Optimizing Decision-Making for Human-Robot Collaboration

By Mélodie DANIEL

Thèse de Doctorat de l'Université Clermont Auvergne en Électronique et Systèmes

Reviewers before the defense

Sénen BARRO AMENEIRO	Professor at Universidad de Santiago de Compostela
Helder Jesus ARAUJO	Professor at University of Coimbra
Philippe FRAISSE	Professor at Université de Montpellier

Jury members

Roland LENAIN	Jury president	Research Director at INRAE (TSCF unit)
Helder Jesus ARAUJO	Reviewer	Professor at University of Coimbra
Philippe FRAISSE	Reviewer	Professor at Université de Montpellier
Claire DUNE	Examiner	Associated professor at Université de Toulon
Roberto IGLESIAS RODRIGUEZ	Examiner	Associated professor at Universidad de Santiago de Compostela
Youcef MEZOUAR	Thesis director	Professor at Clermont Auvergne INP - Sigma Clermont
Sébastien LENGAGNE	Thesis supervisor	Associated professor at Clermont Auvergne INP - Polytech Clermont
Juan Antonio CORRALES RAMÓN	Thesis supervisor	Researcher at Universidad de Santiago de Compostela

Doctoral School: Sciences pour l'ingénieur
Research laboratory: Institut Pascal - ISPR axis - MACCS team
University: Université Clermont Auvergne (UCA)
Clermont-Ferrand, France
Date of thesis defense: 08 December 2022

Abstract

Robots are gradually making their way from industries and laboratories into our daily lives. Whether they serve as companions, teachers, receptionists, cleaners, or meet other needs, these robots aim to improve our quality of life. However, robots' decisional autonomy remains the major challenge in robotics. To increase the robot's autonomy, the researchers tend, on the one hand, to classify the collaborations based on different criteria to gather the commonalities between the human-robot collaborations. The goal is to detect the similar steps that the robot must be able to accomplish to perform the various tasks. On the other hand, other works focus on enhancing one or many of the fields required to set up a human-robot collaboration. The robot must perform four standard steps to set up a human-robot collaboration: perception, decision-making, motion execution, and evaluation.

This thesis aims to optimize the human-robot collaboration performance by improving the robot's decision-making process. We evaluate the collaboration performance based on different changeable performance metrics. Hence, an optimized collaboration aims to benefit humans, such as getting the task done faster or reducing the effort of human agents. However, an unoptimized collaboration will bring nothing to humans or, on the contrary, will represent a nuisance, such as slowing them down or overloading them, even if the task is finally accomplished.

We start by developing a global framework that optimizes the robot's decision-making process. We apply this framework to a non-intuitive assembly task, i.e., complex cognitive processing is required to find the right place for each piece of the proposed assembly game. We want to enhance the task time to completion by a collaborative human-robot team without having to increase its physical capabilities (i.e., perception, trajectory planning, or low-level control). Our proposed framework can improve human-robot collaboration while considering different performance metrics. These metrics are considered regardless of the behavior of the human agent.

We then apply this framework to a second application more complex (i.e., deforming a soft object) that requires increasing the robot's manipulation dexterity by improving its low-level control. Indeed, we will consider this second application that requires increasing the robot's manipulation dexterity to maximize the optimization of the collaboration performance. A human-robot collaborative team will have to co-manipulate the soft object to make it reach desired shapes. The collaborative team can achieve this application using a deep reinforcement learning approach. The idea is to train the agent (a single-arm robot or a dual-arm robot) in simulation and to test it in real by replacing the second robotic arm with the human agent.

Keywords: Human-robot collaboration, Decision-making, Game theory, Reinforcement learning.

Résumé

Les robots se fraient progressivement un chemin depuis les industries et les laboratoires jusqu'à notre vie quotidienne. Qu'ils servent de compagnons, d'enseignants, de réceptionnistes, de nettoyeurs ou répondent à d'autres de nos besoins, ces robots ont pour objectif d'améliorer notre qualité de vie. Cependant, l'autonomie décisionnelle des robots reste un défi majeur dans le domaine de la robotique. Pour augmenter l'autonomie du robot, les chercheurs ont tendance à classer les collaborations en fonction de différents critères afin de rassembler les points communs entre les collaborations homme-robot. L'objectif est de détecter les étapes similaires que le robot doit être capable d'accomplir pour réaliser les différentes tâches. D'autre part, d'autres travaux visent à améliorer un ou plusieurs des domaines nécessaires pour qu'une collaboration homme-robot soit mise en place. Le robot doit effectuer quatre étapes standards pour mettre en place une collaboration homme-robot : la perception, la prise de décisions, l'exécution de mouvements et l'évaluation.

Cette thèse se concentre sur l'optimisation des performances de la collaboration homme-robot en améliorant le processus décisionnel du robot. Nous évaluons la performance de la collaboration à l'aide de différentes métriques modifiables de performance. Ainsi, une collaboration optimisée a pour but d'apporter des bénéfices aux humains, tels que l'accomplissement plus rapide de la tâche ou la réduction de l'effort des agents humains. En revanche, une collaboration non optimisée n'apportera rien aux humains ou, au contraire, représentera une nuisance, comme le fait de les ralentir ou de les surcharger et ce même si la tâche est finalement accomplie.

Nous commençons par développer un framework global qui optimise le processus décisionnel du robot. Nous appliquons ce framework à une tâche d'assemblage non intuitive, c'est-à-dire qu'un raisonnement cognitif complexe est nécessaire pour trouver le bon endroit où placer chaque pièce du jeu d'assemblage proposé. Nous voulons améliorer la réalisation de la tâche par une équipe homme-robot collaborative sans avoir à augmenter les capacités physiques de ce dernier (à savoir la perception, la planification de trajectoires ou le contrôle bas-niveau). Le framework que nous proposons peut améliorer la collaboration homme-robot en prenant en compte différentes métriques de performance. Ces métriques sont prises en compte indépendamment du comportement de l'humain.

Nous appliquons ensuite ce framework à une deuxième application plus complexe (à savoir la déformation d'un objet mou) qui nécessite d'augmenter la dextérité de manipulation du robot en améliorant son contrôle bas-niveau. En effet, nous prenons en considération cette deuxième application, qui nécessite d'augmenter la dextérité de manipulation du robot, afin de maximiser l'optimisation de la performance de la collaboration. Une équipe collaborative homme-robot devra co-manipuler un objet déformable afin de lui faire atteindre des formes désirées.

L'équipe collaborative peut réaliser cette application à l'aide d'une approche d'apprentissage par renforcement profond. L'idée est d'entraîner l'agent (un robot à un ou deux bras) en simulation et de le tester en situation réelle et ce en remplaçant le second bras robotique par l'agent humain.

Mots-clés : Collaboration homme-robot, Prise de décisions, Théorie des jeux, Apprentissage par renforcement.

Acknowledgements

My Ph.D. was a period full of surprises and intense scientific and personal learning. At the same time, there have been many challenging moments where I doubted that I could overcome them. There are many people without whom I would not have been able to achieve this work and whom I would like to thank in this section.

Firstly, I would like to thank my supervisors, Sébastien LENGAGNE, Juan CORRALES, and Youcef MEZOUAR, for giving me this great opportunity. I'm grateful for their advice, support, and encouragement and for their time devoted to me over the past years. In particular, for having brought me their help in all that I wanted to undertake and for having helped me overcome the period of Covid, which strongly penalized my experiments with humans.

Secondly, I want to express my gratitude to the thesis reporters, Helder Jesus ARAUJO, Philippe FRAISSE, and Senén BARRO AMENEIRO, for agreeing to review this dissertation. I sincerely thank Roland LENAIN, Helder Jesus ARAUJO, Philippe FRAISSE, Claire DUNE, and Roberto IGLESIAS RODRIGUEZ for accepting to be members of the thesis defense jury.

I would also like to thank all the people involved in the MACCS research team for their help and support. In particular, I would like to thank Laurent LEQUIEVRE for his precious advice and assistance concerning the programming and setting up the experimental frameworks we had to carry out to valorize our research work. I also want to thank Miguel ARANDA for his valuable suggestions and guidance regarding the writing of our papers and his explanations in the classical robotic control field. I am grateful to Mohammad ALKHATIB for his exceptional management of the existing experimental platforms and for all the improvements he made to make them fit our needs. Likewise, I am thankful to Marc NIVOIX for his help in carrying out all the mechanical work required to set up the experimental platforms. I am grateful to Reza SHETAB and Omid AGHAJANZADEH, whom I exchanged daily and supported throughout my thesis, especially during the period of covid when very few people were present in the laboratory.

I want to thank Benoît THUILOT, my Master's supervisor, who pushed me to surpass myself, encouraged me to do a Ph.D., and believed in me more than I believed in myself. I want to thank Céline TEULIERE for creating a reinforcement learning group that gathers all the people working on this theme in the laboratory, facilitating the exchange of experiences. I am grateful to all the great people I have not mentioned from the Institut Pascal who made these three years an unforgettable experience.

I am grateful to Paula LOPEZ MARTINEZ for inviting me for a three-month research mobility in the Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS) lab

belonging to the Santiago de Compostela university in Spain. I want to thank everyone there for their great welcome and help, notably Roberto IGLESIAS RODRIGUEZ, for his excellent advice to improve my work.

Finally, I am thankful to my amazing parents, family, and friends for always accepting and supporting me in my choices. You have always been there for me, even when we were far away, or I had to focus more on my research than you, and I know that you will always be there.

*****

This work has received funding from the Auvergne-Rhône-Alpes Region through the ATTRIHUM project and from the European Union's Horizon 2020 research and innovation programme under grant agreement No 869855 (Project 'SoftManBot').

My mobility to the laboratory CiTIUS in Santiago de Compostela in Spain has received funding from ERASMUS + Stages and the project I-SITE Clermont of CAP 20-25.

We are grateful to the Mésocentre Clermont Auvergne University / AuBi platform for providing help, computing, and storage resources.

We received free foam samples from "Maison du Caoutchouc et de la Mousse" (`https://www.maison-mousse.com/`).

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AHP Analytic Hierarchy Process

DM Decision-Making

DDPG Deep Deterministic Policy Gradient

DL Deep Learning

DNNs Deep Neural Networks

DRL Deep Reinforcement Learning

DLO Deformable Linear Object

FC Fully Connected

HRC Human-Robot Collaboration

IL Imitation Learning

MDP Markov Decision Process

MADRL Multi-Agent Deep Reinforcement Learning

MAU Multi-Attribute Utility

NE Nash Equilibrium

PHER Percentage of Human Errors Reduction

PIEF Perfect-Information Extensive Form

ReLU Rectified Linear Unit

RSM Response Surface Methodology

SO Soft Object

TL Transfer Learning

Part I

State-of-the-art of human-robot collaboration and the robot's decision-making process

Chapter 1

General introduction

Contents

Nowadays, robots are introduced more and more into our daily lives. In the next decade, we can expect that they will no longer be just mechanical tools used primarily in industries but will become true partners and companions in their own right. That is why there is an increasing

interest in studying how robots should behave in environments shared with humans and how they can collaborate to solve tasks.

1.1 Human-robot collaboration

The Human-Robot Collaboration (HRC) is the study of collaborative processes in which human and robot agents work together to achieve shared goals [13]. The HRC is an interdisciplinary research field that combines classical robotics, human-robot interaction, artificial intelligence, computer science, sociology, ergonomics, cognitive sciences, and psychology [14]. The HRC is useful in daily life applications such as therapy [3, 4], companionship [2], and education [1]. Figure 1.1 presents some daily life applications of HRC. The HRC is used in industrial tasks, such as assembly [5], sawing [8], welding [7], and surface polishing [15]. Figure 1.2 presents some industrial applications of HRC.

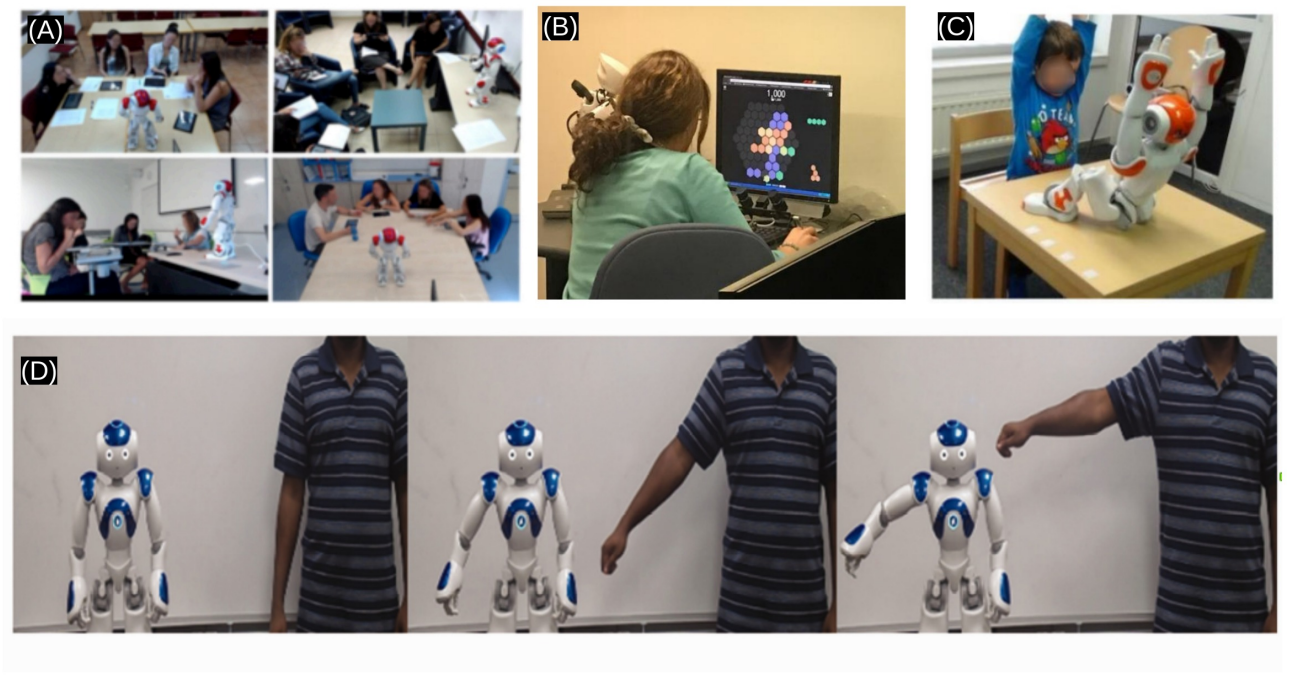


Figure 1.1: Examples of daily life applications of HRC: (A) Education [1], (B) Companionship [2], (C) Autism disorders therapy [3], and (D) Rehabilitation therapy [4].

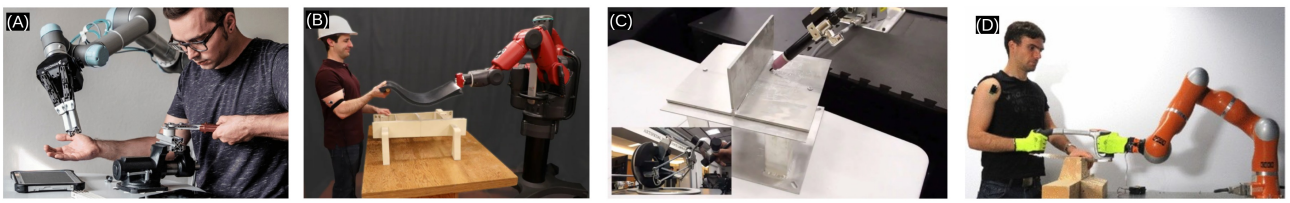


Figure 1.2: Examples of industrial applications of HRC: (A) Assembly task [5], (B) Co-manipulation task [6], (C) Welding task [7], and (D) Sawing task [8].

The HRC is challenging to set up because the more the task the robot performs in col-

laboration with the human is complex, the more the robot should be able to make decisions autonomously. For example, in the case of a mobile robot working in a museum to make tourists visit, the robot must distinguish the group of tourists it guides from other people, avoid hitting people or objects, respect social distancing, speak only when it is in front of a piece of art, ... [16]. The application can become more complicated when robots and humans cooperate to perform a common task, such as sharing the load of heavy objects. In addition to what the robots have to handle previously (obstacle avoidance, human detection, etc.), they have to predict the intentions of the humans to coordinate their movements with them [17].

Despite the recent development of several well-performing robots equipped with rich proprioception sensing and actuation control, these robots are still far from fully autonomous to collaborate smoothly with humans. To increase the autonomy of the robots, the researchers tend to classify the collaborations based on different criteria to gather the commonalities between the HRCs. The goal is to detect the similar disciplines that the robot must be able to accomplish to perform the various tasks. Several fields are required to set up a HRC. The robot has to perform four main standard steps: perception, Decision-Making (DM), motion execution, and evaluation.

1.2 Classification of human-robot collaborations

Classifying HRCs according to their similarities helps to increase robots' decisional autonomy. In the case where two robots should have similar capabilities and perform similar actions, they can be replaced by a single more autonomous robot. There are several ways to classify the type of collaboration in the literature. A HRC can be classified according to the level of autonomy of the robots [18], the type of robots that collaborate [19], the category of the task to perform [19, 20], and the benefit that the collaboration brings to the human agents.

1.2.1 Robot's level of autonomy

The level of autonomy describes how much the human is involved in the collaboration [21]. From "human decides about it all" to "computer determines everything and acts independently" might represent the progression of autonomy levels. The classification according to the level of autonomy of the robot proposed in [18] and [19] classifies a HRC into one of the following categories:

- Teleoperation: when the human controls the robot completely, as in the case of remote-controlled drones, whether in an aerial, terrestrial or underwater environment.
- Supervised control: the human commands and monitors the robot's actions but allows it some autonomy to make straightforward actions within the tasks, such as the robot searching for or grasping objects, etc.

- Autonomous systems: the human agent determines the task goal, and the machine has to achieve it.
- Social interaction: social robots are mostly autonomous since they assist or guide people.

1.2.2 Robot's type

We then present the HRC classification based on the type of robots. Robots are traditionally categorized according to their morphology, which typically enables a visual and functional depiction of their use [19]:

- Serial robot: A robot made of a serial kinematic chain that ends up with a gripper.
- Parallel robot: A robot with a mobile platform attached to a fixed base by several parallel identical kinematic chains.
- Mobile robot: A robot with different modalities (i.e., wheels, legs, etc.) that moves in the environment.

1.2.3 Category of the application to be performed

Initially, the classification of the HRC based on the kind of application to be performed was split into two primary categories: industrial and non-industrial applications related to service robotics. Other categories have been added as the use of robots in daily life has advanced, such as the one described by [19], which contains the following application categories: industrial, domestic, medical, military, etc. Cutting-edge publications have suggested a taxonomy that is even more specific. In [20], they introduce new categories, including urban search and rescue, walking aid for the blind, toy, or delivery robot. Other categories can also be added, such as education, physical therapy, cognitive therapy for autism spectrum disorders, etc.

1.2.4 Impact of collaboration on human agents

The problem with using the application to categorize collaboration is that it is sometimes unable to identify the difference between interactions. Let's consider, for example, the case of a collaboration in which a robot helps the elderly. This assistance can be either physical or psychological. In either case, the approach to operating the robot will be very different. The last way to classify the HRCs that we will present is according to the benefit that the collaboration brings to the human agents, i.e., social or physical. It is the most used one in the literature [22]. This way of classification allows us to make a distinction between the interactions that other ways of classification cannot. This is why we have judged that classifying the interactions according to their impact on the human remains legitimate while effectively separating them.

HRC applications can have social and/or physical benefits for humans [11]. Social collaboration tasks include social, emotional and cognitive aspects [23] such as care for the elderly [24, 25], therapy [26], companionship [2], and education [1]. Social robots, such as Nao, Pepper, iCub, etc., are dedicated to this type of task; however, their physical abilities are very limited [27]. For the physical HRC (pHRC), physical contacts are necessary to perform the task. They can occur directly between humans and robots or indirectly through the environment [28]. pHRC applications are mainly used in industrial environments (e.g., assembly, handling, surface polishing, welding, etc. [15]). pHRC is also used in the Advanced Driver-Assistance Systems (ADAS) for autonomous cars [29].

In this thesis, we want to create a framework that works for both social and physical collaboration in contrast to previous works that are majorly only dedicated to one or the other.

1.3 Setting up of a human-robot collaboration

Many components are involved in setting up a HRC. We have gathered them in Figure 1.3. Four standard steps should be fulfilled to set up a robot that collaborates with humans: perception, DM, motion execution, and evaluation. The works available in the literature tend, on the one hand, to either create or improve the components that define how the robots perform a task in collaboration with humans to increase the robots' decisional autonomy. On the other hand, there are works dedicated to evaluating collaboration. Some studies suggested particular relationship between the four standard steps (cf. Figure 1.3) such as active perception [30], reflex [31], and re-planning [32]. Since they did not necessarily consider HRC while designing those special relationships, we mention them as the state-of-the-art and a future perspective for HRC, but we will not explain them in detail in this thesis.

1.3.1 Perception

Perception is the first step in making the robot collaborate with humans. The robot must be capable of perceiving and recognizing its surroundings by relating the data it receives from the sensors to the environment in which it is. This is essential because the robot must be able to find the objects involved in collaboration with the human, detect the human it is collaborating with and their intentions, and avoid operator confusion when several humans are present in the workspace. Perception in the case of the HRC breaks down into the location recognition and the detection of objects, human agents, their actions, and their intentions.

Object detection

Object detection and differentiation between object types are crucial in HRC. Indeed, the robot must distinguish between the obstacles to be avoided, the fragile objects not to be

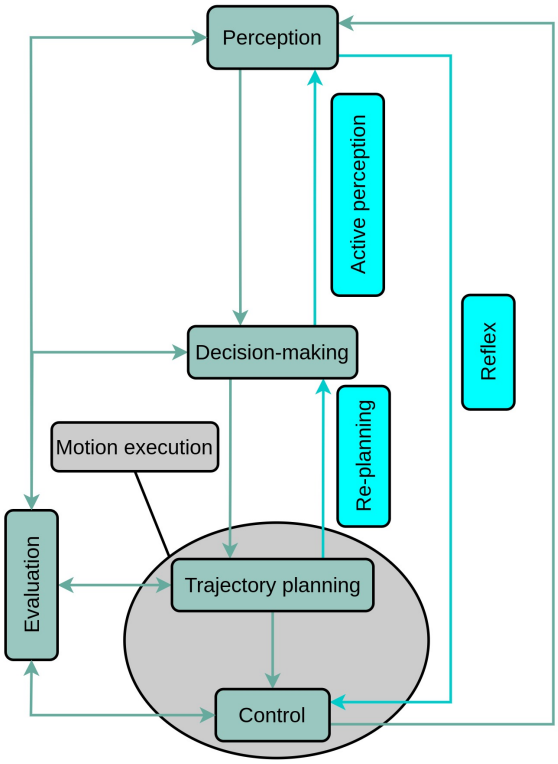


Figure 1.3: Relationship between the different components involved in setting up a HRC. The green arrows show their standard relationship, and the cyan arrows represent their special relationship.

touched because they are easily breakable, and the objects that will be manipulated during the collaboration. Object detection enables the robot to autonomously understand which object is used for what and proactively perform collaborative assistance without specific programming or commanding [33].

