.

[64] S Phaniteja, Parijat Dewangan, Pooja Guhan, Abhishek Sarkar, and K Madhava Krishna. A deep reinforcement learning approach for dynamically stable inverse kinematics of humanoid robots. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1818–1823. IEEE, 2017.

[65] Recen Özaln, Cağri Kaymak, Özal Yildirum, Ayşcgül Ucar, Yakup Demir, and Cüneyt Güzeliş. An implementation of vision based deep reinforcement learning for humanoid robot locomotion. In *2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 1–5. IEEE, 2019.

[66] Javier García and Diogo Shafie. Teaching a humanoid robot to walk faster through safe reinforcement learning. *Engineering Applications of Artificial Intelligence*, 88: 103360, 2020.

[67] Victoria J Hodge, Richard Hawkins, and Rob Alexander. Deep reinforcement learning for drone navigation using sensor data. *Neural Computing and Applications*, 33 (6):2015–2033, 2021.

[68] Guillem Muñoz, Cristina Barrado, Ender Çetin, and Esther Salami. Deep reinforcement learning for drone delivery. *Drones*, 3(3):72, 2019.

[69] Chunxue Wu, Bobo Ju, Yan Wu, Xiao Lin, Naixue Xiong, Guangquan Xu, Hongyan Li, and Xuefeng Liang. Uav autonomous target search based on deep reinforcement learning in complex disaster scene. *IEEE Access*, 7:117227–117245, 2019.

[70] Tuomas Haarnoja, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, and Sergey Levine. Learning to walk via deep reinforcement learning. *arXiv preprint arXiv:1812.11103*, 2018.

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

[72] Eugene Valassakis, Zihan Ding, and Edward Johns. Crossing the gap: A deep dive into zero-shot sim-to-real transfer for dynamics. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5372–5379. IEEE, 2020.

[73] Kristinn Kristinsson and Guy Albert Dumont. System identification and control using genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(5):1033–1046, 1992.

[74] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[75] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *Proceedings of the european conference on computer vision (ECCV)*, pages 699–715, 2018.

[76] Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2100–2110, 2019.

[77] Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5, 2021.

[78] Tu-Hoa Pham, Giovanni De Magistris, and Ryuki Tachibana. Optlayer-practical constrained optimization for deep reinforcement learning in the real world. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6236–6243. IEEE, 2018.

[79] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[80] Nicolò Botteghi, Beril Sirmacek, Khaled AA Mustafa, Mannes Poel, and Stefano Stramigioli. On reward shaping for mobile robot navigation: A reinforcement learning and slam based approach. *arXiv preprint arXiv:2002.04109*, 2020.

[81] René Traoré, Hugo Caselles-Dupré, Timothée Lesort, Te Sun, Guanghang Cai, Natalia Díaz-Rodríguez, and David Filliat. Discorl: Continual reinforcement learning via policy distillation. *arXiv preprint arXiv:1907.05855*, 2019.

[82] Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949*, 2017.

[83] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[84] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[85] Rico Jonschkowski and Oliver Brock. State representation learning in robotics: Using prior knowledge about physical interaction. In *Robotics: Science and systems*, 2014.

[86] Lillian Rigoli, Gaurav Patil, Patrick Nalepka, Rachel W Kallen, Simon Hosking, Christopher Best, and Michael J Richardson. A comparison of dynamical perceptual-motor primitives and deep reinforcement learning for human-artificial agent training systems. *Journal of Cognitive Engineering and Decision Making*, page 15553434221092930, 2021.

[87] Lillian M Rigoli, Gaurav Patil, Hamish F Stening, Rachel W Kallen, and Michael J Richardson. Navigational behavior of humans and deep reinforcement learning agents. *Frontiers in psychology*, page 4096, 2021.

[88] Patrick Nalepka, Paula L Silva, Rachel W Kallen, Kevin Shockley, Anthony Chemero, Elliot Saltzman, and Michael J Richardson. Task dynamics define the contextual emergence of human corralling behaviors. *PloS one*, 16(11):e0260046, 2021.