Location recognition

Location recognition is highly important within HRC because when the robot can determine its location, it can adapt to its social rules. However, the problem of location recognition remains complicated since identifying the same street corner within the whole city or country when it can be captured under different lighting or change its appearance in time is challenging. The fundamental scientific question is what is an appropriate representation of a place rich enough to distinguish similar-looking areas while compactly representing entire cities or countries [34].

Human presence detection

The robot must be able to detect the human agents around it and especially to distinguish the agents it interacts with. Once the agents involved in the collaboration are identified, the robot should track them all over the interaction. The most used technique in the literature to cope with this problem is face detection [35]. Some approaches go behind face detection, such

as the approach based on age detection (which will be the discriminating point to distinguish people) introduced in [36].

Human intention detection

Human intention detection is necessary to allow the robot to recognize and predict human actions. Therefore, it enables the robot to detect the activity of the human agents and subsequently collaborate appropriately with them. Three steps are required to transmit the human's intention to the robot [37]:

- Intent identification: the human's intention must be defined.
- Measurement: the modality (i.e., sensors) by which the robot measures intent information must be determined.
- Interpretation: The robot's understanding of this measure as a representation of intent information must be defined. The way this measure will be integrated into the robot's decision structure when the data is received by it must also be specified.

Human emotion detection

For cognitive collaboration, such as during therapy, the robot must be able to identify the human's state of mind (i.e., how they are feeling). Approaches to recognizing the person's emotions determine them from facial expressions [38], body language [39, 40], voice [38], or a combination thereof. In [40], other modalities, including physiological signals and text, were also studied, and various combinations of two or more of the modalities mentioned previously.

1.3.2 Decision-making

Once the robot can perceive its surroundings, the DM is the second step in making robots collaborate with humans. This step is crucial to make robots capable of intelligence and intuition to bring them out of industries and integrate them into our daily lives. DM approaches are also utilized to enable robots to adapt to humans when working on a task in collaboration.

Robots can adapt to humans in different situations by implementing five steps in a DM process [41]: *(i)* gathering relevant information on possible actions, environment, and agents, *(ii)* identifying alternatives, *(iii)* weighing evidence, *(iv)* choosing alternatives and selecting actions, and *(v)* examining the consequences of decisions. These steps are usually modeled in computer science using a DM method with a strategy and a reward function [42]. The DM method models the relationship between the agents, the actions, the environment, and the task. The DM strategy is the policy of choosing actions based on the value of their reward calculated by the reward function. The reward function (or utility function) calculates a reward for each action. It allows the robot to distinguish between right and wrong actions to perform. That's

why we decided in the remaining of this thesis to work on the formulation of the reward function to optimize the performance of the collaboration.

The DM can be a separate step in setting up a HRC (such as represented in Figure 1.3), but it can also include the perception, trajectory planning, or robot's low-level control step. In this case, the reward function is modified to evaluate more complex actions, including the perception, trajectory, or robot's low-level control.

1.3.3 Motion execution

After the robot has observed its environment and has made a decision about the actions to be performed, it must proceed to the execution of the actions. If the robot has to execute motion actions, it will need to do two steps: plan the trajectory of its movement and finally use a controller to perform the movement. Trajectory planning and robot control are the third and fourth steps to make the robot collaborate with humans. We have grouped these two disciplines because they are strongly linked within the motion execution step.

During the last few years, the complexity of the tasks achieved by robots in collaboration with humans increased exponentially. Especially since the emergence of assistive robots, teleoperated robots, assisted driving systems used in cars, robotic wheelchairs, exoskeletons, and rehabilitation robots [43]. These robots are used to increase the mobility of people with physical or cognitive deficits, assist surgeons, improve the user's strength capabilities, enhance the amount and intensity of physical therapy, etc. Our point here is to show that the human-robot collaboration domain involves different kinds of robots accomplishing various tasks that sometimes have nothing in common except that the robots should have high physical capacity and precision to achieve the task. This leads the robot to plan a safe, short, and economic trajectory and have satisfactory low-level control of the joints to be as precise and efficient as possible.

Trajectory planning and the robot's low-level control are studied within the literature together or separately. The goal of trajectory planning within HRC applications is to find the best motion trajectory without hitting any obstacle while avoiding harming the human agents [44]. Once the motion trajectory is defined, low-level control of the robot's joints is necessary to execute the motion. Many controllers are utilized in the literature within HRC applications [45, 46], such as position controllers, velocity controllers, force controllers, effort controllers, etc.

1.3.4 Evaluation

As soon as the actions are executed, there is only one thing left to do: evaluate the HRC. This is the last step necessary to make the robot collaborate with humans. The evaluation can concern several components involved in setting up the HRC, such as perception, trajectory, and

control. The evaluation can also be higher level, i.e., based on the benefit that the collaboration brought to the human agents. For this, performance metrics can be used. We can classify these metrics according to the type of task the collaborative team performs. This thesis aims to optimize the HRC. The evaluation step is crucial when the objective is to enhance the HRC. The first step for improving the HRC is to be able to evaluate the collaboration because it allows the robot to detect what is bad and target the aspects to be enhanced.

Some works in the literature tend to evaluate the performance of the collaboration using some performance metrics. On the one hand, some works focus on evaluating one specific metric, as done in [47], where the author assesses several human-robot collaborative teams performing different tasks using the fluency metric. On the other hand, other works create a global framework to evaluate, in general, the HRC based on several metrics. In [48], the authors developed a global framework to assess the HRC based on more than twenty performance metrics, among which the cognitive load and the physical ergonomics.

In Table 1.1, we gather the main performance metrics used in the literature [10–12] for evaluating the HRC classified based on the task types (navigation, perception, management, and manipulation social). We define each metric according to its usage in the different task types. As we can notice from Table 1.1, some performance metrics are common to more than one type of task. Consequently, we introduce some standard metrics used for evaluating HRC in general rather than the performance of a specific task. To assess a HRC, we have to choose the best metrics with respect to the goal of the application and the benefit the human agents can expect from the collaboration. In the applications we utilized for evaluating the general framework we propose within this thesis, we consider the time to completion, the number of human errors, and the robot manipulation dexterity.

1.4 Thesis context

This thesis receives funding from the Auvergne-Rhône-Alpes Region in France through the ATTRIHUM project, which includes four laboratories affiliated with the Université Clermont Auvergne (UCA): Institut Pascal, LAPSCO, LIMOS, and PHIER. This project is dedicated to human-robot interaction. It aims, on the one hand, to measure the impact of humans on robots and vice-versa. On the other hand, the project's final goal is to optimize human-robot interaction. From our side in Institut Pascal, we focus on enhancing the performance of HRC by improving the robot's decision-making process and its manipulation dexterity.

1.5 Thesis objective

We want, through this work, to optimize the HRC by bringing more benefit to the human agents from the collaboration. One way to do that is to improve the robot's perception and

Task	Performance metrics	Definition or usability
Navigation	Failure rate	Percentage of navigation tasks completion failure
	Accuracy	The accuracy of the navigation
	Ergonomy or posture	Human ergonomy or posture
	Time to completion	The time needed to complete the task
	Rapidity	The time needed by the robot to adapt itself to the human or vice-versa
Perception	Velocity	The speed of the perception of the robot
	Accuracy	The accuracy of the navigation
	Time to completion	The time needed to complete the task
	Fluency	The fluency of the perception
	Effectiveness	Percentage of the success of the robot's perception
	Number of errors	The number of failures in the robot's perception
Management	Time delivery	The time needed to deliver the request from the robot to the human
	Time request	The time needed by the human (operator) to notice the request
	Number of human errors	The number of times the human cannot identify the situation with awareness
	Number of robot errors	The number of times the robot misinterprets human desires
	Trust	Trust of the human in the robot
	Number of actions	The number of actions needed to accomplish the task from the human and the robot
	Cognitive load	The workload required for the human to adapt to the robot
Manipulation	Positional accuracy	The accuracy of the position reached by the robot
	Positional repeatability	The repeatability of the robot to reach the same position
	Velocity	The speed of the robot to do the manipulation
	Time to completion	The time needed to complete the task
	Rapidity	The time needed by the robot to adapt itself to the human or vice-versa
	Cognitive load	The workload required for the human to adapt to the robot
	Ergonomy or posture	Human ergonomy or posture
	Dexterity	The robot's dexterity in doing the manipulation
	Effort or force	The physical effort (or force) that the human must provide to perform the manipulation
	Number of human errors	The number of times the human cannot identify the situation with awareness
	Number of robot errors	The number of times the robot is misinterpreting human desires
	Number of actions	The number of actions needed to accomplish the task from the human and the robot
Social	Persuasiveness	The ability of the robot to persuade the human about something
	Trust	Trust of the human in the robot
	Engagement in social characteristics	Engagement in social characteristics such as emotion, dialogue, and personality. The engagement can be measured through the robot's acquisition time for capturing human attention and the duration of holding human interest
	Compliance	The compliance of the robot in appearance, adherence to norms, etc.
Common metrics	Effectiveness	The percentage of the mission that was accomplished with the designed autonomy
	Time to completion	The time needed to complete the task
	Number of human errors	The number of times the human cannot identify the situation with awareness
	Number of robot errors	The number of times the robot misinterprets human desires
	Number of actions	The number of actions needed to accomplish the task from the human and the robot
	Cognitive load	The workload required for the human to adapt to the robot
	Self-awareness	The robot knows its accuracy
	Autonomy	The robot autonomy

Table 1.1: Some metrics considered for the evaluation of HRC classified based on the task types [10–12].

control techniques. The other one is to enhance the DM process used by the robot by considering

the performance metrics. The second way allows optimizing the collaboration from a higher level since we can consider different metrics, especially as the robot's perception and control can be included in the robot's DM process. This work aims to increase the performance of HRC through the DM process of the robots.

DM techniques are used to make robots able to adapt themselves to humans while accomplishing a task in collaboration. A DM process comprises, as mentioned previously, three main parts: a DM method, a DM strategy, and a reward function (or utility function). Previous studies' goal in the context of HRC is to adapt the robot to human behavior to achieve a task without considering how well the interaction unfolds. Some works propose dedicated frameworks to improve the HRC performance by considering a fix performance metrics that cannot be easily changed within their architecture.

We aim to develop a global framework in which we focus the reward function on increasing the performance of the collaboration between humans and robots based on several changeable performance metrics. For that, we isolate the impact of the performance metrics in the reward function such that we can change them within the same framework. The reward function will comprise three parts: the first is for optimizing the metrics, the second is for dealing with the task constraints, and the third is for considering the robots' physical abilities and ameliorating their manipulation dexterity. Figure 1.4 presents an overview of our framework. We test our framework through a challenging task in terms of DM process in Part II and a complex task in terms of control in Part III.

1.6 Contributions

Our first contribution is the proposed DM framework we introduced in Figure 1.4 that can deal with the change of the performance metrics used for optimizing the HRC. State-of-the-art techniques consider the HRC as an optimization problem in which the reward function is defined to accomplish the task regardless of how well the interaction is performed. When the performance metrics are considered, they cannot be easily changed within the same framework.

In contrast, our DM framework can easily handle the change of the performance metrics from one case scenario to another. Our method treats HRC as a constrained optimization problem where the utility function (or reward function) is split into three main parts. Firstly, a part that deals with the performance metrics to assess the performance of the collaboration. It is the only part that is altered when modifying the performance metrics. It allows for control over how the interaction proceeds and ensures that the robots' behaviors will be directly adjusted to those of the human agents involved in the collaboration. Secondly, a part that handles the task constraints by specifying how to complete the task. Thirdly, a part that contains the robots' physical abilities. Accordingly, the utility function may be designed to enhance the robot's

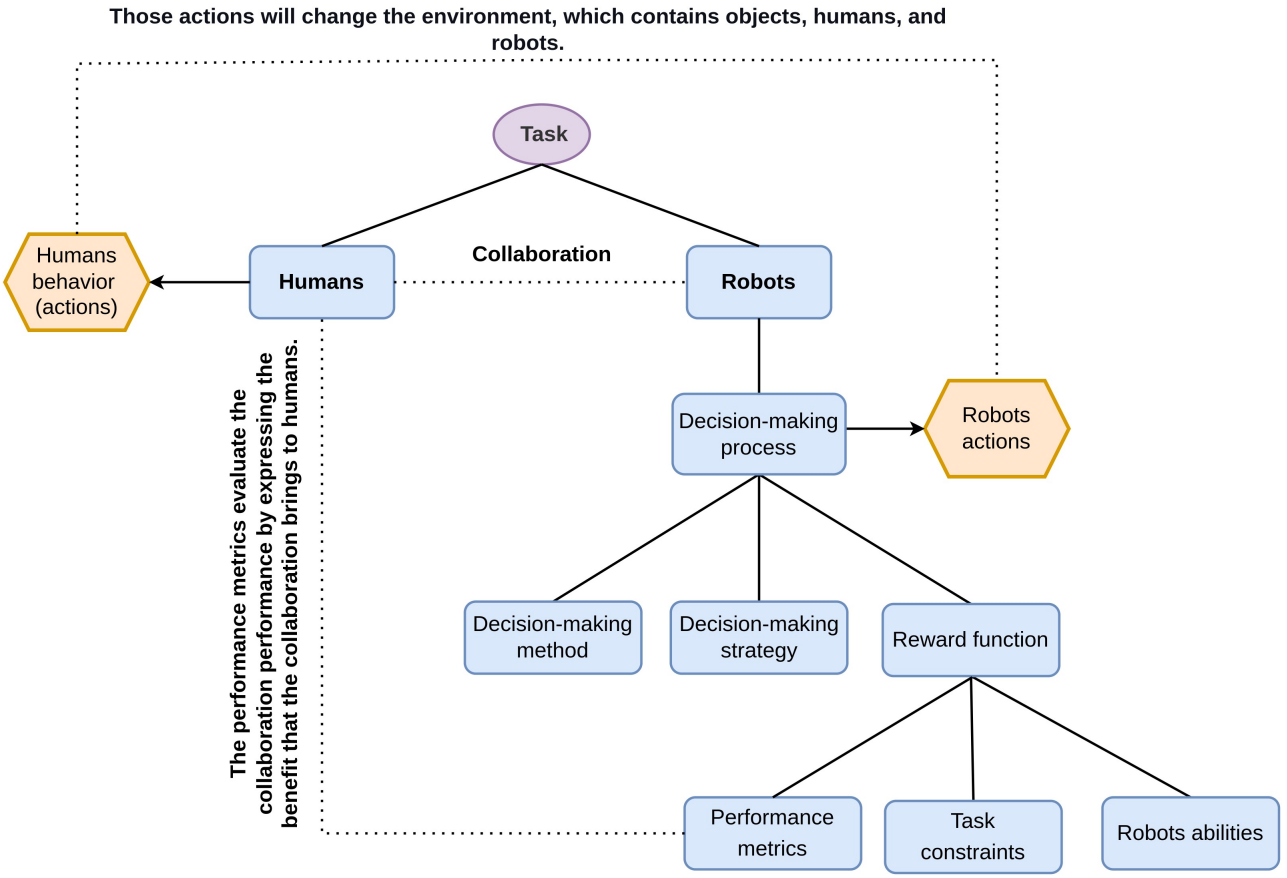


Figure 1.4: Overview of our global framework for optimizing the HRC through improving the robot's DM process.

dexterity in manipulating objects.

We evaluated our framework through two applications. Firstly, we tested our framework for achieving an assembly task by a human and a Nao robot. Since social robots such as Nao have significant limitations for manipulation (their motions are slow and inaccurate) that are difficult to improve or even impossible, we concentrated the reward function on enhancing the performance of the collaboration based on several changeable performance metrics without ameliorating the agents' abilities. We consider within our framework a changeable unrestricted number of performance metrics that are usually optimized no matter how the human is behaving. To summarize, the contributions of our framework when applied to the assembly task are:

- the easily change of the performance metrics from one scenario to another without changing anything in our formalization except the part in the reward function related to the metrics, and

- the improvement of the collaboration performance with the possibility of improving or not the robot's manipulation dexterity since we isolate that part in the reward function.

Secondly, to further optimize HRC performance, we have to enhance the manipulation

dexterity of the robot. For that, we start by improving the robot's manipulation dexterity while it performs a soft objects (e.g., deformable linear objects) deformation task without collaborating with humans. For this application, we consider the improvement of the robot's manipulation dexterity in the reward function thanks to a Deep Reinforcement Learning (DRL) algorithm. Our proposed solution is easier generalizable than the DRL frameworks presented in the literature. Indeed, their agent (either a single-arm robot or a dual-arm robot) is trained to perform a manipulation from constant initial to constant target deformations, and it is not trained to deal with different configurations. To sum up, the contributions of our framework when it is applied to a manipulation task are:

- Its generalizability, i.e., we train the agent only once (using a specific Soft Object (SO)), and it can deform the SO starting from a different initial position and end up with a different desired shape. Moreover, the agent can make the SO reach an untrained position, i.e., we train the agent on a small workspace and test it on a bigger one.
- It can achieve a more complex task than the ones performed in the literature. Indeed, the agent deforms a foam bar by making some selected mesh nodes reach the corresponding desired positions in 3D space, potentially involving complex torsion motions.

In the future, our idea is to replace one of the two robotic arms with a human in real tests to make the human-robot collaborative team able to perform a co-manipulation task involving SOs.

1.7 Summary of the Thesis

This thesis is organized as follows. Part I makes a general introduction about HRC and sums up the state-of-the-art of the robot's DM approaches for HRC in two chapters. Chapter 1 makes a general introduction about HRC and presents how the HRC can be classified. It also introduces the main disciplines required for setting up a HRC and the performance metrics used to evaluate it. The robot has to perform four main standard steps: perception, DM, motion execution (trajectory planning and low-level control), and evaluation. This chapter also presents an overview of our proposed framework for optimizing HRC performance based on several changeable metrics and summarizes our thesis contributions. Since this work focuses on improving the DM by evaluating it through performance metrics to enhance the HRC, Chapter 2 reviews the work in the literature that uses the DM process of the robot to optimize the HRC performance. It presents the existing DM processes in the state-of-the-art that either focus on high-level decisions or include one or more of the other HRC setting up steps (perception, trajectory planning, or low-level control).

Part II introduces our DM framework in the context of HRC. Contrary to the literature DM frameworks, which cannot easily manage the change of considered performance metrics

to optimize the HRC, our can easily handle the performance metrics change from one case scenario to another. This part is split into two chapters. Chapter 3 presents our framework from the mathematical point of view. Chapter 4 presents the carried out experiments in real and simulation on an assembly task (i.e., a game based on a construction kit). In this application, we optimize the HRC without having to increase the robot's physical abilities.

In Part III, the goal is to integrate the improvement of the robot's manipulation dexterity by making it accomplish a more complex task (i.e., deforming a SO) while enhancing its low-level control. This part contains three chapters. Chapter 5 introduced the background on the deformation of soft linear objects, deep reinforcement learning approaches, and the deep deterministic policy gradient algorithm. Chapter 6 presents the results of the conducted experiments on the SO to prove the effectiveness of enhancing the robot's physical dexterity thanks to a deep reinforcement learning approach. We train the robot only once (using a specific deformable object), and it can deform the SO starting from a different initial position and ending up with a different desired shape. Moreover, the robot can make the deformable object reach an untrained position. Chapter 7 presents the results of the conducted experiments on the SO manipulated by a dual-armed system. Chapter 8 is the general conclusion of this thesis which summarizes the work achieved within this thesis and discusses our perspectives.

1.8 List of Publications

1.8.1 Journal

Hani Daniel Zakaria, M., Lengagne, S., Corrales Ramón, J. A., & Mezouar, Y. (2021). General Framework for the Optimization of the Human-Robot Collaboration Decision-Making Process Through the Ability to Change Performance Metrics. *Frontiers in robotics and AI, 8,* 736644. `https://doi.org/10.3389/frobt.2021.736644`

1.8.2 Conference

Hani Daniel Zakaria, M., Aranda, M., Lequièvre, L., Lengagne, S., Corrales Ramón, J. A., & Mezouar, Y. (2022). Robotic Control of the Deformation of Soft Linear Objects Using Deep Reinforcement Learning. In *2022 18th IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE.

Chapter 2

Decision-making process

Contents

The previous chapter introduced the main fields necessary for setting up a Human-Robot Collaboration (HRC). We decided in this thesis to focus on the robot's Decision-Making (DM) process to optimize the collaboration by keeping the robot's abilities unchanged in the first place. In the second place, we used the robot's DM even to improve its manipulation dexterity. In this chapter, we introduce the most popular methods, strategies, and utility functions used in the DM process within HRC.

A DM method models the relationship between the agents, the actions, the environment, the task, etc. A strategy defines how to select the optimal actions (or alternatives) each agent can make based on the reward calculated by the reward function (or utility function) for each action. All methods and strategies can be used to achieve different tasks, and there is no pre-written rule that implies that one will necessarily perform better than the others.

2.1 Decision-making methods

DM methods are used, as mentioned before, to model the relationship between the task, the agents accomplishing it, their actions, and their impact on the environment. In this section, we introduce the DM methods used in the state-of-the-art. Probabilistic methods, deep learning, and game theory are among the most widespread DM methods. In the literature, fewer works utilize analytic hierarchy process, optimization methods, and ensemble learning as DM methods. Table 2.5 introduces the advantages and disadvantages of these DM methods.

2.1.1 Probabilistic methods

Probabilistic methods are the first and most widely used in DM processes. Markov's decision processes (e.g., Markov chains) are the most used ones. This is because a Markov Decision Process (MDP) models the uncertainty of the effects that the action will have on the environment as well as the uncertainties that the observations will correctly represent the current state of the environment and the agents. A MDP evaluates each state-action pair by attributing to the action a reward that is calculated relative to the HRC's objectives [49]. For example, a partially observable MDP is used to compute the human agent's trust in the robot. This computed trust is integrated into the robot's DM process so that it can infer the human agent's trust in it all over the interaction, reason about the effect of its actions on human trust, and choose actions that maximize team performance [50].

Some studies use other DM methods such as Gaussian processes (e.g., Gaussian mixtures), Bayesian optimization (e.g., likelihood functions), and belief functions. A Gaussian process is helpful when the stochastic process random variables follow a normal distribution. It is utilized mainly to detect human actions and intentions, such that the robot considers them while choosing its actions. In [51], a learning method based on a Gaussian mixture model representing the human's movement trajectories is utilized to predict the space occupied by the human agent in the workspace. This information is incorporated into the robot's DM process to prevent the robot from hitting the human when it decides on the best trajectory to perform.

Bayesian optimization is a probabilistic approach based on Bayesian inference [52]. This means that we exploit previously observed (known) events to predict the probability of future events using a probability density function such as a likelihood function. Bayesian optimization is

mainly used to compensate for uncertainties, for example, in the case of a robot that complements the human's capabilities by compensating for uncertainties of different tasks [53].

Probabilistic DM method	Advantages	Disadvantages
Markov decision processes	They are generalizable because they are adaptive to changing and unknown characteristics of the environment [54]. They have an excellent rapidity, accuracy, and flexibility ratio [55].	They have a large number of settable parameters; therefore, they are time-consuming [54]. In the case of an inaccurate model, the results will be wrong since it might lead to executing undesired behaviors [55].
Gaussian processes	They can be generally used as simple methods with fewer settable parameters. They provide safety guarantees by ensuring the error prediction [56].	The data have to follow a Gaussian distribution (i.e., normal distribution) [57]. For applications that require high computational complexity, the Gaussian process is used offline on batches of data. This is inadequate with applications where a fast adaptation through online learning is necessary to ensure safety [56].
Bayesian optimization	It models and considers the worst-case scenario [58].	The computational cost will be high for complex problems because calculating the worst-case scenario will take lots of iterations [58].
Belief functions	They can deal with ignorance, imprecision, and uncertainty [59].	They often lead to cumbersome or even intractable calculations [59].

Table 2.1: Advantages and disadvantages of probabilistic DM methods.

The theory of belief functions (also named Dempster-Shafter theory or evidence theory) generalizes the Bayesian theory [59]. It is a general framework for reasoning with uncertain information. In other words, this theory combines available evidence from different sources to evaluate the degree of belief the agent can have in that evidence, thanks to the belief functions. In [60], the authors develop a system that enables multimodal sensing of ongoing human actions and predict their behaviors during a human-robot collaboration in the operating room. They used the Dempster-Shafter theory for sensor fusion so that the robot robustly knows the human action and considers it when making decisions.

Furthermore, some complex tasks require combining two or several probabilistic DM methods. For example, in [61], a robot has to move in coordination with a human while transporting an object. Consequently, the robot estimates the human impedance and its motion intention for making decisions about its movements thanks to Bayesian optimization. Furthermore, it considers the stiffness of the environment by calculating the covariance of a Gaussian mixture model. Table 2.1 presents the advantages and disadvantages of each probabilistic DM method.

2.1.2 Deep learning

The interest in using Deep Learning (DL) in DM methods began very early due to unsatisfactory results of probabilistic methods in managing uncertainties in complex tasks. Nowadays, the usage of DL is widespread as a DM method. DL is a variety of artificial neural network-based machine learning in which input attributes (i.e., data) are processed through successive layers to extract higher-level features progressively [62]. A neural network consists of several nodes or neurons. Within each node, there is a set of inputs, weight, and a bias value [63]. The bias shifts the activation function (which decides whether a neuron should be activated or not [64]) by adding a constant to the input data. The weight transforms the input data within the network's hidden layers [62, 63]. Weights and biases are both learnable parameters inside the network. A model consisting of a set of weights and biases is used to establish the relationship between the input attributes and the desired output attributes by predicting the values of the output attributes [63]. The learning goal is to reduce errors between those predictions and the desired output attributes. This can be made using an optimization algorithm (based on a descent gradient approach), which changes the weights and the biases so that the errors between the predictions and the desired outputs are reduced [62, 65]. The deep neural networks are classified into: supervised learning, unsupervised learning, and Deep Reinforcement Learning (DRL) [66]. The advantages and limitations of each DL method are presented in Table 2.2.

On the one hand, supervised learning attempts to discover the relationship between input attributes (i.e., input dataset) and a target or output attribute (e.g., a class label in classification or a real number in regression) [65, 67]. A model represents this relationship. The goal is to learn the model which predicts the correct value of the target attribute given the new input attribute [68]. For example, a dual-arm robot collaborates with a human to perform an assembly task. The robot does the intended action in response to the specific gesture made by the human when it recognizes it, thanks to a supervised learning method [69].

On the other hand, unsupervised learning tries to find patterns or structures output for the input attributes [70]. The goal is to build representations of the input that can be used for the DM process, predicting future inputs, etc. [71]. It is used for clustering and dimensionality reduction. For example, an unsupervised learning algorithm detects human motion to ensure their safety while collaborating with robots to achieve industrial tasks. The robot's objective is to estimate the remainder of the human motion trajectory given an observed part of the human motion to prevent interference while performing a complementary task [72]. To summarize, supervised learning utilizes labeled input and output data, while unsupervised learning does not.

As regards DRL, the agent's goal is to learn how to interact with its environment. The agent interacts with the environment by making actions [71]. These actions change the state of the environment to the next state. The agent's observation of the environment describes the

changes that happened by moving from the current state to the next one [22]. As a result of making an action, the agent also receives from the environment a reward that evaluates the action taken with respect to the desired learning goal [71]. The long-term goal of the agent is to maximize the future rewards it receives over iterations (or steps) [22]. Figure 2.1 presents this DRL procedure. DRL is close related to decision theory and control theory [71]. The usage of DRL in HRC applications is vast. It can be used for cognitive-oriented interaction, such as task allocation, or physical-oriented interaction, such as co-manipulation. In [73], a DRL algorithm is used for optimizing an assembly task allocation. Whereas, in [74], a DRL algorithm is utilized to make the robot adapt to human movements by adjusting its motion control while co-manipulating (lifting) an object.

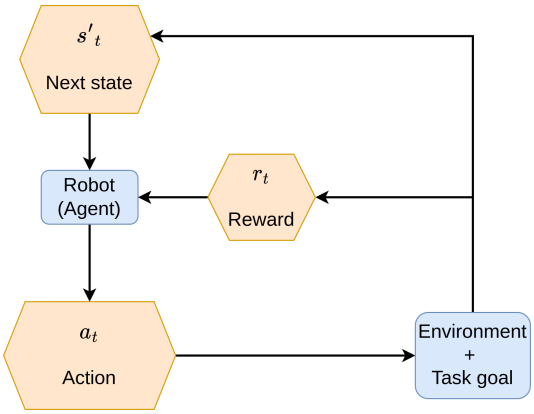


Figure 2.1: DRL procedure.

DL method	Advantages	Disadvantages
Supervised learning	It can tolerate noisy data and manage to classify untrained patterns. It can be used when the knowledge about the relationship between attributes and classes is slight [75].	It involves a long learning time and needs large training datasets; therefore, it is unsuitable for all applications. It requires several parameters to be determined empirically, such as network structure, number of hidden layers, and number of neurons in each layer [75].
Unsupervised learning	It can perceive what the human mind cannot. There is less complexity compared to supervised learning because there is no need to interpret the associated labels [76].	Since there is no label or output measure to certify their usefulness, it is difficult to determine whether the results will be valuable. The results are often less accurate than using supervised learning [76].
Deep reinforcement learning	The agent learns to interact with its environment and is getting better while learning [77].	The agent is poorly performing at the beginning of the training [77].

Table 2.2: Advantages and disadvantages of DL methods.

Two main issues should be overcome when using DRL: the computation time [78] and using a real robot to perform the task while the training was carried out in simulation [79]. The use of complex DRL networks increases learning computation time. This increase is related to the

number of objectives the chosen actions must satisfy, especially if these objectives conflict [78]. One approach to resolving this problem is exploiting the Transfer Learning (TL) principle to apply it to DRL [80]. TL is based on one of the human psychological learning principles. When a human learns to perform a task, their knowledge serves as a starting point for performing another task that has similarities with the first. When this concept is applied to DRL, the information about the different states of the environment and the learning about the correct actions to perform are shared across tasks, ensuring a faster learning process [78, 80]. For example, this solution has been adopted to transfer models learned by agents among human-robot collaborative assembly industrial tasks [81].

Another solution to this problem is to use Imitation Learning (IL), which aims to mimic human behavior to complete the application [82]. The starting DRL policy is the one learned through IL which reduces the DRL computation time [83]. The DRL algorithms combined with IL and an environment representation are the main elements for successful learning with a performance level as good as humans when performing the same task. The policies obtained can generalize well and adapt rapidly to new scenarios [82]. For instance, this approach has been used to make a human-robot collaborative team achieve a co-manipulation task (e.g., lifting a table) [84].

Another approach is to use parallel learning, which leads to executing multiple agents in parallel on various environment instances. Learning from parallel agents experiencing various different states ensures that the training data are decorrelated and can be collected faster [85,86]. This improves the overall learning time while achieving a better result from the generalization point of view [87]. For example, this solution has been adopted to optimize the assembly task completion time achieved by a human-robot collaborative team [88]. Table 2.3 presents the advantages and limitations of each approach we presented to solve the DRL computation time issue.

The second problem is that the agent's training is made using simulated environments, and the target is to test the agent in real. The training is done in simulation due to the limitations of gathering real-world data and the risks of training a real robot [79]. When the models are implemented in real robots, the difference between the simulated and real worlds degrades the performance of the learned policies [89]. To close this sim-to-real gap and achieve more effective policy transfer utilizing sim-to-real transfer learning techniques, multiple research efforts are currently focused on this topic [79]. The selected sim-to-real techniques change depending on the task to be performed, and the sensors used in real [79, 90]. Domain Randomization remains the most widely used sim-to-real technique in the literature but not necessarily the best [90]. It consists of sampling simulation parameters (such as camera position, light position, textures, etc.) from probability distributions centered at a noisy estimate of the ground truth [79, 90]. As a result, the agent can disregard small changes in the environment,

making it more robust to domain changes. There are other sim-to-real techniques such as [79]: Domain adaptation, imitation learning, meta-learning, knowledge distillation, etc. For instance, a sim-to-real approach has been used to construct a human-robot shared control framework for robotic surgery [91].

Approach	Advantages	Disadvantages
Transfer learning	Transfer learning will help perform deep learning tasks with fewer data and resources than needed for "classical" learning techniques [92].	Transfer learning only works if the initial and target tasks are similar enough to enable transferring what the agent learned and avoid a negative transfer [92].
Imitation learning	The adaptability and generalizability of the trained model are improved through imitation learning [82]. It boosts learning efficiency. It can be used in conjunction with other learning mechanisms [83].	Demonstration data or a means of obtaining a supervised signal of desired behavior are necessary for imitation learning [82]. Obtaining such data is challenging for many applications or even impossible. The quality of the demonstrator will limit the performance of the agent [83].
Parallel learning	Parallel learning allows the agent to learn faster [93]. It makes the agent generalize the learned model easier [87].	Parallel learning requires using many CPU cores that have high computing power [87]. It is harder to implement and debug.

Table 2.3: Advantages and disadvantages of the approach used to reduce the Deep Reinforcement learning (DRL) computation time.

2.1.3 Graph theory, matrix approach, and game theory

Graph theory and the matrix approach represent a task, including the environment and the agents, as a graph (i.e., tree) or in a matrix shape [94]. These ways of representing a task are interesting because they give a unified formalism for many different applications. Graph theory has been utilized, for example, to represent the relationships between the main elements of collaborative workspaces within the anthropocentric paradigm for HRC developed in [95].

Game theory is originally a branch of applied mathematics that provides a model for analyzing games (i.e., situations) in which players (i.e., agents) make interdependent decisions, whether they are collaborating or competing [96]. The game theory uses either the graph theory or the matrix approach to represent the task. Game theory investigates the interactions between players, while graph theory and matrix approach focus on the large-scale topology of an interaction network.

Game theory methods in DM processes have only recently been exploited. They can model most of the tasks performed by a group of agents (players) in collaboration or competition, whether the choice of actions is simultaneous or sequential. In the case where the choice of actions is simultaneous, the task is represented using the normal form, also called matrix form modeling [97]. When the choice of actions is sequential, the task is represented by the

extensive form, also known as tree form modeling [98]. Both representation ways are presented in Figure 2.2. Game theory can be helpful for tasks where agents have complete knowledge of the environment and other agents (these "games" are called perfect information games). It can still manage when agents do not have all the information (these "games" are called imperfect information games). This signifies that game theory can deal with uncertainties and a lack of information.

The game theory methods have been used in different HRC applications, for instance, in analyzing and detecting the human agent behavior to adapt the robot one to it [99, 100]. This led to a better collaboration performance. Game theory has also been utilized in HRC in mutual adaptation to achieve industrial assembly scenarios [101].

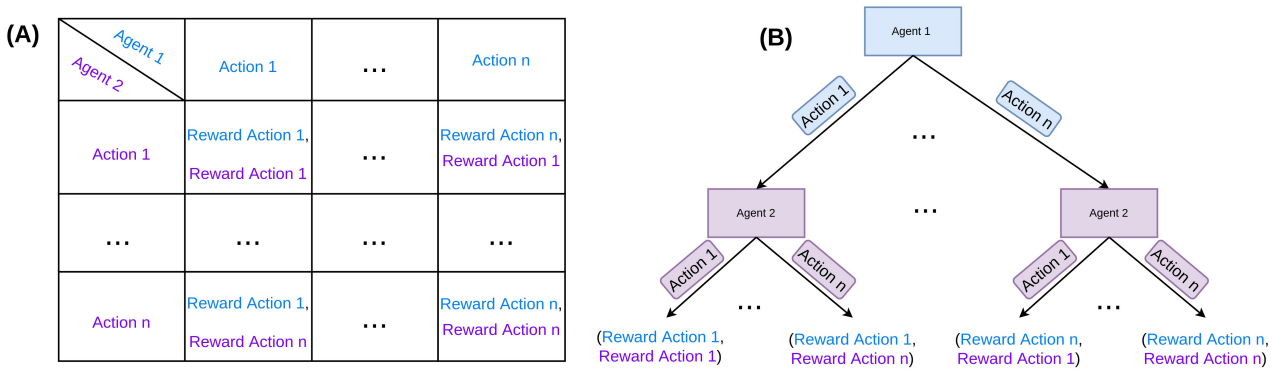


Figure 2.2: Task representation ways in game theory. **(A)** Normal form. **(B)** Extensive form.

2.1.4 Analytic Hierarchy Process

Other classification-based DM methods represent the task, the agents, and their actions as a scheme, such as the Analytic Hierarchy Process (AHP). It is a classification process that orders the task hierarchically, with the objective set coming first, then the metrics at the next level, and finally the solutions (or actions) at the lowest level [102]. Figure 2.3 presents an AHP classification. This method is commonly utilized for applications such as for a robot that performs an assembly task while ensuring the safety of the human agents [103].

2.1.5 Optimization methods

The optimization methods have been utilized in some HRC research as DM methods. Among them are the metaheuristic optimization algorithms (or metaheuristics) and the Response Surface Methodology (RSM).

A metaheuristic is a high-level, problem-independent algorithmic framework that provides guidelines for finding a sufficiently good solution to an optimization problem, particularly with incomplete information, imperfect information, or limited computational capacity [109]. Even though there are many others, the most famous instances of metaheuristics include

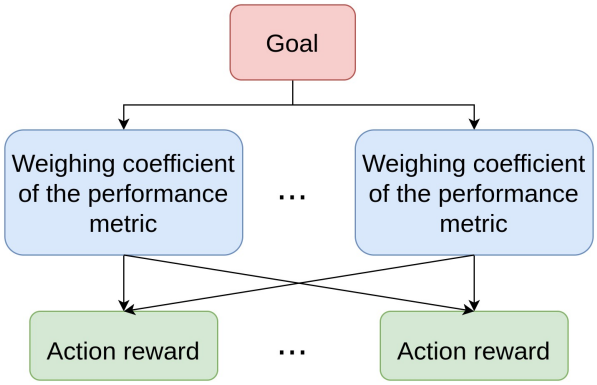


Figure 2.3: Example of an AHP classification.

Optimization method	Advantages	Disadvantages
Metaheuristics	For large or complex problems, metaheuristics are often superior alternatives to more traditional (exact) approaches [104]. They often offer a better trade-off between solution quality and computation time. Metaheuristics are more flexible than exact methods because they do not restrict the formulation of the optimization problem (i.e., requiring constraints or objective functions) [105].	Because of their flexibility, metaheuristics require significant problem-specific adaptation in order to perform well [105].
Response Surface Methodology (RSM)	RSM reduces the number of trials, which saves time and cost while determining the interaction between the independent variables and modeling the system mathematically [106].	The experimental data are fitted to a second-order polynomial model. It is incorrect to assume that all curvature-based systems can be modeled using a second-order polynomial model [107]. Additionally, it is imperative to do absolute experimental validation of the model's estimated values [108].

Table 2.4: Advantages and disadvantages of the optimization DM techniques.

genetic/evolutionary algorithms, neighborhood search, bee algorithms, and ant colony optimization [105]. In contrast to exact methods, which ensure that the best answer will be discovered in a limited (albeit frequently impossible) amount of time, metaheuristics do not. Consequently, they are created to find a "good enough" answer in a "fast enough" computation time [105]. As a result, they are immune to the combinatorial explosion, a phenomenon in which the complexity of the problem has an exponential effect on the computation time required to find the optimal solution [109]. For example, the bee algorithm and the neighborhood search among metaheuristics were used to consider the robot selection and assignment within a HRC in industrial assembly lines [110, 111].

The goal of the RSM is to design a suitable functional relationship (i.e., response surface) between the input variables and the response of interest (i.e., the desired output). Although the exact function is uncertain, a low-degree polynomial model can provide an approximation [112]. For instance, such a method has been used to do facial recognition of a human interacting

with a robot by aligning the contour model of their face with their actual face [113]. RSM can also enable the robot to adjust its movement when achieving a task in collaboration with a human agent by learning through human guidance [114]. Table 2.4 presents the advantages and disadvantages of optimization techniques.

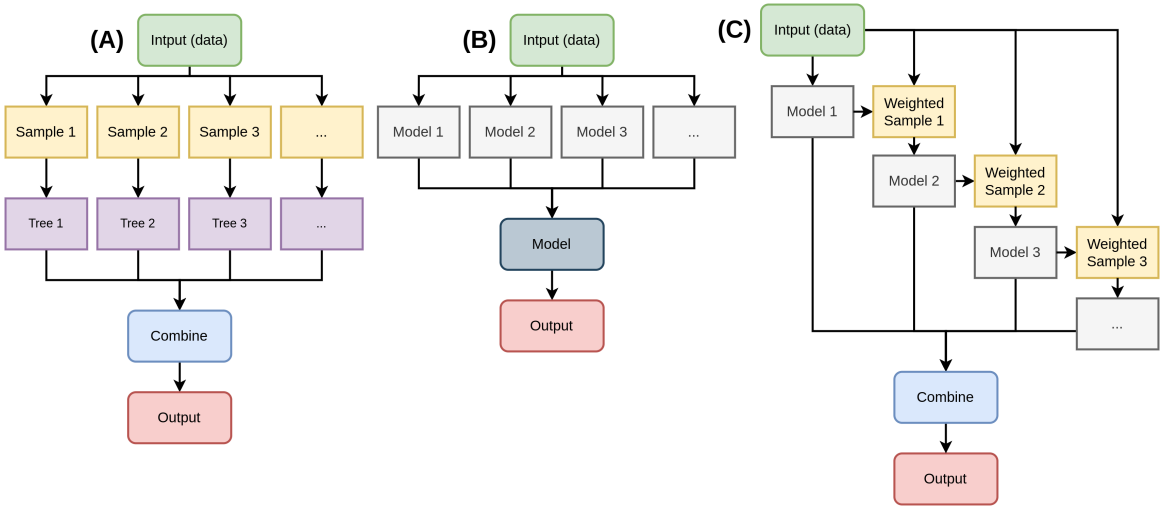


Figure 2.4: The three principal categories of ensemble learning techniques. **(A)** Bagging ensemble. **(B)** Stacking ensemble. **(C)** Boosting ensemble.

DM method	Advantages	Disadvantages
Probabilistic methods	They are generalizable because they can adapt to changes since they consider uncertainties, lack of knowledge, and risk evaluation [115].	Their computation time is long, and it is complicated to interpret their results [115].
Deep learning	They can analyze and learn massive amounts of data in a supervised or unsupervised way and learn how to interact with the environment [76].	They require a lot of data for training, the learning is slow, and the result is not always guaranteed [76].
Graph theory	It is an intuitive technique whose results are easy to understand and interpret. It can be used in combination with another DM method [116].	It is unstable because a small change in the data leads to a big change in the graph (tree) structure [116].
Game theory	It can deal with uncertainties and lack of information about the environment and the other agents' actions [98].	The protocols for interaction should be precisely defined, whereas, in the real world, they are often ambiguous for multi-human interactions. It is impossible to generalize the solution since it depends on the task and the interaction between the agents [98].
Analysis Hierarchy Process (AHP)	Pairwise comparisons make the agents able to compare alternatives relatively easily [117].	In some cases, some irregularities might appear for the alternatives estimation since this strategy is based on pairwise comparisons [102].
Optimization methods	Optimization methods reduce the time of collecting the data [118]. They can adapt to new situations when the environment is constantly changing, as is the case for many real applications.	Since the optimization solution is global and there is frequently more than one reason for an unexpected output or outcome, it is more challenging to settle a good model and debug [119].
Ensemble learning	It is a robust DM method with a reduced computation time compared to classical learning methods [120].	It is complex to set up because it has a lot of parameters, classifiers, and training ensembles to adjust [120].

Table 2.5: Advantages and disadvantages of DM methods.

2.1.6 Ensemble learning

Ensemble learning is a general meta-approach to machine learning that aims to improve predictive performance by mixing the predictions from various models [121]. Three techniques rule the field of ensemble learning, even though there are an apparently infinite amount of ensembles you can create for your predictive modeling [122]. So much so that it is a topic of study that has given rise to numerous more specialized approaches. Bagging, stacking, and boosting are the three primary categories of ensemble learning techniques. Each technique main principles are [121]:

- Bagging entails averaging the predictions from many decision trees fitted to various samples of the same dataset.
- Stacking implies fitting numerous different model types on the same data while using a separate model to learn how to combine the predictions optimally.
- Boosting includes sequentially adding ensemble members that correct previous models' predictions and generate a weighted average of the predictions.