[89] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[90] Petros Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019.

[91] Patrick Mair and Rand Wilcox. Robust statistical methods in r using the wrs2 package. *Behavior research methods*, pages 1–25, 2019.

Chapter 7

Appendix

A ROS implementation

In the Appendix we provide some more technical implementation details about the ROS integration. The integration was achieved using both the ROS C++ API (roscpp) and the Python API (rospy).

In Fig. 7.1 we present the complete ROS graph. Each vertice of the graph (ellipse) represents a node, each edge (arrow) a topic and the direction of the edge means that the outgoing node publishes to the topic while the incoming subscribes to it. The nodes $/ur\_hardware\_interface$, $/ros\_control\_controller\_spawner$, $/ros\_control\_stopped\_spa–wner$, $/robot\_state\_publisher$ and $/controller\_stopper$ provide the functionality of the UR3 ROS driver[1] and are not be presented in detail as they were not developed during the thesis.

The $/teleop\_twist\_keyboard$ node provides the functionality for reading the keyboard input and publishes a $geometry\_msgs :: Twist$ ROS message, which represents the human action, to the $/human\_action\_topic$ topic. The $/rl\_control$ node initiates the game and runs the RL loop. It accepts the human action by subscribing to the $/human\_action\_topic$ topic, samples the DRL agent action based on the action selection procedure described in Section 2 and publishes the action to the $/agent\_action\_topic$ topic as a $std\_msgs :: Float64$ message. Furthermore, it checks if the game ended by comparing the current EE position and velocity to the goal position and velocity. When the game ends, it publishes its outcome and the score of the team to the $/score\_topic$ topic as a custom ROS message and sends a request to the $/reset$ service (not shown in the ROS graph) for resetting the game as described in Section 4 .2. The last functionality of the $/rl\_control$ node is that it publishes a $std\_msgs :: Bool$ message to the $/train\_topic$ topic. The value of the message is "true" during the training of the DRL agent and the idea is to prevent the user from controlling the robot during the training procedure.

The computation of the commanded velocities according to equations 4.3 and 4.5 is implemented in the $/robot\_movement\_generation$ node. The node subscribes to the $/human\_action\_topic$ and to the $/agent\_action\_topic$ topics for accessing the human and agent actions respectively and to the $/ur3\_cartesian\_velocity\_controller/ee\_state$ topic in order to get the current state of the EE and publishes the commanded velocities to the $/ur3\_cartesian\_velocity\_controller/command\_cart\_vel$ topic as a $/geometry\_msgs ::$

[1]`https://github.com/UniversalRobots/Universal_Robots_ROS_Driver`

Twist message. Furthermore, it implements the */reset* service. Lastly, the */score_visual–ization* node is responsible for visualizing the outcome of the game and the score of the team (Fig 4.3). The visualization is based on the “pygame” python library.