Figure 2.4 presents the principle difference between the main three categories of ensemble learning techniques. Ensemble learning is mainly utilized in HRC to make the robot learns to collaborate from human demonstrations. For example, a robot learned to write Chinese characters by imitating a human while writing [123]. In [124], a robot learns new words by playing with human agents in the "name game": one agent says a word that belongs to a theme (e.g., animals), and the next agent has to find another word that belongs to the same theme, and that begins with the last letter of the previous word.

2.2 Decision-making strategies

The DM strategy is the policy of choosing actions based on the value of their reward calculated by the reward function (or utility function). This section presents the most used strategies for multi-criteria DM in HRC as well as some of their application areas. Table 2.6 presents the advantages and limitations of each strategy and some usage examples within HRC. Theoretically, any strategy can be associated with any DM method. However, practically, only most of the strategies can be coupled to any DM method, whereas some are exclusively used in combination with a specific DM method. Let's define the most well-known strategies:

- Dominance: All the actions whose rewards are dominated by others are eliminated. This strategy is the most used one in DM processes [98]. It can be coupled with any DM method.
- Minimax (or minmax): Agents try to either lower their own maximum losses or lower

the maximum reward that the other agents will receive [125]. This strategy has been associated with many DM methods such as Game theory and DL methods.

- Maximin (or maxmin): It tries to maximize each agent's gain when the others try to do everything to minimize their rewards [98]. This strategy has been associated with many DM methods such as Game theory, MDP, and DRL.

- Pareto optimality: An action profile is Pareto optimal if we cannot change it without penalizing at least one agent [98]. This strategy is mainly used in Game theory.

- Nash Equilibrium (NE): Each agent responds to the others in the best possible way [126]. The best response is the best actions an agent can choose, whatever others have done. This is the main strategy used in Game theory.

- Stackelberg duopoly model: The agents decide sequentially, one agent (the leader) makes their decision first, and all other agents (followers) decide after. The optimal action of the leader will be the one that maximizes its own reward and minimizes the follower's rewards [127]. This means that the leader always has the biggest reward. This strategy is primarily used in Game theory.

- ELECTRE (Elimination EtChoix Traduisant la REalité): It was initially used to eliminate the unacceptable alternatives in the real test. Then it was improved to allow the agent to select the best choice with the most benefits and the least conflicts while considering the performance metrics [102]. Regarding outranking the alternatives based on prioritizing the choices among the performance metrics, the ELECTRE strategy performs well [128]. It has been coupled with probabilistic DM methods.

- PROMETHEE (Preference Ranking Organization METHods for Enrichment Evaluations): They are also outranking strategies such as ELECTRE. Still, they differ in the way of calculating alternative ranks, i.e., the alternative rank is computed through a positive and a negative outranking flow for each alternative [129]. Those strategies were introduced to overcome one of the limitations of ELECTRE by identifying the alternatives' strengths and weaknesses [117].

- Lexicographical order: It is based on a straightforward ranking of the actions by priority. A choice alternative is superior to another alternative if and only if it is better than the other alternative in the most crucial performance metric on which the two alternatives vary [130]. In other words, the lexicographic choice rule does not permit trade-offs between the different performance metrics if there is a strict preference. For example, let's consider the scenario where precision is prioritized over speed. A speed difference cannot make up for a precision difference in that situation. The lexicographic decision rule selects the action with the highest precision regardless of speed. This strategy has been used with

probabilistic DM methods and tree structures.

- Consensus: This strategy is used when all agents have to make the same decision since it considers the observations and decisions made by each agent and gives a general decision that all agents will apply [131]. The selected decision is the one that all members can feel comfortable with, even if they are not unanimous about it. The consensus is usually coupled with probabilistic DM methods such as Bayesian optimization.

- Multi-Attribute Utility (MAU) models or function: A utility (or reward) is attributed to each action by evaluating it with more than one performance metric [132]. Then, the best action with the higher reward is selected (such as with the dominance strategy). MAU can be utilized with probabilistic DM methods.

DM strategy	Advantages	Disadvantages	Example of HRC applications
Dominance	It is a very intuitive strategy that usually gives a solution for interactions with strict dominance [98].	It cannot deal with a situation with more than one optimal choice [98].	Researchers used dominance strategy to assess the human's confidence in a robot in [133].
Maximin	It guarantees that the agents choose the actions with the highest reward in the worst-case scenario [98].	If the current scenario is not the worst-case scenario, the agents' chosen actions may not be the best [98].	The maximin strategy ensured human safety within the robot's motion planning when sharing their workspaces [134].
Minimax	Minimax is a beneficial DM strategy for reducing the risks taken by an agent by promoting an exhaustive evaluation of the search space while considering all the alternatives [125].	Applying the minimax strategy to complex tasks (i.e., with a large tree and a huge branching factor) results in decreased performance and efficiency since evaluating unnecessary nodes or branches slow down the process of finding solutions [125].	The minimax strategy is used to train a generative adversarial network to recognize human facial emotions to help the robot interact with the human agent [135].
Pareto optimality	This strategy ensures that the agent chooses actions that maximize their own reward [98].	If the order in which agents make their decisions changes, the actions they choose will also change since the actions with the highest rewards will also change [98].	It is used in disassembly and remanufacturing tasks [136].
Nash Equilibrium (NE)	The NE strategy ensures that all agents choose the optimal actions, considering that all agents seek to maximize their reward [98].	It does not guarantee the uniqueness of the solution, i.e., most interactions will typically have more than one NE. This strategy may occasionally provide solutions that most players disagree on (e.g., in the prisoner's dilemma game) [98].	A NE strategy ensures human safety in a nearby environment during a pick-and-place task [137].
Stackelberg duopoly model	This strategy helps both agents to make the optimal choice [127].	This strategy is only formulated for the case where two agents interact and cannot deal with a multi-agent interaction case [127].	This strategy is used in a collaborative scenario between a human and a car to predict the driver's behavior in a specific scenario [138] such as the driver's steering behavior in response to a collision avoidance control [139].

ELECTRE	They are powerful to outrank the alternatives based on the performance metrics [128].	They are time-consuming [102]. ELECTRE strategies prevent the direct identification of the alternatives' strengths and weaknesses [117].	ELECTRE strategy was used to perform a preference outranking analysis of the four available HRC configurations in a real automotive industry application to identify the configuration that best exploits the potential of HRC [140].
PROMETHEE	They can identify and consider the alternatives' strengths and weaknesses in the decision [117].	They require the assignment of weight values but do not provide a methodology that assigns them [117].	A robot used a PROMETHEE strategy to target the right components to disassemble during the disassembly process of an end-of-life automobile by a human-robot collaborative team [141].
Lexicographical order	Lexicographic order is one of the most straightforward choice strategies in DM [130].	A preference among the performance metrics should be defined [130].	This strategy is used to classify the primitive features of human multimodal data (e.g., videos, motion data, applied force, etc.) to enable robots to learn how to perform object manipulation actions based on human manipulation data [142].
Consensus	It considers all the data observed by all the agents to make an optimal general decision [143].	The agents cannot make different decisions [143].	The consensus strategy is utilized for making a robot initiate a conversation with the human it collaborates with [144].
Multi-Attribute Utility (MAU) models	MAU can make decisions taking into account multi-performance parameters, agents' preferences, and uncertainty [132].	This strategy is hardly applicable in the case of a unique performance metric to optimize [132]. A huge amount of inputs is essential at each process step to accurately record the agents' preferences [117].	Robots use the MAU strategy to perform motion planning in human proximity to reach a goal position while ensuring human safety by avoiding collisions with them [145].

Table 2.6: Advantages and disadvantages of DM strategies with an example of their applications within HRC.

2.3 Utility or reward functions

The utility is a reward calculated by the utility function (or the reward function) to express the value of an action. Thanks to these utilities, the DM strategy can choose the right actions. One possibility to increase the performance of the collaboration is to act on the reward function by considering performance metrics. This possibility is the most used in the literature. The reward function is designed specifically for a particular problem. It evaluates the action with respect to the desired goal to be achieved. Most contributions in the literature focus on defining this reward function in a new way to optimize collaboration.

Some previous works, known as leader-follower systems [11], considered task accomplishment (and no performance metrics) in their utility functions because they focused on complex task accomplishment. Their decision process is focused on choosing the actions that increase the robot's dexterity to accomplish the task without considering how the collaboration is done (i.e.,

the performance metrics). For example, in [17], a human-robot collaborative team was carrying a table to move it from one room to another. The goal was to ensure mutual adaptation between the agents by having the human also adapt to the robot. Another relevant example is [146], where robots influence humans to change the pre-defined leader-follower agents to rescue more people when a plane or ship crashes into the sea. None of the performance metrics in Table 1.1 is considered in this type of work.

On the other hand, other works deal with maximizing the collaboration performance by promoting mutual adaptation [147, 148] or reconsidering the task allocation [149]. They, then, include performance metrics (see Table 1.1). In these works, they only consider one or two unchangeable performance metrics for the HRC evaluation in their utility function: postural or ergonomic optimization [150, 151], time consumption [152], trajectory optimization [153], cognitive aspects [154], and reduction of the number of human errors [155].

However, they considered that the performance metrics are not changeable without significant changes in their framework. A relevant example is [156] where, by changing the task allocation, the authors make the robot respect the real-time duration of the assembly process while following the necessary order to assemble the parts. In this case, they considered one metric (the time to completion) since respecting the part's assembly order is a constraint to accomplish the task. However, this time metric cannot be replaced by another (e.g., effort or velocity) using this framework.

To solve this problem, we introduce in Chapter 3 a framework that integrates an unrestricted number of changeable performance metrics. The main contribution of this framework is that it considers an unrestricted number of performance metrics that can change from one collaboration to another.

2.4 Conclusion

In this chapter, we focused on introducing the DM process of the robot since the goal of the thesis is to optimize the HRC by improving it. We explained that the DM process comprises three parts: *(i)* a DM method, which represents the relationship between the agents, the actions, the environment, and the task, *(ii)* a DM strategy which is the policy of choosing actions based on the value of their reward calculated by the reward function, and *(iii)* a reward function (or utility function) which calculates a reward for each action. We introduced the most popular DM methods, strategy, and reward functions used for HRC in the state-of-the-art.

In Chapter 3, we present our framework that can improve the HRC while considering different performance metrics. It can deal with the change of the considered metrics because we isolate the impact of the performance metrics in the utility function (i.e., the reward function). Then, there is no need to change the way the task is formalized. These metrics are considered regardless

of human behavior. To this end, we split the utility function into three parts: the first is for optimizing the metrics, the second is for dealing with the task constraints, and the third is for considering the robots' physical abilities and ameliorating their manipulation dexterity.

Part II

Decision-making framework

Chapter 3

General framework for optimizing the human-robot collaboration decision-making process by changing performance metrics

---

Contents

In the previous chapter, we presented the Decision-Making (DM) methods, DM strategies and reward functions (or utility functions) used within Human-Robot Collaboration (HRC). In this chapter, we introduce our new DM framework in the context of HRC. State-of-the-art techniques consider the HRC as an optimization problem in which the utility function (the name used in game theory), also called the reward function in reinforcement learning, is defined to accomplish the task regardless of how well the interaction is performed. When the performance metrics are considered, they cannot be easily changed within the same framework.

In contrast, our DM framework can easily handle the change of the performance metrics from one case scenario to another. Our method treats HRC as a constrained optimization problem where the utility function (or reward function) is split into three main parts:

- A set of rewards that evaluate the collaboration's performance. This is the only part that is modified when changing the performance metrics. It gives control over the way the interaction unfolds, and it also guarantees the adaptation of the robot's actions to the human ones in real-time.
- A set of constraints that define how to accomplish the task.
- A set that regroups the robots' physical abilities.

Accordingly, the utility function (or reward function) can be designed to ameliorate the robot's manipulation dexterity. We start this chapter by introducing the motivation and the advantages of having such a framework. Then, we position our work compared to the state-of-the-art. Finally, we explain our formalization in detail.

3.1 Motivation

Nowadays, HRC is a fast-growing sector in the robotics domain. HRC aims to make everyday human tasks easier. It is based on the exchange of information between humans and robots sharing a common environment to achieve a task as teammates with a common goal [28]. HRC applications can have social and/or physical benefits for humans [11].

As explained in Chapter 2, robots can adapt to humans in different situations thanks to its DM process that is usually modeled in computer science using a DM method with a strategy and a utility function [42]. The DM method models the whole situation (including the environment, the actions, the agents, the task restrictions, etc.). The strategy defines the policy of choosing actions based on the value of their reward. The utility function (i.e., reward function) evaluates each action for each alternative by attributing a reward to it. As shown in Section 2.3, a possibility to improve the HRC performance is to act on the reward function by considering performances metrics (cf. Table 1.1).

In this chapter, we optimize and quantitatively assess the collaboration between robots and humans based on the resulting impact of some changeable performance metrics on human agents. Hence, an optimized collaboration aims to bring a benefit to humans, such as getting the task done faster or reducing the effort of human agents. However, an unoptimized collaboration will bring nothing to humans or, on the contrary, will represent a nuisance, such as slowing them down or overloading them, even if the task is finally accomplished. The main contribution of this chapter is that the proposed framework allows for optimizing the performance, based on some changeable metrics, of the collaboration between one or more humans and one or more

robots. Contrary to previous works, our framework allows us to easily change the performance metrics without changing the whole way the task is formalized since we isolate the impact of the metrics in the utility function.

This contribution's benefit is to increase the performance of the collaboration with the possibility of improving or not the robot's manipulation dexterity. This is important in relevant practical cases, for instance, when using social robots that have great limitations (e.g., slowness in their movements and/or reduced dexterity) [27], and it is not easy or even possible to ameliorate their abilities drastically. Therefore, our work provides an interesting solution to enhance collaboration performance with such limited robots.

Our framework uses the state-of-the-art DM process composed of a DM method, a strategy, and a utility function. We divide the utility function (or reward function) into three main parts: the collaboration performance evaluated by a reward according to one or several performance metrics, the task accomplishment, which is considered as a constraint since we only deal with achievable tasks, and a set that regroups the robot's physical abilities. In the two following sections, we briefly recall how utility functions are used in the literature and mention our contributions with respect to them.

3.2 Utility or reward functions

As mentioned in Chapter 2, the utility is a reward calculated by the utility function (or reward function) to express the value of an action. Thanks to these utilities, the DM strategy can choose the right actions. Some previous works in the literature only considered task accomplishment (and no performance metrics) in their utility functions because their focus was on complex task accomplishment. For example, in [17], a human-robot collaborative team was carrying a table to move it from one room to another. The goal was to ensure mutual adaptation between the agents by having the human also adapt to the robot. In this type of work, none of the performance metrics in Table 1.1 is considered.

More recent works include performance metrics (see Table 1.1). However, they considered that they are not changeable without significant changes in their framework. A relevant example is [156] where, by changing the task allocation, the authors make the robot respect the real-time duration of the assembly process while following the necessary order to assemble the parts. In this case, they considered one metric (the time to completion) since respecting the part's assembly order is a constraint to accomplish the task. However, this time metric cannot be replaced by another (e.g., effort or velocity) using this framework.

3.3 Contributions

Unlike the utility functions used in the state-of-the-art works, we consider a changeable unrestricted number of performance metrics (from Table 1.1) that are usually optimized no matter how the human is behaving. To summarize our contributions, we propose a framework that allows us to:

- easily change the performance metrics from one scenario to another without changing anything in our formalization except the part in the utility function related to the metrics, and
- increase the performance of the collaboration with the possibility of improving or not the robot's manipulation dexterity since we isolate that part in the reward function.

In the following section, we define the problem formalization and present the utility function, which optimizes the performance metrics and aims to accomplish the task as a constraint.

3.4 Formalization

A HRC[1] consists of a global environment $\{\mathbf{E}\}$ and a task T. The environment state E^k at each iteration k (with $k \in [1, k_f]$, where k_f is the final iteration of the task) comprises objects, and a group of n agents (humans and robots), each of them can carry out a finite set of actions. E^k changes according to the actions chosen by the agents. The global environment $\{\mathbf{E}\}$ is the set of changes in the environment state at each iteration.

$$\{\mathbf{E}\} = \{E^1, E^2, ..., E^k, ..., E^{k_f}\} \tag{3.1}$$

Since the possible actions may change at each iteration, we define $\{\mathbf{A}\}$ as the global set of feasible actions for each iteration k: $\{\mathbf{A}\}^k$.

$$\{\mathbf{A}\} = \{\{\mathbf{A}\}^1, \{\mathbf{A}\}^2, ..., \{\mathbf{A}\}^k, ..., \{\mathbf{A}\}^{k_f}\} \tag{3.2}$$

The set $\{\mathbf{A}\}^k$ contains a set of feasible actions for each agent i (with $i \in [1, n]$) at iteration k denoted by $\{\mathbf{A}_i\}^k$.

$$\{\mathbf{A}\}^k = \{\{\mathbf{A}_1\}^k, ..., \{\mathbf{A}_i\}^k, ..., \{\mathbf{A}_n\}^k\} \tag{3.3}$$

$A^k_{i,a}$ is the a^{th} feasible action of agent i where $a \in [1, l]$ and l is the number of feasible actions of the agent i at time k.

$$\{\mathbf{A}_i\}^k = \{A^k_{i,1}, ..., A^k_{i,a}, ..., A^k_{i,l}\} \tag{3.4}$$

[1]We denote functions by lower case letters in bold, sets and subsets between braces with upper case letters in bold, indexes by lower case letters, parameters by upper case letters, and vectors (i.e., profiles) by letters in bold topped by an arrow between parenthesis.

At each iteration, an action profile $(\vec{\mathbf{A}})^k$ groups the actions chosen by each agent i denoted by $A_i^k \subset \{\mathbf{A}_i\}^k$.

$$(\vec{\mathbf{A}})^k = (A_1^k, ..., A_i^k, ..., A_n^k) \tag{3.5}$$

The optimal action profile $(\vec{\mathbf{A}})^k_{opt}$ at iteration k is computed through the DM function $\mathbf{d}_{M,S}$ as presented in (3.6). $\mathbf{d}_{M,S}$ relies on the DM method M, the DM strategy S, and the utility profile $(\vec{\mathbf{U}})^k_A$ that contains all the utilities for all possible actions $\{\mathbf{A}\}^k$ at iteration k. The DM method M takes into account the constraints related to the task, such as the order in which the agents act, i.e., sequentially or simultaneously. The DM strategy S defines the way the agents must choose the actions according to their utilities contained in $(\vec{\mathbf{U}})^k_A$, as presented in Figure 3.1.

$$(\vec{\mathbf{A}})^k_{opt} = \mathbf{d}_{M,S}\left((\vec{\mathbf{U}})^k_A\right) \tag{3.6}$$

The utility profile $(\vec{\mathbf{U}})^k_A$ is computed by the utility function $\mathbf{f}_u$ based on different sets including: *(i)* the set of performance metrics $\{\mathbb{M}\}$ (cf. Table 3.1), *(ii)* the set of constraints $\{\mathbf{G}\}$ to be respected in order to make the task T progress for accomplishing it, *(iii)* $\{\mathbb{A}\}$ the set that regroups the robots physical abilities, *(iv)* $\{\mathbf{R}\}$ the reward of each action in the profile action which is calculated according to the task and the metrics, and *(v)* $\{\boldsymbol{\epsilon}\}$ a set of weighting coefficients (between 0 and 1) used to determine the importance of each metric (e.g., favoring one metric over the others, especially when it is in opposition to others). We get:

$$(\vec{\mathbf{U}})^k_A = \mathbf{f}_u(\{\mathbb{M}\}, \{\mathbf{G}\}, \{\mathbb{A}\}, \{\mathbf{R}\}, \{\boldsymbol{\epsilon}\}) = (U^k_{A_1}, ..., U^k_{A_i}, ..., U^k_{A_n}) \tag{3.7}$$

Let us discuss how one can make changes to the different elements involved in (3.7). To only change the scenario of the collaboration by changing the performance metrics $\{\mathbb{M}\}$ in $\mathbf{f}_u$, we first need to change the value of the metrics $\{\mathbb{M}\}$, and the value of the reward $\{\mathbf{R}\}$ of each action, and afterward recalculate the utilities $(\vec{\mathbf{U}})^k_A$. To only modify the agent's actions, the utilities $(\vec{\mathbf{U}})^k_A$ should be recalculated for the new actions. To change the robots involved in the collaboration, the robots' abilities $\{\mathbb{A}\}$ should be adjusted. To change the task, we will need to modify the constraints $\{\mathbf{G}\}$, which define the task by setting the conditions that allow the agent to only choose among the actions which permit to make the task progress. It will then be necessary to recalculate the utilities $(\vec{\mathbf{U}})^k_A$. We can, of course, combine several modifications (e.g., changing the performance metrics and the task) by making the appropriate adaptations (e.g., first modifying the metrics $\{\mathbb{M}\}$, the rewards $\{\mathbf{R}\}$, and the task constraints $\{\mathbf{G}\}$, and afterward recalculating the utilities $(\vec{\mathbf{U}})^k_A$).

3.5 Performance metrics

As long as a metric can be formulated mathematically or at least can be measured during the execution of the task and expressed as a condition to calculate the task rewards, it can

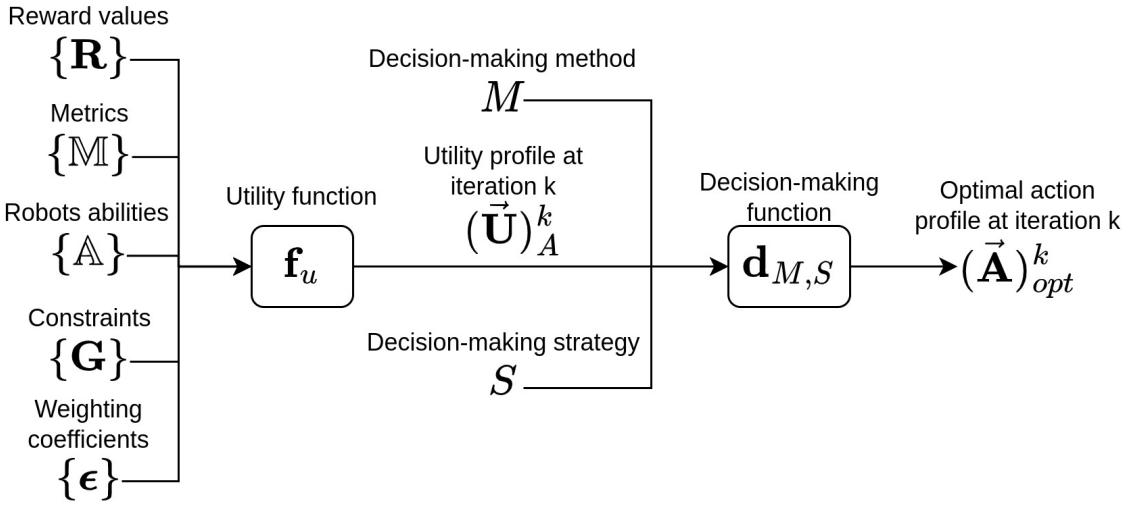


Figure 3.1: Block diagram of our formalization of the DM process used to calculate the optimal action profile $(\vec{\mathbf{A}})_{opt}^k$ at iteration k.

be considered in choosing the actions through the performance metrics $\{\mathbb{M}\}$. Some of the performance metrics that we can consider are presented in Table 3.1. Table 3.1 is a simplified version of Table 1.1 presented in Chapter 1.

Task	Navigation	Perception	Management	Manipulation	Social	Common metrics that can be used for all task types
Performance metrics	Failure rate, accuracy, ergonomy or posture, time to completion, and rapidity	Velocity, accuracy, time to completion, effectiveness, and number of errors	Time delivery, time request, number of human and robot errors, trust, and cognitive load	Positional accuracy and repeatability, velocity, dexterity, time to completion, and effort or force	Persuasiveness, engagement in social characteristics, trust, and compliance	Time to completion, number of human and robot errors, autonomy, cognitive load, and effectiveness

Table 3.1: Some metrics considered for the evaluation of HRC classified based on the task types [10–12].

3.6 Example of a classical HRC application treated by our framework

To illustrate how (3.6) and (3.7) can be settled, let us consider the example of a collaborative team composed of a human and a robot, each holding an edge of a gutter on which there is a ball [9]. Their goal is to position the ball, for instance, in the center of the gutter (cf. Figure 3.2). Note that this is just an example and that all the elements such as action sets, constraint sets, DM method, DM strategy, and performance metrics can be replaced within our framework. Our solution using our formalization for such a task can be as follows:

- Agents: Agent 1 is the human, and agent 2 is the robot. Both agents are making decisions simultaneously.

- Human actions $\{\mathbf{A}_1\}^k$: They are the angles of inclination of the gutter. The actions are continuous. The set of human actions remains the same for all the iterations ($\{\mathbf{A}_1\}^k = \{\mathbf{A}_1\}$).

- Robot actions $\{\mathbf{A}_2\}^k$: They are angles of inclination of the gutter by the robot's end-effector. The DM function will provide the correspondent joint values needed to reach the desired position by the end-effector. The actions are continuous (since it is a continuous control task). The set of robot actions remains the same for all iterations ($\{\mathbf{A}_2\}^k = \{\mathbf{A}_2\}$).

- Environment: The environment contains the human hand position, the robot's end-effector position, and C_b is the position of the center of the ball. The environment changes at each iteration according to the agents' actions.

- Constraints $\{\mathbf{G}\}$: The inclination angles of the gutter should be, for instance, between $[-30°, 30°]$; other values will be penalized. This is because even though both agents (the robot and the human) can incline the gutter more, if they do so, the ball will move too fast, and the task will no longer be achievable.

- Performance metrics $\{\mathbb{M}\}$: Time to completion and human posture. Human posture is measured by ISO standards that define some uncomfortable work postures [157]. These uncomfortable postures (or positions) will lead, for example, to find that when the human inclines the gutter with an angle out of the interval $[-20°, 20°]$, it is getting painful for them.

- Rewards $\{\mathbf{R}\}$: They will be calculated by the following equation: $-\left\|C_b - C_g\right\| * \lambda$. Where C_b is the position of the center of the ball, C_g is the position of the center of the gutter (the desired position), and λ is a fixed gain for a case scenario. λ allows to privileged an action according to the performance metrics ($\{\mathbb{M}\}$) and the constraints ($\{\mathbf{G}\}$).

- Weighting coefficients $\{\boldsymbol{\epsilon}\}$: It is equal to 1 for both performance metrics.

- Robot's abilities $\{\mathbb{A}\}$: They should be equal to the robot's joints limits, limiting the attainable inclination angles (the robot's workspace). The importance of optimizing the robot's motion dexterity is to make the robot able to take full advantage of its physical abilities. Otherwise, it will not be able to perform the action (the desired angle of inclination), even if this action is comprised within its workspace.

- DM method M: It can be, for instance, a reinforcement learning process that is based on trial and error learning. The agent 2 (the robot which learns) in state s^k makes an action $A_{2,a}$ which changes the state to s^{k+1}. The observation the agent got from the environment

E^{k+1} describes the changes that happened by moving from state s^k to s^{k+1}. The reward $R(s^k, A_{2,a})$ evaluates the taken action $A_{2,a}$ (which leads to the new state s^{k+1}) with respect to the desired learning goal. The state s^k is made up of C_b, C_g, the position of the robot's end-effector, and the human hand position. The learning procedure of all reinforcement learning algorithms consists of learning the value that is attributed to the state $V(s^k)$ defined below.

- DM strategy S: It can be, for example, the dominance strategy. Once the $V(s^k)$ are learned for all possible states, the optimal actions can be chosen. Most of the reinforcement learning algorithms are based on the Bellman equation for choosing the optimal actions [158]:

$$V(s^k) = \max_{A_{2,a}}(R(s^k, A_{2,a}) + \gamma V(s^{k+1})) \tag{3.8}$$

γ is the discount factor that determines how much agent 2 cares about rewards in the distant future relative to those in the immediate future. $\max_{A_{2,a}}$ is the strategy S for choosing the action (i.e., dominance strategy).

Figure 3.2: A human-robot collaborative team. Each agent holds an edge of a gutter on which there is a ball [9]. They try, for instance, to position the ball in the center of the gutter.

As we can see from the previous example, the DM method manages the way agents act (simultaneously or sequentially) as well as the different types of actions (continuous or discrete). It is also necessary to ensure that the DM strategy can handle the nature of the actions (discrete or continuous) and how they are chosen (sequentially or simultaneously). As our framework allows us to easily change the DM method and strategy, we just have to select them according to the nature of the actions and how they are chosen. In this example, the robot's manipulation dexterity is improved since we enhance the action made by the robot. This consists of defining how the robot should move its arm to achieve the task while optimizing the performance metrics.

3.7 Conclusion

In the previous chapter, we presented the most used DM methods in the literature, such as probabilistic methods, deep learning, and game theory. We also introduced the most popular DM strategies, such as dominance, minimax, maximin, and Nash Equilibrium (NE). Then, we showed that the reward functions utilized in the state-of-the-art DM frameworks do not easily consider the change of the performance metrics used to optimize the HRC.

In this chapter, we introduced our new DM framework that can handle the change of the performance metrics from one case scenario to another. Our framework solves the difficulty of changing the performance metrics within the same framework without redefining the whole way the task is formalized. It can increase the performance of the collaboration with the possibility of improving or not the robot's manipulation dexterity since we isolate that part in the reward function.

In the next chapter, we apply our framework to make a human-robot collaborative team achieve an "assembly task" (i.e., a game based on a construction kit) non-intuitive (i.e., it is complicated to know which piece to put where). Since we utilized the Nao robot, which has great physical limitations, we are enhancing the HRC without having to increase Nao's manipulation dexterity (i.e., through improving the perception, the trajectory planning, or the low-level control). In the following chapter, the robot's DM process is based on NE and Perfect-Information Extensive Form (PIEF) from game theory. Thus, the robot can deal with collaborative interactions considering different performance metrics, such as optimizing the time to complete the task, considering the probability of human errors, etc. We chose PIEF from game theory as DM method because of its sequential nature, which is suitable for HRC applications and, more specifically, the "assembly task" we want to make the collaborative team able to achieve. We selected NE as DM strategy because it ensures optimality regarding the choice of actions, which is what we seek to guarantee.

The proposed framework in this chapter and the tests conducted in real and simulation in the next chapter to assess our framework when it is applied to the "assembly task" were published in an international journal under the reference: Hani Daniel Zakaria, M., Lengagne, S., Corrales Ramón, J. A., & Mezouar, Y. (2021). General Framework for the Optimization of the Human-Robot Collaboration Decision-Making Process Through the Ability to Change Performance Metrics. *Frontiers in robotics and AI, 8,* 736644. `https://doi.org/10.3389/frobt.2021.736644`.

Chapter 4

Evaluation of our framework which optimizes the human-robot collaboration

---

Contents

In the previous chapter, we introduced our new Decision-Making (DM) framework, which can be used to optimize the Human-Robot Collaboration (HRC) by ameliorating the robot's DM process. This framework can handle the performance metrics change from one case scenario to another. It overcomes the challenge of modifying the performance metrics within the same framework without redefining the whole way the task is formalized.

In this chapter, we introduce the DM method and the DM strategy that we used to make a human-robot collaborative team able to perform an "assembly task". We conduct real and simulated tests to prove the effectiveness of our framework. We test three different utility function case scenarios in which the reward values change according to the chosen performance metrics. In the state-of-the-art case scenario, no metric is optimized. In the real experimental tests, the time to completion metric is optimized. In the simulated tests, we optimize the time to completion by considering the probability of human errors and the time each agent takes to make an action.

4.1 Problem statement

We address the problem of the optimization of the HRC DM process through the ability to change performance metrics. A human-robot collaborative team performs an "assembly task", i.e., a game[1] that involves placing cubes to build a path between two figurines (cf. Figure 4.1). The objective is to increase the collaboration performance based on different performance metrics without having to ameliorate the robot's manipulation dexterity (since Nao's physical abilities are very limited). The difficulty is to be able to change the performance metrics within the same framework without having to redefine the whole way the task is formalized. In the following sections, we will detail the components of our framework that we will use to solve this problem.

[1]Camelot Jr. is a game created by Smart Games: https://www.smartgames.eu/uk/one-player-games/camelot-jr

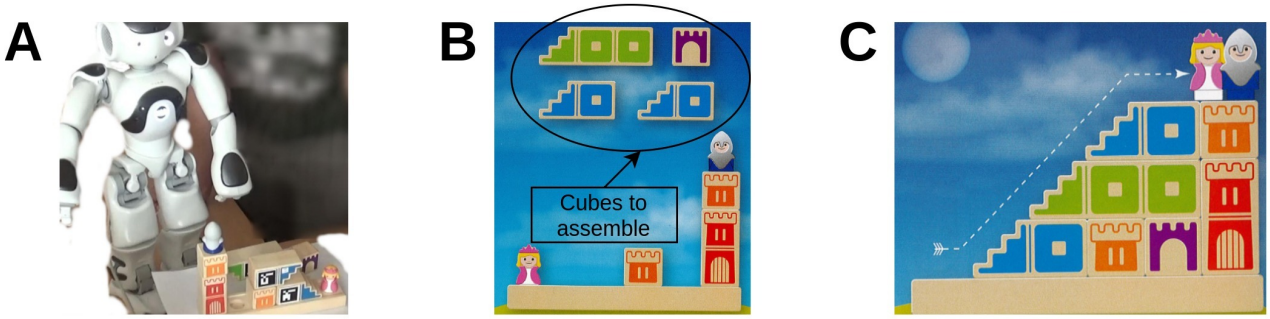


Figure 4.1: Agents solving the Camelot Jr. game. **(A)** Agents play sequentially: the human starts to play, and then it is the robot's turn. **(B)** This puzzle starts with four cubes to assemble. **(C)** The cubes are correctly assembled, and the puzzle is solved (i.e., a path composed by cubes is created between both figurines).

4.2 Robot's DM process

In this section, we will explain the DM method and strategy we will use within our framework. To illustrate our contributions, we define a constant DM method M and strategy S. We assume as DM method the Perfect-Information Extensive Form (PIEF) from the game theory (environment and actions are known) in which the full flow of the game is displayed in the form of a tree. Using Nash Equilibrium (NE) as the strategy of the DM process ensures optimality regarding the choice of actions, which is what we seek to guarantee.

4.2.1 DM method

DM methods are used, as mentioned before, to model the relationship between the task, the agents accomplishing it, their actions, and their impact on the environment. As presented in Chapter 2, probabilistic methods, deep learning, and game theory are considered among the most widespread DM methods. Game theory methods in DM processes have only recently been exploited. They can model most of the tasks performed by a group of agents (players) in collaboration or competition, whether the choice of actions is simultaneous (normal form, also called matrix form modeling [97]) or sequential (extensive form, also known as tree form modeling [98]).

We choose the game theory as a DM method due to its simplicity and effectiveness in modeling most interactions between a group of participants and their reactions to each other's decisions. We specifically use the extensive form due to its sequential nature, which is suitable for HRC applications and, more specifically, the one we want to make the collaborative team able to achieve.

As DM method M in (3.6) we used the PIEF. Using this method, the agent has all the information about the actions and decisions of other agents and the environment. A game (or task or application) in PIEF in game theory is represented mathematically by the tuple $T = (\{\mathbf{N}\}, \{\mathbf{A}\}, \{\mathbf{H}\}, \{\mathbf{Z}\}, \boldsymbol{\chi}, \boldsymbol{\rho}, \boldsymbol{\sigma}, \{\vec{\mathbf{U}}\})$ [98], with:

- T represents the game (i.e., the task) as a tree (graph) structure.
- $\{\mathbf{N}\}$ is a set of n agents.
- $\{\mathbf{A}\}$ is a set of actions of all agents for all iterations.
- $\{\mathbf{H}\}$ is a set of non-terminal choice nodes. A non-terminal choice node represents an agent that chooses the actions to perform.
- $\{\mathbf{Z}\}$ is a set of terminal choice nodes; disjoint from $\{\mathbf{H}\}$. A terminal choice node represents the utility values attributed to the actions A_i^k each agent i chose in an alternative (i.e., a branch of the tree).
- $\boldsymbol{\chi}$: $\{\mathbf{H}\} \mapsto \{\mathbf{A}\}_{@H}$ is the action function, which assigns to each choice node H a set of possible actions $\{\mathbf{A}\}_{@H}$.
- $\boldsymbol{\rho}$: $\{\mathbf{H}\} \mapsto \{\mathbf{N}\}$ is the agent function, which assigns to each non-terminal choice node an agent $i \in \{\mathbf{N}\}$ who chooses an action in that node.
- $\boldsymbol{\sigma}$: $\{\mathbf{H}\} \times \{\mathbf{A}\} \mapsto \{\mathbf{H}\} \cup \{\mathbf{Z}\}$ is the successor function, which maps a choice node and an action to a new choice node or terminal node.
- $\{\vec{\mathbf{U}}\} = \{(\vec{\mathbf{U}})_A^1, ..., (\vec{\mathbf{U}})_A^k, ..., (\vec{\mathbf{U}})_A^{k_f}\}$ is the global utility profile for all iterations.

We apply this structure to represent the task in the following sections. In our case, since the number of nodes is small, $\boldsymbol{\chi}$, $\boldsymbol{\rho}$, and $\boldsymbol{\sigma}$ are straightforward functions (cf. Figure 4.4).

From a high-level perspective, a perfect-information game in extensive form is simply a tree (e.g., Figure 4.4) which consists of:

- Non-terminal nodes (squares): each square represents an agent that will choose actions.
- Arrows: each one represents a possible action (there are as many arrows as available actions $\{\mathbf{A}_i\}^k$ for agent i at iteration k).
- Terminal nodes (ellipses): each ellipse represents the utilities calculated for each action chosen by each agent in an alternative (i.e., a branch of the tree).

Note that this kind of tree is made for all the possible alternatives (considering all the actions an agent might choose), even if some of them will never happen (the agent will never choose some of the available actions). In this way, the tree represents all possible reactions of each

agent to any alternative chosen by the others, even if, in the end, only one of these alternatives will really happen.

4.2.2 DM strategy

As mentioned in the previous chapters, the DM strategy is the policy of choosing actions based on the value of their reward calculated by the utility function (i.e., the reward function). As presented in Chapter 2, the most used strategies for multi-criteria DM coupled with Game theory in HRC are dominance, Pareto optimality, NE, and Stackelberg duopoly model [98].

We choose the NE as a DM strategy because it makes each agent responds to the others in the best possible way. The best response is the best actions an agent can choose, whatever others have done. Applying this strategy leads to having an optimized collaboration, and this is what we seek to guarantee.

As DM strategy S in (3.6) we used NE. The game T can be divided into subgames T^k at each iteration. In game theory [98], we consider a subgame of T (in PIEF game) rooted at node H as the restriction of T to the descendants of H. A subgame perfect Nash equilibrium of T is all action profiles $(\vec{\mathbf{A}})^k$ such that for any subgame T^k of T, the restriction of $(\vec{\mathbf{A}})^k$ to T^k is a NE of T^k.

NE in pure strategy (game theory) at iteration k is reached when each agent i best responds to the others (denoted by $-i$). The Best Response (BR) at k is defined mathematically as:

$$A_i^{*k} \in BR(A_{-i}^k) \text{ iff } \forall A_i^k \in \{\mathbf{A}_i^k\}, U_{A_i^*|A_{-i}}^k \geq U_{A_i|A_{-i}}^k \tag{4.1}$$

Hence, NE will ultimately be expressed as follows: $(\vec{\mathbf{A}})_{opt}^k = (A_1^k, ..., A_i^k, ..., A_n^k)$ is an optimal profile of actions following Nash's equilibrium in pure strategy iff $\forall i, A_i^k \in BR(A_{-i}^k)$.

From a high-level perspective, to ensure that the actions chosen by one agent are following the NE strategy, it is enough to verify that each agent chooses the actions that have the maximum possible utilities. The definition of the utility function depends on the task and the performance metrics we want to optimize in each case scenario. In the next sections, we present all the case scenarios we suppose to solve the task and the correspondent utility functions.

4.3 The task

We chose to solve Camelot Jr. as a task. To successfully complete this task, all the cubes must be positioned correctly to build a path between the two figurines (cf. figure 4.1). We have divided the task completion process into iterations, during which each agent chooses an action sequentially.

4.3.1 Experiments context

We make the collaborative team ($\{\mathbf{N}\}$), composed of a human (h) and the humanoid robot Nao (r), do a task (T) that consists in building puzzles. Nao is much slower than the human ($t_{A_r} > t_{A_h}$) in doing physical tasks (e.g., pick-and-place tasks), and we want to minimize the total task time ($\{\mathbb{M}\}$). This slowness depends on the nature of the robot itself (its motor capacity combined with the use of its integrated camera) and the puzzle's complexity. The puzzle is more complex for the robot as the number of cubes to assemble increases. It is quite different for the human agent; the complexity depends on their "intelligence", which means that the puzzle is easier as the human is "intelligent". By "intelligent", we mean that the human can discover rapidly and without making mistakes where the correct position of each cube is.

The advantage of collaborating with the robot is that it knows the solution to the construction task. Therefore, the robot is always performing well, even if it is slower than the human. The human agent, however, can make mistakes. The human begins to play, and then, it is the robot's turn. The robot will correct the human's move if this move is wrong. The changes in the robot's DM process between the three case scenarios, including all the details we will present in the following sections, are shown in Figure 4.6.

4.3.2 Assumptions

To illustrate the contributions of our framework, we consider the following assumptions:

- The task is always achievable. We solve the task while optimizing the performance metrics through the utility function. The optimization of the metrics does not have an impact on the solvability of the task.

- We limit the number of agents to two: a human (h) and a robot (r). Hence, $\{\mathbf{N}\} = \{h, r\} \implies n = 2$.

- We limit agents to choose only one discrete action per iteration (i.e., $|A_i^k| = 1$) and to maximize only one metric (time to completion) in the real experiment and two metrics (time to completion by considering the probability of human errors) in the simulated experiments.

- The task is performed sequentially through iterations. An iteration k includes the human making an action, then the robot reacting.

- The agent set of actions and the time the agent takes to make an action are invariable by iteration.

4.3.3 The actions

The set of human actions (4.2) and the set of robot actions (4.3) are the same at every iteration, and each one of them consists of three actions:

- $A_{h,g} \equiv A_{r,g}$: perform the good action (i.e., grasp a free cube and release it at the right place).
- $A_{h,w} \equiv A_{r,w}$: wait (i.e., the agent does nothing and passes its turn).
- $A_{h,b}$: perform the bad action (i.e., the human makes an error: grasping a free cube and releasing it at the wrong place).
- $A_{r,c}$: correct a bad action (i.e., the robot removes the cube from the wrong place).

$$\{\mathbf{A}_h\}^k = \{\mathbf{A}_h\} = \{A_{h,g}, A_{h,w}, A_{h,b}\} \tag{4.2}$$

$$\{\mathbf{A}_r\}^k = \{\mathbf{A}_r\} = \{A_{r,g}, A_{r,w}, A_{r,c}\} \tag{4.3}$$

4.3.4 Utility calculation

The following equation is the adaptation of (3.7) to the current task. So, the utility of every available action a for each agent i is calculated as follows:

$$U^k_{A_{i,a}} = U_{A_{i,a}} = \left(\frac{1}{t_{A_{i,a}}} \times G_{A_{i,a}} \times R_{A_{i,a}} \right) t \tag{4.4}$$

with:

- $t_{A_{i,a}}$: the duration of action a of agent i,
- t: the total time for an iteration ($t = \sum_{i=1}^{n} t_{A_{i,a}}$, here having $n = 2$, therefore $t = t_{A_{h,a}} + t_{A_{r,a'}}$),
- $G_{A_{i,a}}$: the constraint that ensures the task progression by penalizing the actions which make the task regress (cf. Table 4.1),
- and $R_{A_{i,a}}$: the reward of action a of agent i.

We did not consider improving the robot's manipulation dexterity $\{\mathbb{A}\}$ within this utility function because, as mentioned previously, Nao's physical abilities are very limited.

4.3.5 Strategy of action's choice

In our formalization, we mentioned that the agents are choosing NE as the DM strategy. But since the behavior (the DM strategy) of each human is different from one to another, we

cannot claim that they will follow the NE for choosing their actions. For the robot, however, we restrict it to choose the actions by using the NE strategy. That is why the robot chooses the action with the highest utility knowing the one chosen by the human. Note that, in our case scenarios, the robot reacts to the human's action since they are doing the task sequentially, and the human starts.