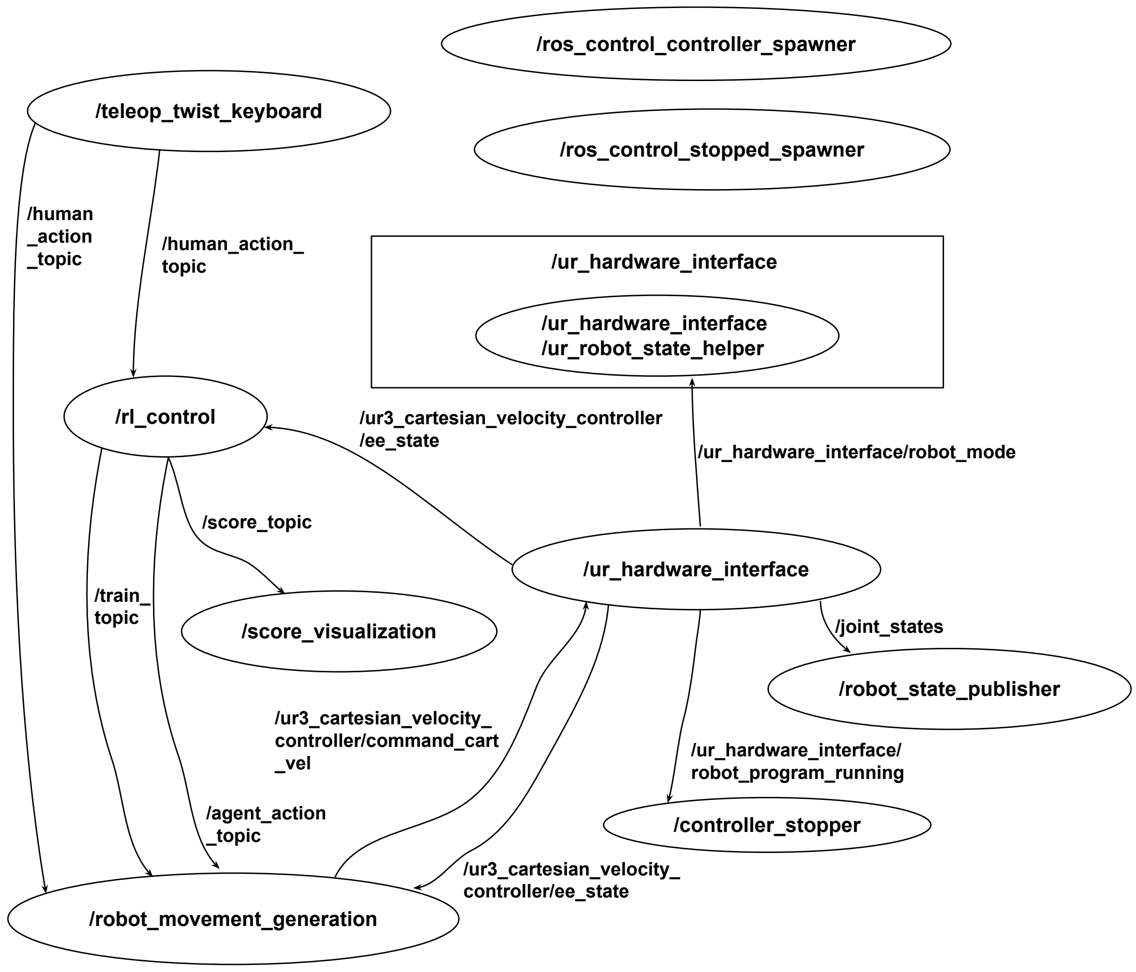

/ros_control_controller_spawner
/teleop_twist_keyboard
/ros_control_stopped_spawner
/human_action_topic
/human_action_topic
/ur_hardware_interface
/ur_hardware_interface/ur_robot_state_helper
/rl_control
/ur3_cartesian_velocity_controller/ee_state
/ur_hardware_interface/robot_mode
/score_topic
/ur_hardware_interface
/train_topic
/score_visualization
/joint_states
/robot_state_publisher
/ur3_cartesian_velocity_controller/command_cart_vel
/ur_hardware_interface/robot_program_running
/controller_stopper
/agent_action_topic
/ur3_cartesian_velocity_controller/ee_state
/robot_movement_generation

Figure 7.1: ROS graph of the system

B Letter

Συνοδευτική Ενημερωτική Επιστολή
Έρευνα για τη συνεργασία ανθρώπου-ρομπότ

Έχετε εκδηλώσει ενδιαφέρον να συμμετάσχετε στη διεξαγωγή έρευνας που γίνεται στα πλαίσια εκπόνησης διπλωματικής εργασίας του ΔΜΠΣ στην Τεχνητή Νοημοσύνη που συνδιοργανώνεται από το Πανεπιστήμιο Πειραιά και το ΕΚΕΦΕ “Δημόκριτος”[2]. Η έρευνα έχει ως στόχο την ανάπτυξη μεθόδων για εύρυθμη και ασφαλή συνεργασία ανθρώπου και ρομποτικού βραχίονα σε κοινό εργασιακό χώρο και συγκεκριμένα σε εργασίες όπου απαιτείται συνδυασμός κινήσεων από τους δύο συνεργάτες.