	Action	$G_{A_{i,a}}$	Task progress
Human	$A_{h,g}$	1	Progression
	$A_{h,w}$	0	No progression
	$A_{h,b}$	-1	Regression
Robot	$A_{r,g}$	1 if $A_h \neq A_{h,b}$ -1 otherwise	Progression
	$A_{r,w}$	0	No progression
	$A_{r,c}$	1 if $A_h = A_{h,b}$ -1 otherwise	Progression

Table 4.1: The value of the constraint of the task accomplishment for each action: making the task progress ($G_{A_{i,a}} = 1$), making no progression ($G_{A_{i,a}} = 0$), and making the task regress ($G_{A_{i,a}} = -1$).

4.4 State-of-the-art utility function

In state-of-the-art techniques, there is no optimization of the task. This is equivalent to always consider: $\forall a, i \ R_{A_{i,a}} = 1$ in our approach (in (4.4)). For each iteration (each agent chooses an action with a utility), we can represent the task with the tree structure of Figure 4.4.A. We will refer in the rest of this chapter to this case scenario by using C_1.

In this case, using NE, the robot's reaction to the human action will be as follows: $A_{r,g}$ if the human chose $A_{h,g}$ or $A_{h,w}$, and $A_{r,c}$ if the human chose $A_{h,b}$.

4.5 Real experiments with time metric

We conducted tests[2] with a group of 20 volunteers. The objectives were to prove that the framework is applicable to a real task and to check human adaptation to the robot. Two of those tests are presented as an example in the video available on `https://drive.google.com/drive/folders/1fYcaI7un88hL99Sr5fPtpNQcTLWxiIrL?usp=sharing`.

[2]The experiment protocol was approved by the ethics committee of the Clermont-Auvergne University under the number: IRB00011540-2020-48

4.5.1 Experiment procedure

After explaining the game rules to the participants, we asked them to complete two puzzles to make sure they understood the gameplay. Afterward, we asked each participant to complete three puzzles, chosen randomly among five, by collaborating with the Nao robot. All these puzzles are presented in Appendix A in Part IV.

The participant began the game. Then, it was Nao's turn. It continued until the puzzle was done. At each time, the participant had 20 s (t_{A_h}) to make an action or to decide to skip their turn. Nao takes on average 60 s (t_{A_r}) to do an action. It was skipping its turn when humans were well-doing and correcting them when they made an error. Nao did not move the cubes on its own (for human safety), but it was showing and telling the human which cube should be moved and where by pointing it. Figure 4.2 illustrates, as an example, the steps of solving puzzle two by a participant and Nao.

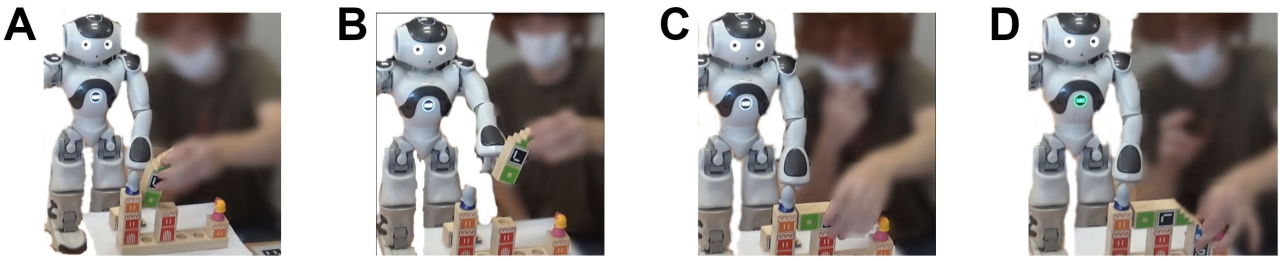


Figure 4.2: Example of the solving steps of puzzle two by a participant and Nao. **(A)** The human puts a cube in a wrong position. **(B)** Nao asks him to remove that cube. **(C)** The human puts a cube in a correct position, then the robot does nothing. **(D)** The human puts another cube in a correct position, and the puzzle is solved.

4.5.2 Implementation of the conducted experiments

In figure 4.3, we present a block diagram of our implementation of the conducted experiments. First, the participant finishes their turn. Then, it is the turn of the robot which will be mainly made up of three parts:

- Perception: Nao's camera captures the markers; then, using the Aruco library [159], we can estimate the pose of the cubes. This allows us to compute the puzzle's state and identify the human action according to the last observation.
- DM process: The robot chooses its action according to the human's one. The utility function calculates a utility for each of the robot's actions. The robot chooses, based on the NE strategy, the action that has the highest utility.
- Robot's action: If the robot passes its turn ($A_{r,w}$), it tells the human. In this case, the robot does not have to make any movement. On the contrary, if the robot corrects the

human's action ($A_{r,c}$) or helps them by indicating where to place a cube ($A_{r,g}$), the robot will have to speak and move its arm to point out the cube to move.

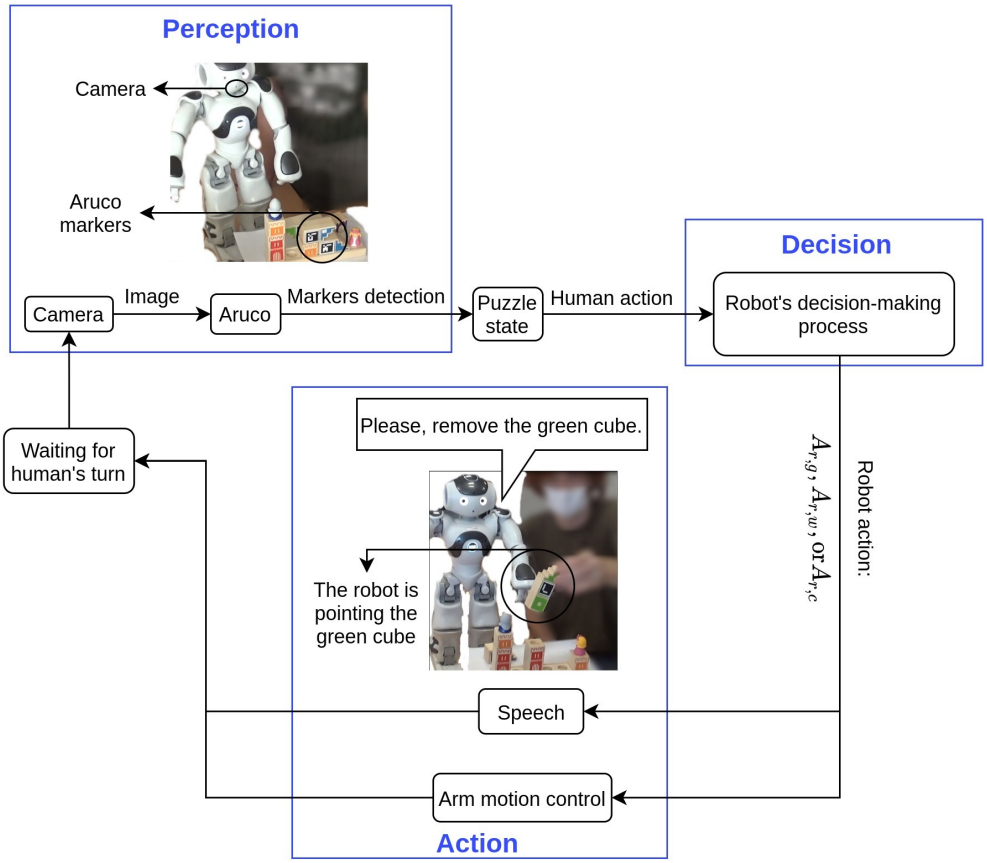


Figure 4.3: Implementation of the conducted experiments using ROS.

4.5.3 Utility function for optimizing the time to completion

The reward values (4.5) in the utility function (4.4) ensure to maximize the time metric by penalising the action taken by the robot (the slower agent, i.e., $R_{A_{i,a}} = -1$) if the human (the faster agent denoted by i') chooses the correct action (denoted by a'). This penalization will prevent the robot from interfering with the human actions if the human makes the right decision:

$$R_{A_{i,a}} = \begin{cases} -1 & \text{if } G_{A_{i,a}} > 0 \text{ and } G_{A_{i',a'}} = 1 \text{ and } t_{A_{i',a'}} < t_{A_{i,a}} \\ 1 & \text{otherwise} \end{cases} \tag{4.5}$$

Thus, for each iteration, we can represent the task with the tree structure of Figure 4.4.B. We will refer in the rest of this chapter to this case scenario using C_2. In this case, using NE, the robot's reaction to the human action will be as follows: $A_{r,w}$ if the human chose $A_{h,g}$, $A_{r,g}$ if the human chose $A_{h,w}$, and $A_{r,c}$ if the human chose $A_{h,b}$.

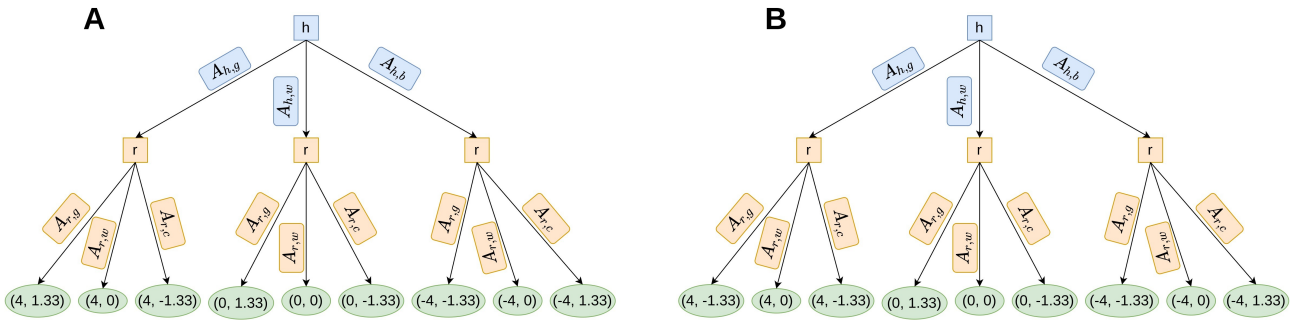


Figure 4.4: Tree representation of the task based on the utility function in C_1 and C_2. Notice that the difference between both figures is the utility value of the action $A_{r,g}$ of the robot (1.33 and -1.33). It is because C_1 (on the contrary to C_2) does not minimize the time, so the robot continues to make an action even if the robot is slower than the well-performing human. **(A)** This tree is obtained by simulating an iteration of the task without optimization (C_1). The utilities (first for human agent and second for robot in green ellipses) are calculated for $t_{A_h} = 20\ s, t_{A_r} = 60\ s$ and $t = 80\ s$. **(B)** This tree is obtained by simulating an iteration of the task optimized by the time metric (C_2). The utilities (first for human agent and second for robot in green ellipses) are calculated for $t_{A_h} = 20\ s, t_{A_r} = 60\ s$ and $t = 80\ s$.

4.5.4 Results

Experiments with humans (presented in Section 4.5.1) were those where the robot used the utility function optimizing the time metric (case 2 (C_2)). It was very difficult to have enough participants to also test the case where the robot does not optimize any metric (the state-of-the-art case (C_1)). The only change in the procedure of the experiments using C_1 will be that even if the human is well-doing, the robot will not pass its turn ($A_{r,w}$) but will perform the good action ($A_{r,g}$). Hence, to compare the achieved results of our technique and the state-of-the-art techniques, we assumed that human actions remain the same in the case C_1 as in the case C_2, and we merely changed the robot reactions.

We chose to keep human actions unchanged between the two cases to ensure that only the switching of the utility function (C_2 to C_1) affects the robot reaction and not the influence of human behavior. Table 4.2 provides an example of a scenario for solving puzzle two with C_2 and C_1 (Figure 4.2). We also calculated in Figure 4.5 the average time and the standard deviation of the measured times among the experiments (C_2) and the deducted times (C_1).

In C_2, we assumed that if the human does the good action once, they will continue to do it each time. We notice from Figure 4.5 that C_1 works better when the human is not "intelligent", i.e., they make lots of errors. That is why, the standard deviation values using C_2 are bigger than those using C_1. This is the case for the last three puzzles where the average time using C_2 is bigger than using C_1. For the first puzzle, however, the average time using C_2 is smaller than using C_1, but the standard deviation values using C_2 are bigger than using C_1. The standard deviation values of this puzzle (using C_1 and C_2) are the biggest ones among all puzzles presented in Figure 4.5. Having big standard deviation values means that this puzzle was harder to solve

		Iteration 1	Iteration 2	Iteration 3	Total time
C_1	Human actions	$A_{h,b}$ (20 s)	$A_{h,g}$ (20 s)		160 s
	Robot reactions	$A_{r,c}$ (60 s)	$A_{r,g}$ (60 s)		
	Iteration time	80 s	80 s		
C_2	Human actions	$A_{h,b}$ (20 s)	$A_{h,g}$ (20 s)	$A_{h,g}$ (20 s)	140 s
	Robot reactions	$A_{r,c}$ (60 s)	$A_{r,w}$ (20 s)		
	Iteration time	80 s	40 s	20 s	

Table 4.2: The adaptation of time calculation from C_2 to C_1 for the resolution of one scenario of puzzle two.

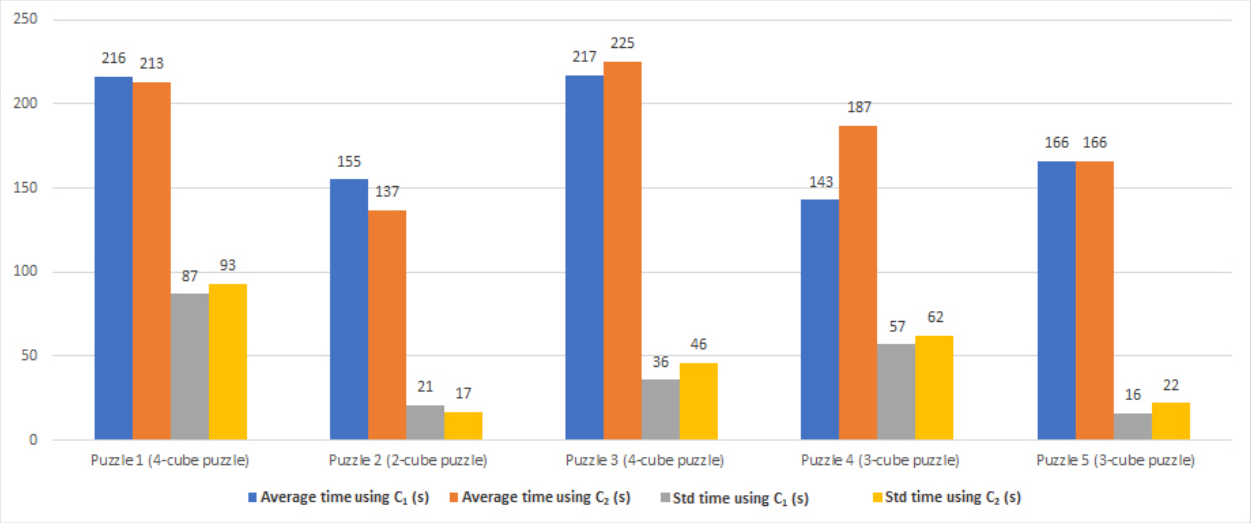


Figure 4.5: The average time and the standard deviation in seconds of the time taken to do the task with the state-of-the-art utility function (C_1) and the utility function used to optimize the time to completion (C_2), which is our contribution.

for some participants and easier for others. That is why the average time and the standard deviation values using C_2 and C_1 do not have the same trend.

On the contrary, C_2 performs better when the human is "intelligent". Therefore, the time taken to accomplish the task depends on human "intelligence", that is related to the probability of human errors and the ratio between the time each agent takes to do an action. Without taking into account these two additional metrics, we cannot optimally ensure to minimize the time to completion when the human makes many mistakes.

In the next case (C_3), we present a third utility function that takes into account the time taken by the agents to make an action and optimizes the time to completion by encouraging the human agent to reduce the number of errors. Each metric has the same weight $\epsilon = 1$ (3.7) since all these metrics are compatible. It means that optimizing one metric depends on optimizing the others.

4.6 Simulated experiments with time and number of human errors metrics

We use case (C_3) to prove that our framework can handle the changes in the performance metrics from one case scenario to another. In this case (C_3), we select between C_1 and C_2, the case that minimizes the total time by considering the probability of human errors and the ratio between the time each agent takes to make an action. The difference between C_1 and C_2 lies in the robot reaction when the human agent makes the good action ($A_{h,g}$). With C_1, the robot makes the good action ($A_{r,g}$), while with C_2, the robot decides to wait ($A_{r,w}$), to not slow down the human. Figure 4.6 presents an algorithmic block diagram showing which case the robot will choose to make an action.

4.6.1 Assumptions on Humans

We did not have enough participants to do real tests, so we chose to do simulated tests. For this, we simulated the human decision process as a probability distribution among the set of feasible actions such that : $P(A_{h,g}) = I_1,\ P(A_{h,w}) = I_2,$ and $P(A_{h,b}) = I_3 = 1 - (I_1 + I_2)$. $I_1,\ I_2,$ and I_3 are variable from one participant to another and $0 < I_1 + I_2 \leq 1$.

4.6.2 Utility function for optimizing the time to completion while considering the probability of human errors

Compares to (4.4), only the reward values ($R_{A_{i,a}}$) change. The reward values of the utility function for C_3 are calculated by the following function:

$$R_{A_{i,a}} = \begin{cases} -1 & \text{if } G_{A_{i,a}} > 0 \text{ and } G_{A_{i',a'}} = 1 \text{ and } t_{C_2} < t_{C_1} \\ 1 & \text{otherwise} \end{cases} \tag{4.6}$$

Where $t_{C_2} < t_{C_1}$ decides which case (1 or 2) is the best to optimize the total time (cf. Figure 4.6) and thus reduce the number of human errors. So, if $t_{C_2} < t_{C_1}$ is true, C_2 will be faster than C_1, and vice versa. t_C is the generic equation for calculating time payoffs t_{C_1} and t_{C_2} (4.7). It considers the probability that the human agent will perform each feasible action ($P(A_{i'})$ = probability distribution of human actions) which we assume as known, and the time that the agents will take to make an action. $t_{A_{i',a'}}$ is the time required for the other agent i' (i.e., human) to make the chosen action a' and $t_{A_{i,a}}$ is the time taken by the agent i (i.e., robot) to react by making the action a. N_c is the number of cubes correctly placed by taking actions a and a'.

C_2 did not work well because it was assuming that if the human does the good action once, they will continue to do it each time. That is why in (4.5) the comparison of the times ($t_{A_{i',a'}} < t_{A_{i,a}}$) was not including the probability of the human actions (including probability of

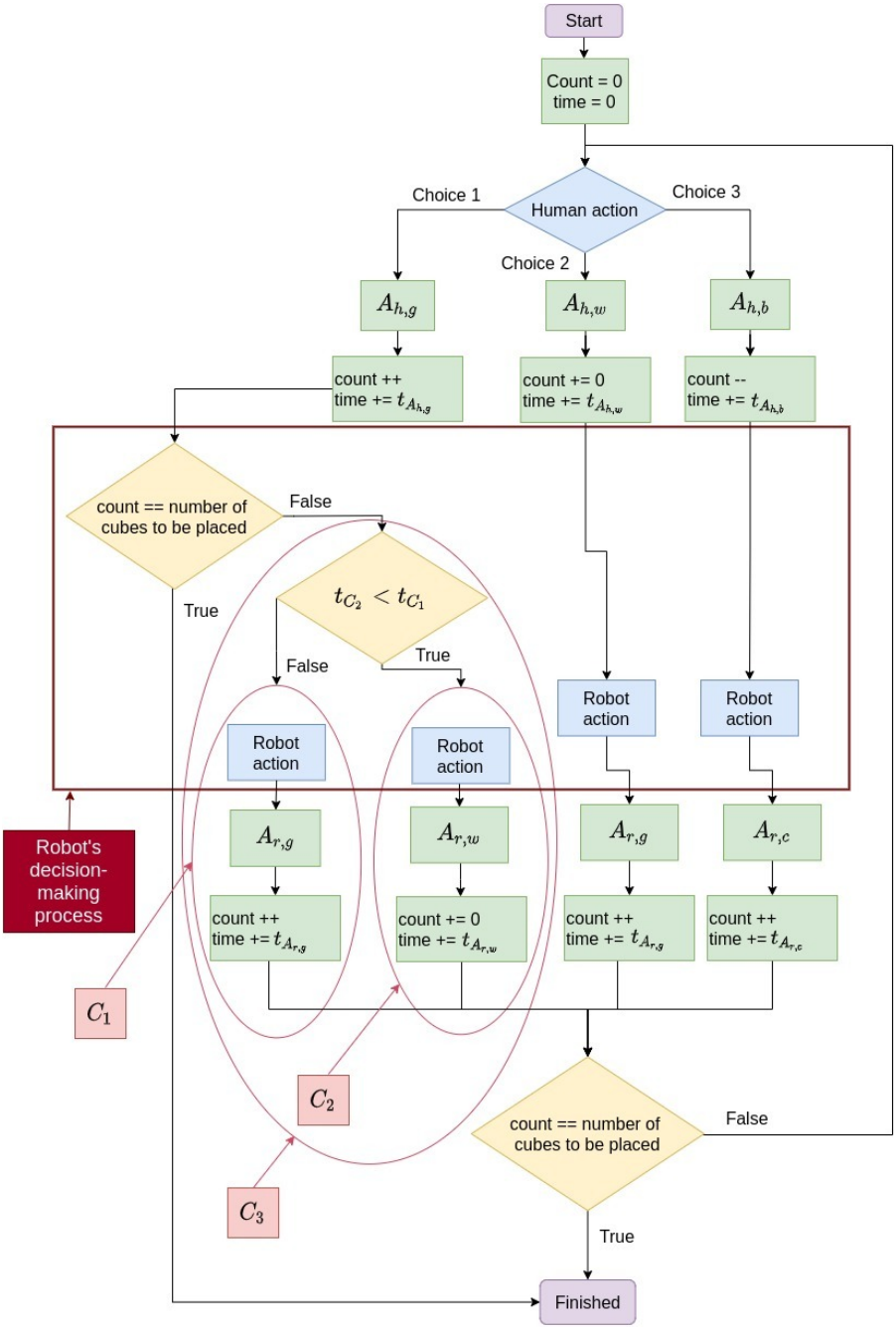


Figure 4.6: C_3 algorithm block diagram.

making errors). In case the human often performs the bad action (e.g., $I_3 \geq 0.6$), the robot is encouraged not to wait but to perform the good action (C_1), despite its slowness. This is done to reduce the number of iterations and thus reduce the number of times the human will make a mistake, as they will have fewer turns to play (i.e., reducing the number of human errors). That is why in C_3, we consider the probability distribution of human actions, including that of doing the bad action (committing errors) while calculating t_c (cf. (4.6)). The robot chooses C_1 if the human will make many errors and C_2 in the opposite case.

$$t_C = \frac{\sum_{a'=1}^{l} P(A_{i',a'})(t_{A_{i',a'}} + t_{A_{i,a}})}{\sum_{a'=1}^{l} P(A_{i',a'})N_c} \tag{4.7}$$

4.6.3 Simulation conditions

A simulated test depends on:

- The values of I_1 and I_2 (we tested for $I_1 = (0 : 0.1 : 1)$ and $I_2 = (0 : 0.1 : 1)$ except for $I_1 = I_2 = 0$).
- The ratio between t_{A_h} and t_{A_r} (we tested for 1/1, 1/1.5, 1/2, 1/3, 1/4, 1/5).
- The number of cubes required to solve the puzzle (we tested for 2, 3, 4, and 5).
- The number of simulations (we conducted 10000 simulations to calculate the average time and the standard deviation).