Πρακτικές πληροφορίες

Η διεξαγωγή της έρευνας γίνεται στο χώρο του εργαστηρίου Roboskel (κτίριο Κεντρικής Βιβλιοθήκης, ΕΚΕΦΕ Δημόκριτος). Η διαδικασία συλλογής δεδομένων ολοκληρώνεται σε **μια επίσκεψη** που θα διαρκέσει περίπου 2 ώρες. Η συμμετοχή σας γίνεται σε εθελοντική βάση και δεν έχει κανένα όφελος για σας, οικονομικό, ή οποιασδήποτε άλλης φύσης.

Η διαδικασία / Μπορώ να διακόψω την διαδικασία·

Στα πλαίσια της διεξαγωγής του πειράματος καλείστε να συνεργαστείτε με ένα ρομποτικό βραχίονα για να ελέγξετε από κοινού τη θέση του και να τον μεταφέρετε σε μία θέση-στόχο. Εσείς ελέγχετε την κίνηση του ρομποτικού βραχίονα κατά έναν άξονα μέσω ενός πληκτρολογίου. Το ρομπότ είναι υπεύθυνο να ελέγχει την κίνησή του σε άλλο άξονα (κάθετο σε αυτόν του ανθρώπου). Συνολικά θα εκτελέσετε 140 δοκιμές. Κάθε δοκιμή ολοκληρώνεται είτε όταν επιτευχθεί ο στόχος είτε αν παρέλθουν 20 δευτερόλεπτα. Στο τέλος κάθε δοκιμής θα ενημερώνεστε για την από κοινού επίδοσή σας με το ρομπότ. Ανά τακτά χρονικά διαστήματα θα γίνονται διαλείμματα ώστε να αποφευχθεί οποιαδήποτε κόπωση.

Η συμμετοχή σας είναι εθελοντική. Μπορείτε να εγκαταλείψετε τη διαδικασία οποιαδήποτε στιγμή.

Τι είδους δεδομένα θα συλλεχθούν και πως θα χρησιμοποιηθούν·

Στην αρχή της διαδικασίας θα σας ζητηθεί να συμπληρώσετε ένα ερωτηματολόγιο με κάποια δημογραφικά στοιχεία. Κατά τη διάρκεια και στο τέλος του πειράματος, θα κληθείτε να σχολιάσετε διάφορες πτυχές της συνεργασίας σας με το ρομπότ.

Επίσης, άλλα δεδομένα που θα συλλεχθούν είναι χαρακτηριστικά της κίνησης του βραχίονα (π.χ. θέση, ταχύτητα), στοιχεία σχετικά με τον αλγόριθμο Τεχνητής Νοημοσύνης που χρησιμοποιείται και οι ενέργειες του ανθρώπου, δηλαδή τα πλήκτρα που πατάει κατά την εξαγωγή των πειραμάτων. Τα δεδομένα θα χρησιμοποιηθούν για ερευνητικούς σκοπούς.

Η συλλογή όλων των δεδομένων θα γίνει ανώνυμα με τη χρήση κωδικού ονόματος συμμετέχοντα (π.χ. Σ-1, 2, κτλ)

Τι είδους πληροφορίες θα είναι διαθέσιμες δημόσια·

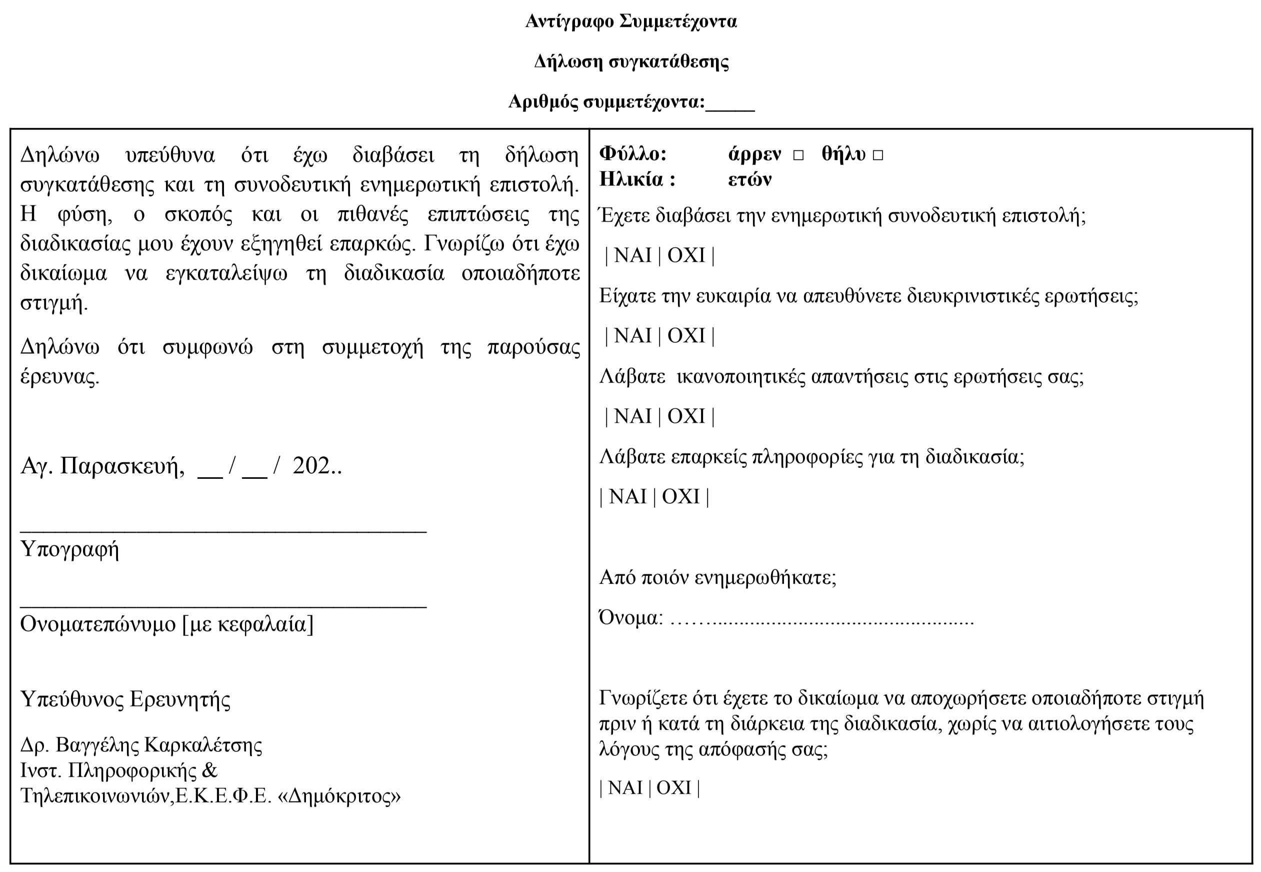
Δημόσια σε περίπτωση δημοσίευσης θα γίνουν διαθέσιμα τα αποτελέσματα στατιστικής ανάλυσης επί των συλλεχθέντων δεδομένων καθώς και μεμονωμένες απαντήσεις σε ανοιχτού τύπου ερωτήσεις χωρίς **να συναφθεί καμία πληροφορία που αφορά κωδικό όνομα ή άλλα δημογραφικά στοιχεία του ατόμου.**