4.6.4 Simulation Results

We illustrate the efficiency of our utility function C_3 by showing the improvement in time to completion and the reduction of the number of human errors obtained while solving the puzzles.

Time improvement

We validate the efficiency of our utility function C_3 by comparing the resulted average total times with similar cases using C_1 over 10000 simulations[3]. Like real experiments, we assumed that human actions are constant, and we change merely the robot actions. We calculate the time improvement (4.8) by comparing the average total times ($\overline{D}_{C_3}$) calculated using the utility function of C_3 to the average total times ($\overline{D}_{C_1}$) calculated using the state-of-the-art utility function (i.e., C_1).

$$\text{Percentage of time improvement} = \frac{\overline{D}_{C_1} - \overline{D}_{C_3}}{\overline{D}_{C_1}} * 100 \tag{4.8}$$

This is illustrated in Figure 4.7 for a 4-cube puzzle with a ratio $t_{A_h}/t_{A_r} = 1/5$. As it can be observed, the experiment times are improved up to 66.7 %. Another example is given in Figure 4.8 for a 3-cube puzzle with a ratio $t_{A_h}/t_{A_r} = 1/3$. This ratio is the same as the one we had while doing the real experiment with Nao and a human participant. In this case, the experiment times are improved up to 40 %. The percentage of the time improvement depends on how much the human participant is "intelligent".

[3]All the results are presented on `https://github.com/MelodieDANIEL/Optimizing_Human_Robot_Collaboration_Frontiers`

Theoretically, however, this percentage can reach a value close to 100 % for a very small time taken by the human (which leads to a very small $\overline{D}_{C_3}$) and a very big time taken by the robot (which leads to a very big $\overline{D}_{C_1}$). We can note that having the time improvement percentage equal to 0 signifies that we are using C_1; while utilizing C_2 increases the value of the time improvement percentage. It means that in the worst-case scenario, the efficiency of our formalization is as the state-of-the-art peers.

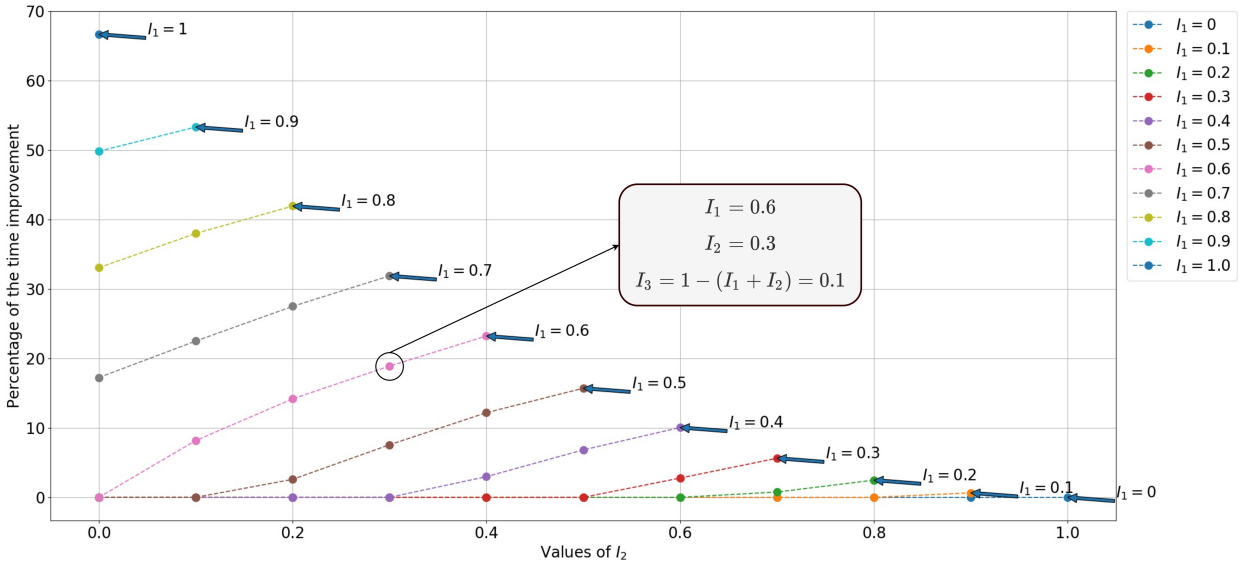


Figure 4.7: Percentage of time improvement between C_3 and C_1 for a 4-cube puzzle. $t_{A_h} = \{15, 0, 15\}$ and $t_{A_r} = \{75, 0, 75\}$, so the ratio $t_{A_h}/t_{A_r} = 1/5$. $P(A_{h,g}) = I_1$, $P(A_{h,w}) = I_2$, and $P(A_{h,b}) = I_3 = 1 - (I_1 + I_2)$. In this figure, each dotted line is equivalent to a specific I_1 value. Each dot corresponds to a I_2 value (read on the x-axis). For each dot knowing I_1 and I_2, we can deduce its I_3 value using $I_3 = 1 - (I_1 + I_2)$. For illustrating, we give I_1, I_2, and I_3 values of the dot marked in the figure.

Reduction of the number of human errors

For reducing the time to completion, we consider the probability of human errors in (4.7). So, we choose between C_1 and C_2, the case which minimizes the time by reducing the number of iterations needed for solving the puzzle. This means choosing the case which reduces the number of human errors as explained in Section 4.6.2. We calculate, in (4.9), the Percentage of Human Errors Reduction (PHER) using the difference between the predicted probability of human errors I_3 and the average (over the 10000 simulations) measured probability of human errors $\frac{\overline{N_{h_e}}}{N_{h_a}}$.

$$\text{PHER} = \begin{cases} \left(\frac{I_3 - \left(\frac{\overline{N_{h_e}}}{N_{h_a}} \right)}{I_3} \right) * 100 & \text{if } I_3 > 0 \\ 0 & \text{if } I_3 = 0 \end{cases} \tag{4.9}$$

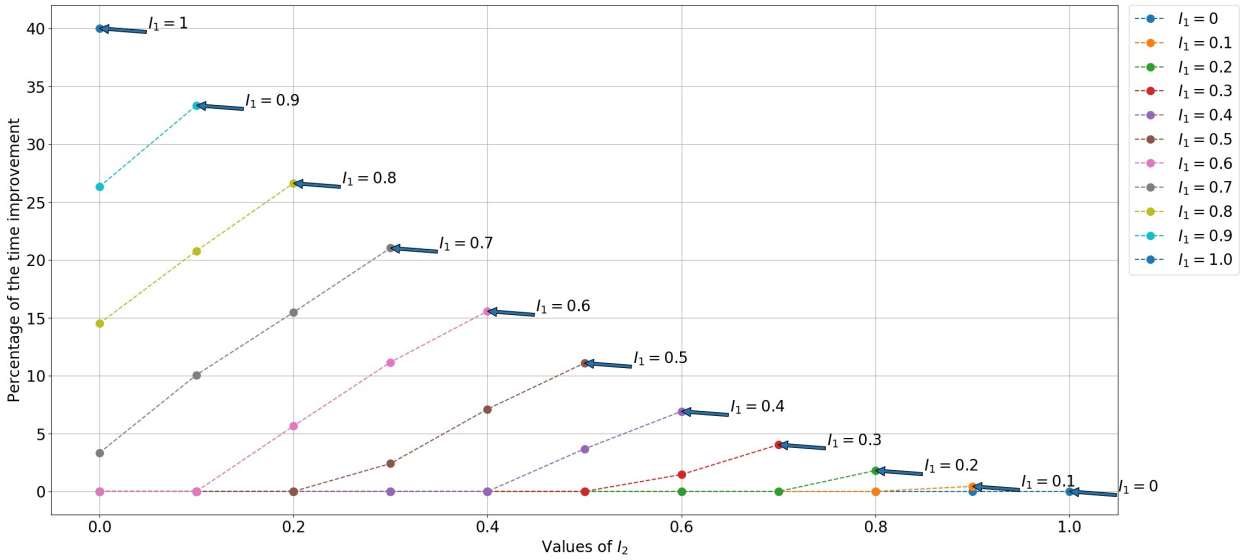


Figure 4.8: Percentage of time improvement between C_3 and C_1 for a 3-cube puzzle. $t_{A_h} = \{15, 0, 15\}$ and $t_{A_r} = \{45, 0, 45\}$, so the ratio $t_{A_h}/t_{A_r} = 1/3$. In this figure, each dotted line is equivalent to a specific I_1 value. Each dot corresponds to a I_2 value (read on the x-axis). For each dot knowing I_1 and I_2, we can deduce its I_3 value using $I_3 = 1 - (I_1 + I_2)$.

Where I_3 is the predicted probability that the human makes a wrong move (makes an error), N_{h_e} the measured number of human errors, and N_{h_a} the measured total number of human actions. So, $\frac{N_{h_e}}{N_{h_a}}$ will be the measured probability that the human makes an error after one simulation. The reduction of the number of human errors is as big as $\overline{\frac{N_{h_e}}{N_{h_a}}}$ is small.

The reduction percentage of the number of human errors increases with the reduction of t_{A_r} (the time the robot takes to make an action) and the reduction of the number of cubes that should be assembled to solve a puzzle. In other words, the human will have fewer turns to play and so fewer chances to make mistakes. The best result we got is presented in Figure 4.9 (for a 2-cube puzzle with $t_{A_h} = t_{A_r}$): the reduction percentage of the number of human errors is up to 50.6 %. Another example is given in Figure 4.10 for a 3-cube puzzle with a ratio $t_{A_h}/t_{A_r} = 1/3$. This ratio is the same as the one we had while doing the real experiment with Nao and a human participant. In this case, the percentage of human errors reduction is up to 27.9 %. The result can be better in case that the robot is faster than the human in performing an action ($t_{A_r} < t_{A_h}$). Note that, when the I_3 is equal to 0 %, the percentage of human errors reduction is also equal to 0 %. It means that the human never makes errors, so there is nothing that needs to be improved.

Appendix B introduces Table B.1, which contains the numerical values of the percentage of time improvement and the reduction of the number of human errors for all the figures presented in this section. In the next section, we will precise the computation and execution time of the tests conducted in real and simulation.

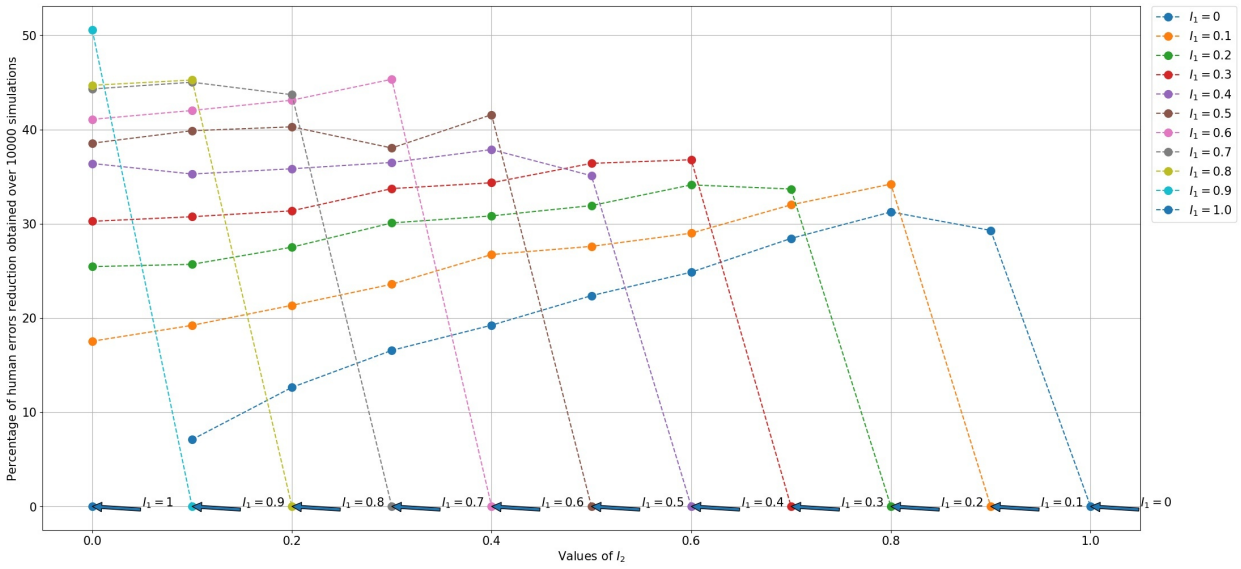


Figure 4.9: Percentage of human errors reduction between the predicted probability of human errors and the measured one for a 2-cube puzzle. $t_{A_h} = \{15, 0, 15\}$ and $t_{A_r} = \{15, 0, 15\}$, so the ratio $t_{A_h}/t_{A_r} = 1/1$. $P(A_{h,g}) = I_1$, $P(A_{h,w}) = I_2$, and $P(A_{h,b}) = I_3 = 1 - (I_1 + I_2)$. In this figure, each dotted line is equivalent to a specific I_1 value. Each dot corresponds to a I_2 value (read on the x-axis). For each dot knowing I_1 and I_2, we can deduce its I_3 value using $I_3 = 1 - (I_1 + I_2)$.

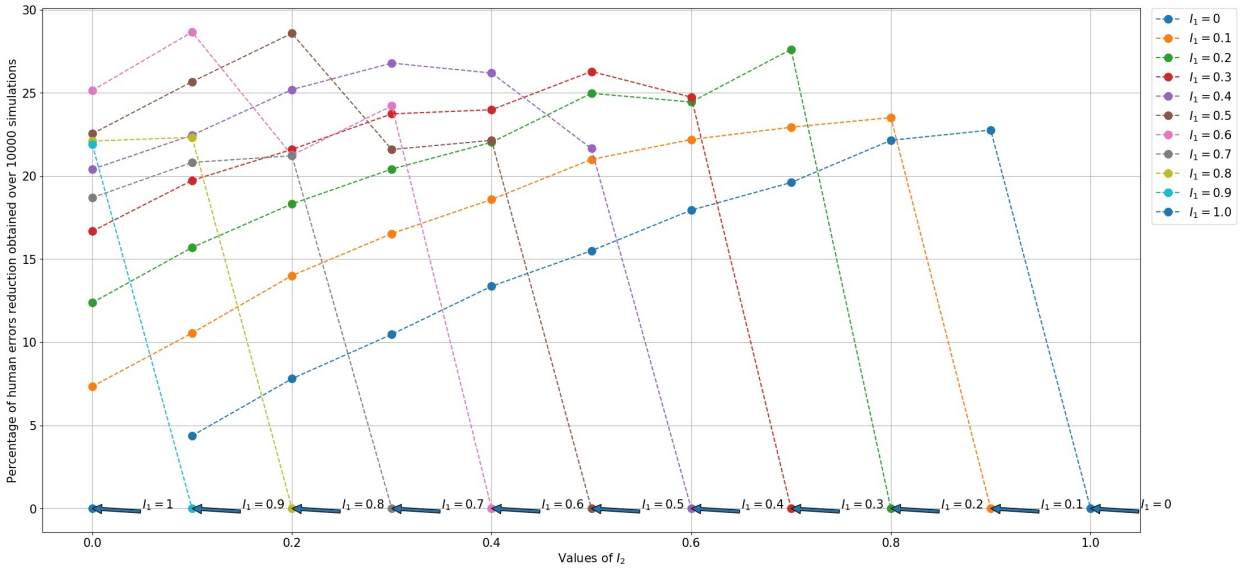


Figure 4.10: Percentage of human errors reduction between the predicted probability of human errors and the measured one for a 3-cube puzzle. $t_{A_h} = \{15, 0, 15\}$ and $t_{A_r} = \{45, 0, 45\}$, so the ratio $t_{A_h}/t_{A_r} = 1/3$. In this figure, each dotted line is equivalent to a specific I_1 value. Each dot corresponds to a I_2 value (read on the x-axis). For each dot knowing I_1 and I_2, we can deduce its I_3 value using $I_3 = 1 - (I_1 + I_2)$.

	Step	Time in seconds
Real tests	Computation time of the robot's DM process (by applying our formalization)	The robot takes an average of 0.5 s to choose the action to perform after knowing the state of the puzzle through the perception part.
	Time taken by the robot for the perception of the puzzle state	The robot takes between 20 s and 30 s depending on how well the cubes are placed and how many cubes are left to assemble.
	Time taken by the robot for doing a physical movement	The robot takes on average 15 s for doing a physical movement.
	Waiting time for the robot when it gives an indication to the human	The robot waits between 5 s and 15 s each time it gives an indication to the human, depending on its complexity (for example, to ask the human to remove a cube, the robot waits for 5 s, and to ask the human to take a certain cube and place it in a certain position, the robot waits for 15 s).
	Global time taken by the robot to perform an action	It is between 20 s and 60 s, depending on the complexity of the movement (the number of cubes left to assemble at this iteration) and if the robot gives indications to the human. We considered that it was 60 s.
	Global time taken by the human to perform an action	The human takes between 1s and 30 s, depending on the complexity of the movement (if they know what to do or not). We considered that it was 20 s.
Tests in simulation	Time required for all probability distributions of possible human actions without printing the figures (such as Figures 4.8 and 4.10)	The Python code takes between 80 s and 100 s on a Dell laptop with an Intel Core i7 CPU and 32 GB RAM.

Table 4.3: Computation and execution times of the experiments in real and in simulation.

4.7 Computation and execution time of the tests

Table 4.3 presents all the computation and execution times of the experiments in real and in simulation. As we can notice, the average computation time of our DM framework is 0.5 s. This computation time is suitable for the targeted real tasks on which we want to apply this framework.

4.8 Conclusion

In the previous chapter, we proposed a new formalization of the robot's DM process to perform the task and accomplish it more efficiently. In this chapter, we assessed through the experiments that our formalization could be applied to feasible tasks and optimize the HRC in terms of all defined metrics. We also proved through the experiments that we can change the three studied case scenarios by changing the performance metrics in the utility function (i.e., reward function) without changing the entire framework.

Validating this, experiments are carried out by simulating the task of solving the construction puzzle. It shows that using our proposed utility function instead of the state-of-the-art utility

function improves the experiment time up to 66.7 %, hence improving the HRC without extending the robot's abilities. Theoretically, this improvement can reach a value close to 100 %. We also got a percentage of human errors reduction up to 50.6 % by considering the predicted probability that the human makes errors for optimizing the time to completion.

As we can notice from the conducted experiments, even though we were able to optimize the collaboration, the improvement was restrained because the abilities of collaborative robots are very limited, especially Nao. More precisely, while achieving the "assembly task", the robot was much slower than the human, which disabled the optimization of the performance metrics. Nao was unable to do the pick and place task adequately. To maximize the optimization of collaborative performance, we need to increase the robot's manipulation dexterity.

In the next part, we will focus on improving the robot's manipulation dexterity while it performs a soft object manipulation task thanks to a Deep Reinforcement Learning (DRL) approach. We chose to perform a deformable object manipulation task because it is a more challenging problem to solve than rigid object manipulation. In the next chapter, we will present all the background of the manipulation of soft objects (especially the deformable linear objects) and the DRL approaches we use.

Part III

From decision to action

Chapter 5

Background on the manipulation of Soft Linear Objects, Deep Reinforcement Learning, and Deep Deterministic Policy Gradient

Contents

The physical capabilities of collaborative robots are restrained. Thus, from the results we got in the previous chapter, the collaboration improvement was limited even though we could optimize it utilizing our framework developed in the last part. The pick and place application was too complex regarding the physical abilities of the robot. Indeed, Nao was significantly slower than the human agent, which prevented the optimization of the performance metrics from being maximized.

In this part, we want to focus on enhancing the robot's manipulation dexterity to further improve the collaboration performance. We aim to make the human-robot collaborative team able to perform more complex tasks requiring the robot to be at least as fast as the human agent to be able to adapt to them. To achieve this, we will also use a Decision-Making (DM) approach. We will utilize an approach based on Deep Learning (DL) to avoid the problems faced by *classical* control methods to represent complex tasks since they do not easily consider changes in the task components [160, 161]. Indeed, to change any of the many components of complex tasks using the *classical* control methods, the entire architecture (i.e., the equations) must be redesigned. Adding extra agents or changing the steps needed to accomplish the task can be considered examples of changes requiring fundamental alterations to the architecture design.

An excellent example of a complex application is to make the collaborative team perform a co-manipulation task in which a human and a robot deform a Soft Object (SO) to make it reach a final desired shape (cf. Figure 5.1). On the one hand, some works tend to perform this physical interaction using *classical* control methods. On the other hand, some others use decision-making methods and, more precisely, Deep Reinforcement Learning (DRL) based approaches for accomplishing the task. However, they are all considering rigid objects. Table 5.1 presents some of these works and our targeted contribution.

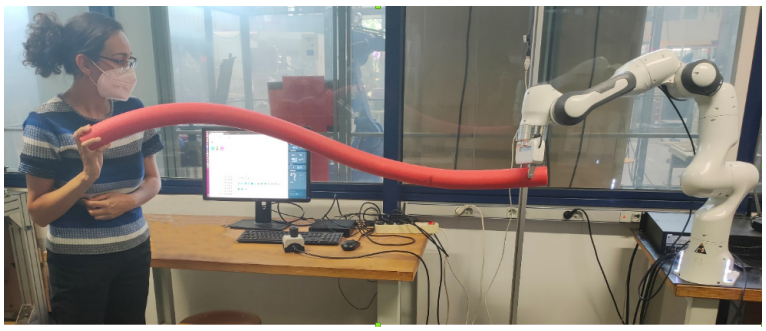


Figure 5.1: A human-robot collaborative team achieves a co-manipulation task in which they are deforming a SO to make it reach a final desired shape.