[2]ηττπ://μσς-αι.ιιτ.δεμοκριτος.γρ/

Οι φόρμες συγκατάθεσης θα φυλαχθούν εμπιστευτικά από το ΙΠΤ. Υπάρχουν κίνδυνοι και ενοχλήσεις κατά τη διάρκεια της συλλογής δεδομένων·
Η συμμετοχή σας στο πείραμα δεν έχει κανένα κίνδυνο για εσάς, ούτε θα αισθανθείτε κάποια ενόχληση κατά τη διάρκειά του. Επίσης δεν χρειάζεται καμία προφύλαξη κατά τη διάρκειά του.
Αν έχετε κάποιες περαιτέρω απορίες για τα προαναφερθέντα, μη διστάσετε να μας ρωτήσετε.
Η συμμετοχή σας είναι εθελοντική. Η άρνηση παροχής συγκατάθεσης δεν επιφέρει καμία αρνητική συνέπεια σε εσάς. Διατηρείτε επίσης το δικαίωμα αναίρεσης της συγκατάθεσής σας. Στην περίπτωση τέτοιας αναίρεσης, τα συγκεντρωμένα δεδομένα σας θα διαγραφούν άμεσα.
Αν έχετε άλλες ερωτήσεις, μπορείτε να απευθυνθείτε στους:
Δρ. Μαρία Δαγιόγλου, τηλ.: 2106503201, εμαιλ: μδαγιογλΐιτ.δεμοκριτος.γρ.

Με εκτίμηση,
Δρ. Βαγγέλης Καρκαλέτσης
Ινστιτούτο Πληροφορικής και Τηλεπικοινωνιών, ΕΚΕΦΕ «Δημόκριτος»

C Consent Form



Αντίγραφο Συμμετέχοντα

Δήλωση συγκατάθεσης

Αριθμός συμμετέχοντα:_____

Δηλώνω υπεύθυνα ότι έχω διαβάσει τη δήλωση συγκατάθεσης και τη συνοδευτική ενημερωτική επιστολή. Η φύση, ο σκοπός και οι πιθανές επιπτώσεις της διαδικασίας μου έχουν εξηγηθεί επαρκώς. Γνωρίζω ότι έχω δικαίωμα να εγκαταλείψω τη διαδικασία οποιαδήποτε στιγμή. Δηλώνω ότι συμφωνώ στη συμμετοχή της παρούσας έρευνας. Αγ. Παρασκευή, __ / __ / 202.. ______________________________ Υπογραφή ______________________________ Ονοματεπώνυμο [με κεφαλαία] Υπεύθυνος Ερευνητής Δρ. Βαγγέλης Καρκαλέτσης Ινστ. Πληροφορικής & Τηλεπικοινωνιών,Ε.Κ.Ε.Φ.Ε. «Δημόκριτος»	**Φύλλο:** **άρρεν** □ **θήλυ** □ **Ηλικία :** **ετών** Έχετε διαβάσει την ενημερωτική συνοδευτική επιστολή; | ΝΑΙ | ΟΧΙ | Είχατε την ευκαιρία να απευθύνετε διευκρινιστικές ερωτήσεις; | ΝΑΙ | ΟΧΙ | Λάβατε ικανοποιητικές απαντήσεις στις ερωτήσεις σας; | ΝΑΙ | ΟΧΙ | Λάβατε επαρκείς πληροφορίες για τη διαδικασία; | ΝΑΙ | ΟΧΙ | Από ποιόν ενημερωθήκατε; Όνομα: ……... Γνωρίζετε ότι έχετε το δικαίωμα να αποχωρήσετε οποιαδήποτε στιγμή πριν ή κατά τη διάρκεια της διαδικασία, χωρίς να αιτιολογήσετε τους λόγους της απόφασής σας; | ΝΑΙ | ΟΧΙ |

Figure 7.2: Consent form