Since making the collaborative team able to deform a SO is a highly challenging task, we divided it into several less complex intermediate steps:

1. We use the DRL approach to make the robot able to deform the SO in Chapter 6. This

	Deep Reinforcement Learning (DRL)	*Classical* control methods
Human-Robot Collaboration (HRC)	In [74], the robot and the human are lifting a heavy rigid object.	In [151], a human is guiding a robot to draw straight lines.
Soft Object (SO)	In [90], the robot folds a towel up, folds a face towel diagonally, and drapes a piece of cloth over a hanger.	In [162], a shape servoing approach is introduced for soft thin-shell objects based on As-Rigid-As-Possible (ARAP) deformation model.
HRC and SO	**Contribution: to the best of our knowledge, there is no work combining DRL with HRC and SO**	In [163], using classical control, they aim to make the robot and the human capable of transporting a piece of cloth.

Table 5.1: State-of-the-art of manipulating a SO by a human and a robot and our targeted contribution.

approach can generalize the initial and final desired deformation of the SO more easily than the existing ones in the literature.

2. We applied this approach to a dual-arm robotic agent that deforms the SO in Chapter 7. The results prove that the approach is as generalizable as when it was tested on a single-arm robot.

3. In the future, we want to apply this approach to the human-robot collaborative team that deforms the SO. We plan to achieve this by having a separate agent for each robotic arm during the training phase and replacing one of the robotic arms with a human agent in the testing phase in real. Training two individual robotic agents can be performed thanks to multi-agent deep reinforcement learning approaches [164–166]. We will discuss this concept further in Chapter 8.

4. As future work, we also want to maximize the optimization of this collaboration by considering some performance metrics such as the time to achieve the deformation, the accuracy of the reached deformation, the robot's dexterity, and the robot's velocity to adapt to the human agent.

The final goal of this part is to adapt the robot's behavior to the human during their collaboration to deform a SO. The robot's DM method will be DRL, and its DM strategy will be dominance (i.e., the robot chooses the actions with the maximal reward). In this chapter, we introduce the background of manipulating deformable linear objects and DRL techniques we use.

5.1 Motivation

The manipulation of deformable objects is currently a relevant topic in robotics research [160, 167]. In particular, robotic manipulation of deformable linear objects is essential for many industrial and daily life applications such as, cable harnessing [168, 169], USB wire soldering [170],

vegetable plant manipulation [171, 172], surgical suturing [173], knot tying [174], and assisted living [175]. The manipulation of a Deformable Linear Object (DLO) by single or dual robotic arms is challenging because of its high number of degrees of freedom [176]. One possible perspective on this problem is to study model-based manipulation planning, as done in [177–182]. The major flaw of the model-based approaches is the inaccuracy of the generated model due to the complexity of the deformation modeling [176]. Another problem with those techniques is that the generated model is only valid for a specific object or a category of objects, such as DLOs. Moreover, the model has an intractable high dimensionality since it specifies all SO deformations configurations [175].

In this work, we are instead interested in the online control of single or dual robotic arms to deform a DLO in a desired way in conditions of high uncertainty and with no knowledge of the object's mechanical deformation model (i.e., we want to develop a model-free method). The works that addressed a similar scenario considered mostly 2D workspaces [168, 183–187], while control in 3D is significantly more challenging due to the higher complexity of object modeling and perception [87]. Some works addressed control in 3D for small deformations [169, 188]. Overall, while *classical* methods have achieved important progress in this field, the existing challenges motivate us to explore a solution based on DRL [161, 176].

The robotics community has increasingly adopted the usage of DRL algorithms to control robots [79]. Most of these works involve working with rigid bodies with no or negligible deformations [90, 160]. However, SO manipulation has many crucial applications, especially in household robotic assistance, medicine, and industry [90, 189]. In industrial automation, DRL has already been identified as interesting in tasks with high modeling uncertainty and the need for high dexterity. For instance, [190, 191] used reinforcement learning for industrial assembly, albeit without having to deal with deformable objects, as we do here.

In the literature, the works based on DRL for manipulating deformable objects are, on the one hand, only formulated for simple tasks [160] such as hanging a cloth [90, 189], folding a cloth [192,193], cable insertion [194], rope knotting [195], or moving a rope [161,175,193,196,197]. On the other hand, most of the SOs used are 2D [160]: the mesh used to model the object is a 2D mesh, i.e., formed by 2D polygons such as triangles. Promoting progress in this regard, SoftGym [198] presented a set of benchmarks for manipulating SOs (including 3D objects) using OpenAI Gym [199] and Python interface.

The main drawback of the existing techniques mentioned previously, whether used in simulation [161, 189] or in real experiments [90], is that they are not easily generalizable [79, 160, 161, 175, 193, 195, 197]. Their agent is trained to perform a manipulation from constant initial to constant target deformations, and it is not trained to deal with different configurations. As an example in DLO manipulation, in [161], the authors control the object shape from some initial states to some desired deformations that are not changeable.

This chapter describes an approach for the robotic control of the shape of DLOs. We use a combination of state-of-the-art DRL algorithms and techniques. We use learning parallelization to make our approach generalizable, i.e., we execute multiple agents in parallel on various environment instances. We focus our study on DLOs. to sum up, the contributions of our control approach are:

1. Its generalizability, i.e., we train the agent (a single-arm or a dual-arm robot) only once (using a specific SO), and it can deform the SO starting from a different initial position and end up with a different desired shape. Moreover, the agent can make the SO reach an untrained position, i.e., we train the agent on a small workspace and test it on a bigger one.

2. It can achieve a more complex task than the ones performed in the literature. As shown in Figure 6.1 and Figure 7.1, the agent deforms a foam bar by making some selected mesh nodes reach the corresponding desired positions in 3D space, potentially involving complex torsion motions. This is made possible by modeling the object with a 3D tetrahedral mesh and via our DRL system design.

We train and evaluate our approach in simulation. The results are shown in Chapter 6 for a single-arm robot and in Chapter 7 for a dual-arm robot. Our evaluation is carried out in diverse conditions, and it validates the capability of the proposed approach. We made some preliminary tests and proposed a solution in Chapter 6 to transfer what the agent learned in simulation to a real single-arm robotic agent. In the next section, we introduce some background on SO manipulation using DRL approaches.

5.2 Background on SO manipulation using DRL

This section gives background on the problem of SO manipulation using DRL. We will focus on discussing aspects that are particularly important for our application.

5.2.1 Representing deformable object shape

The most widely used technique is to represent the SO shape through images [189, 200] instead of modeling it since it is challenging to have a precise model [160]. In [90], a neural network detects the SO shape thanks to supervised learning. The disadvantage of using images is that the computational cost increases, and it is hard to learn afterward (i.e., via DRL) because the resulting state space is large [189]. In [161], a method based on geometry calculations is proposed to represent the object shape. Another method is based on selecting some mesh nodes in the object model describing the deformation and using their positions as state space inputs [160, 189]. We preferred to use the latter technique because it is easier to set up, and it keeps the size of the space-state relatively small, which facilitates the training.

5.2.2 Techniques to deal with the manipulation complexity

The most common technique in the state-of-the-art is to combine Imitation Learning (IL) with reinforcement learning [160]. IL is used to reduce the complexity of the manipulation by using demonstrations given by an expert. Another method that we mentioned previously is to have a detailed perception of the object's shape through images [189]. The drawback of both methods is that they have a high computational cost and a large state space [161, 189]. We prefer to use only a DRL algorithm and select a few mesh nodes that describe the deformation of the object as input to the state space. This way, our state space is small, which makes learning easier.

5.2.3 Physics-based simulator

Usually, the training of the agent is done in simulation, using a physics-based simulation engine [79]. This is because reinforcement learning requires many trial-and-error episodes (i.e., interactions with the environment) to learn the correct policy, which can hardly be done using a real robot to avoid damaging it. OpenAI Gym [199] defines an architecture with the main components needed to train the agent, such as reinitializing the environment, making an action, getting an observation of the state of the environment, and computing the reward. The environment created on the simulator has to have such components. The most popular simulators for deformable object manipulation in the robotics community are MuJoCo [201] and Bullet [202]. We prefer to use PyBullet, the Python interface of Bullet, because it is powerful and open-source. We utilized PyBullet also because most deep learning libraries are developed in Python.

In the following section, we concretely formulate the components used within our DRL control approach.

5.3 Components of our DRL control approach

In this section, we present standard components of DRL techniques that we use. We focus on explaining how we incorporate them together. These elements include the RL procedure, the Bellman equation, the Deep Deterministic Policy Gradient (DDPG) algorithm, and the reward function. In this chapter, we simplified the mathematical notation since we concentrated our work on one agent in the first instance, i.e., a robot with single or dual arms. Since in a first time, we do not try to optimize the performance metrics, we do not take into consideration the sets $\{\mathbb{M}\}$, $\{\mathbf{R}\}$, $\{\boldsymbol{\epsilon}\}$ used to calculate the reward function (or utility function) f_u in Chapter 3 (cf. Figure 3.1). We consider the DRL as our DM method. Our goal is to make the agent choose the actions that increase its manipulation dexterity within the robot's physical abilities $\{\mathbb{A}\}$ (i.e., the robot's articular limitations). The set $\{\mathbb{A}\}$ is settled in the simulator (i.e., Pybullet),

which leads to only having to consider the task constraint $\{\mathbf{G}\}$ within the reward function f_u. Then, the utility profile is reduced to a simple value that will evaluate the actions with respect to $\{\mathbf{G}\}$. From now on, to respect the DRL nomenclature, we will call this value a reward.

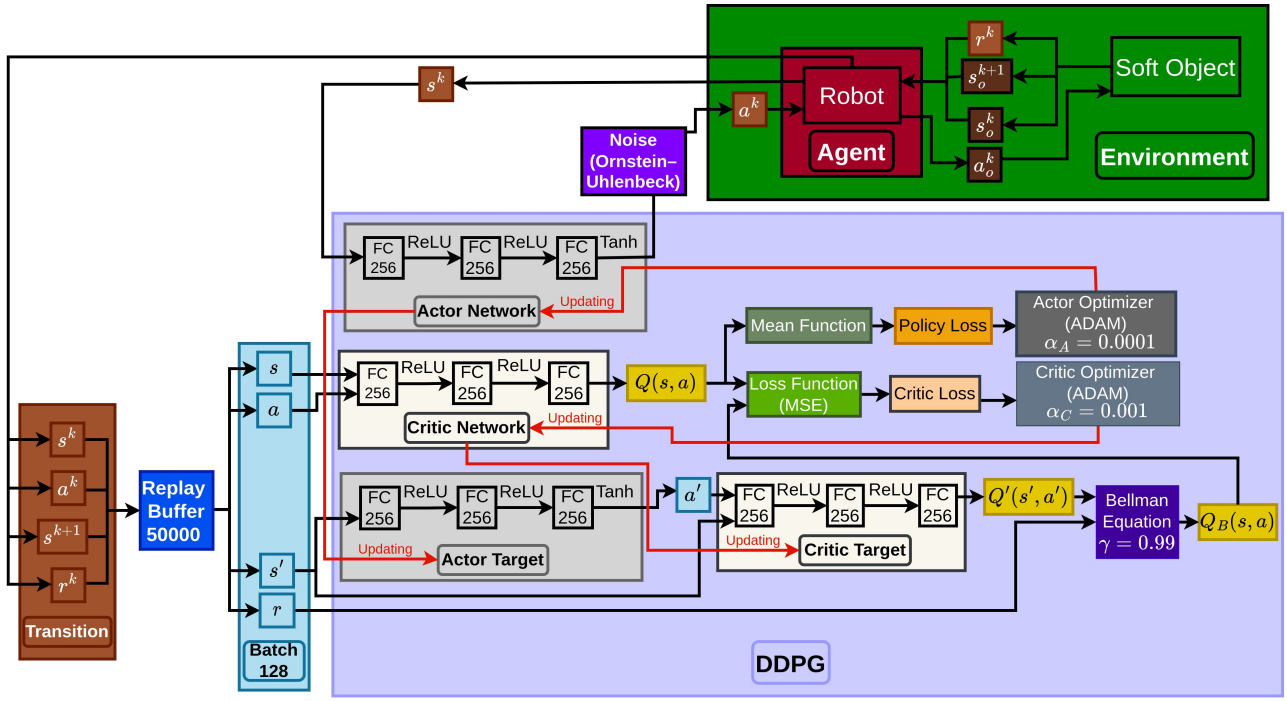


Figure 5.2: Overview of our proposed approach for controlling the deformation of a SO via the DDPG algorithm. The structure of the full DRL system and relevant parameters are displayed.

5.3.1 RL procedure

We consider a classical trial-and-error RL procedure consisting of an agent (e.g., the robot) interacting with the environment (e.g., the SO) based on the policy (or DM strategy) to maximize rewards on discrete timesteps [203]. In each transition (or iteration) k, the agent starts from the state s^k, and takes an action a^k, which changes the state to a next state s^{k+1} [22]. The state s^k and the action a^k are included in the continuous state space $\mathcal{S}$, and the continuous action space $\mathcal{A}$, respectively, i.e., $s^k \in \mathcal{S}$ and $a^k \in \mathcal{A}$.

The observation the agent got from the environment describes the changes that happened by moving from state s^k to s^{k+1}. The reward r^k evaluates the action taken a^k according to the task goal. The agent's goal is to learn the optimal policy $\pi^* : \mathcal{S} \longrightarrow \mathcal{A}$ throughout the different transitions. A transition k is made of an action a^k, a state s^k, a next state s^{k+1}, and a reward r^k.

5.3.2 Off-policy vs on-policy learning

Off-Policy learning algorithms evaluate and improve a policy different from the policy used for action selection [204]. We decided to use an off-policy (as opposed to an on-policy) algorithm

for several reasons. First, when one uses an off-policy algorithm while parallelizing the learning, it speeds up the convergence, i.e., learning can be faster [85, 86]. Parallelizing the learning means executing multiple agents in parallel on various environment instances. We will discuss this concept further in Section 5.3.6. Second, off-policy algorithms are more suitable for an environment where the agent does not have to explore much [204]. This is generally the case for robotic arms applications since the robot's workspace limits the environment exploration. These problems are more frequently solved in the literature using off-policy algorithms such as in [90, 161, 194, 196].

5.3.3 Bellman equation used for critic networks learning

In this section, we introduce the Bellman equation used within the Actor-Critic methods [203] for critic networks learning (i.e., Q-learning), a very popular off-policy learning technique [204]. The Bellman equation [158] is used to calculate a Q-value $Q_B^k(s^k, a^k)$ that evaluates the action a^k chosen in a current state s^k. The Bellman equation (5.1) considers the discount factor ($\gamma \in [0.9, 1]$) and the next Q-value $Q^{k+1}(s^{k+1}, a^{k+1})$ to calculate $Q_B^k(s^k, a^k)$. The discount factor controls how much the DRL learning is considering future rewards. The next Q-value $Q^{k+1}(s^{k+1}, a^{k+1})$ is calculated for choosing the next action a^{k+1} in the next state s^{k+1}.

$$Q_B^k(s^k, a^k) = r^k + \gamma \times Q^{k+1}(s^{k+1}, a^{k+1}). \tag{5.1}$$

5.3.4 Deep Deterministic Policy Gradient (DDPG)

The DDPG is a DRL algorithm based on Actor-Critic methods used for dealing with continuous action spaces [203]. It learns a Q-function and a policy by utilizing off-policy data and the Bellman equation [189]. The actor network (policy network) has as input the state s^k and gives as output the optimal action a^k. The critic network (Q-function network) evaluates the optimality of the action a^k chosen at state s^k by attributing it the Q-value $Q^k(s^k, a^k)$ at transition k. Figure 5.2 presents an overview of the approach established to make the agent deform a SO using the DDPG algorithm. Next, we detail the modules in the algorithm.

Pre-training procedure

The agent applies the action a^k selected by the actor network within the state s^k to the environment in order to store the inputs of the environment (a^k selected in s^k) and its outputs (s^{k+1} and r^k) that constitute the transition k in the replay buffer (cf. Figure 5.2). The training of the actor and critic networks can only begin once the replay buffer contains enough transitions to extract a batch. A batch is composed of elements (i.e., actions a, states s, next states s', etc.) coming from several non-sequential transitions. These transitions are selected randomly.

Training procedure

Making the agent learn from previous memories, i.e., using batches, accelerates learning and breaks undesirable temporal correlations [205]. The training of the critic network consists of reducing the error between the Q-values calculated using the Bellman equation $Q_B(s,a)$ (cf. (5.1)) and the Q-values estimated by the critic network $Q(s,a)$ (cf. Figure 5.2). The Q-values number equals the batch size N_b, i.e., the number of selected transitions to train the agent. The Mean Square Error (MSE) optimization technique is used to reduce that error, i.e., we use the following Critic loss:

$$\text{Critic loss} = \text{MSE}(Q_B(s,a), Q(s,a)). \tag{5.2}$$

The weights of the critic network are updated based on the critic loss. The ADAM optimizer [206] is used to calculate the gradient descent. The Q-values given by the critic network $Q(s,a)$ are used to evaluate the actions a chosen by the actor at states s. Then, the actor's training is based on the Q-value given by the critic network, i.e., the actor loss is equal to the Q-value. Since the agent's training is made from a batch, one obtains as many Q-values $Q(s,a)$ (cf. Figure 5.2) as there are transitions in the batch. The policy loss is calculated by taking an average of the $Q(s,a)$ [203]:

$$\text{Policy loss} = -\overline{Q(s,a)} = -\frac{\sum_{k=1}^{N_b} Q^k(s^k,a^k)}{N_b}. \tag{5.3}$$

The weights of the actor network are updated based on the policy loss.

Target networks

Using a target network is a technique to stabilize learning. A target network is a copy of the main network's weights held constant to act as a stable target for learning for a fixed number of timesteps [203]. We use Polyak averaging to update the target networks (also called soft updating) once per the main network's update [207]:

$$W_{A_T} = \tau W_A + (1-\tau) W_{A_T} \tag{5.4}$$

$$W_{C_T} = \tau W_C + (1-\tau) W_{C_T}, \tag{5.5}$$

where the used terms are:

- W_{A_T}: the weights of the actor target network.
- W_A: the weights of the actor network.
- W_{C_T}: the weights of the critic target network.
- W_C: the weights of the critic network.

- τ: the Polyak factor.

We choose to utilize the DDPG algorithm as our DRL algorithm because it is suitable for continuous action spaces. It has fewer parameters to set than other actor-critic DRL algorithms. It is a powerful tool to generalize the training, combined with parallel learning.

5.3.5 Reward function

The reward function is the key element that allows us to control and optimize the agent policy (or DM strategy) of choosing actions [22]. More details about choosing the suitable reward function are given in [204, 208]. The most straightforward dense reward function for accomplishing our task ($\{\mathbf{G}\}$) is to use a Euclidean distance-based calculation [161]. Therefore, in Chapter 6, our reward r^k is calculated as the average Euclidean distance between the current positions of the selected mesh nodes and their desired positions. To have more accurate results, in Chapter 7, our reward r^k is calculated as the maximum Euclidean distance between the current positions of the selected mesh nodes and their desired positions.

5.3.6 Learning parallelization

The actor-critic DRL algorithm A3C [209] proposes to asynchronously execute multiple agents in parallel on various instances of the environment. That parallelism decorrelates the agent learning data since, at any transition, the parallel agents will be experiencing a variety of different states. Combining batch extraction and learning parallelization for off-policy algorithms ensures that the training data are decorrelated and can be collected faster [85, 86]. Thus, combining both techniques improves the overall learning time while achieving a better result from the generalization point of view. That is why we train the agent using DDPG on a single multi-core CPU, as in [209].

5.4 Conclusion

In this part, we focused on enhancing the robot's manipulation dexterity. We aimed to make the human-robot collaborative team able to perform complex tasks requiring the robot to be at least as fast as the human agent to be able to adapt to them. An excellent example of a complex application is to make the collaborative team perform a co-manipulation task in which a human and a robot deform a SO to make it reach a final desired shape. The final goal of this part is to adapt the robot's behavior to the human during their collaboration to deform a SO. The robot's DM method is DRL, and its DM strategy is dominance (i.e., the robot chooses the actions with the maximal reward). Since making the collaborative team able to deform a SO is a highly challenging task, we divided it into several less complex intermediate steps: making a single robotic arm deform the SO, making dual robotic arms deform the SO, replacing one

of the robotic arms by a human agent and performing co-manipulation of the SO in real, and optimizing the collaboration performance thanks to some performance metrics.

This chapter proposed a robotic control approach for manipulating SOs. A DRL approach is used to make the shape of a deformable object reach a set of desired points by controlling a robotic agent (a single-arm or a dual-arm robot) which manipulates it. Our approach is more easily generalizable than existing ones: it can work directly with different initial and desired final shapes without the need for relearning. We achieved this by using learning parallelization, i.e., executing multiple agents in parallel on various environment instances. We focused our study on deformable linear objects. These objects are interesting in industrial and agricultural domains, yet their manipulation with robots, especially in 3D workspaces, remains challenging.

In the next chapter, we simulate the entire environment, i.e., the SO and the agent (a single or a dual arms robot), for the training and the testing using PyBullet and OpenAI Gym. We use a combination of state-of-the-art DRL techniques, the main ingredient being a training approach for the learning agent based on DDPG. Our simulation results presented for a single-arm robot support the usefulness and enhanced generality of the proposed approach. We also do some preliminary tests and propose a solution in Chapter 6 to transfer what the agent learned in simulation to a real single-arm robotic agent.

This chapter and the results presented in Chapter 6 were published at an international conference under the reference: Hani Daniel Zakaria, M., Aranda, M., Lequièvre, L., Lengagne, S., Corrales Ramón, J. A., & Mezouar, Y. (2022). Robotic Control of the Deformation of Soft Linear Objects Using Deep Reinforcement Learning. In *2022 18th IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE.

Chapter 6

Robotic Control of the Deformation of Soft Linear Objects Using Deep Reinforcement Learning

Contents

The previous chapter introduced a robotic control approach for manipulating Soft Objects (SOs) based on a Deep Reinforcement Learning (DRL) approach that is more generalizable than the existing ones because we parallelized the learning of the agent. In this chapter, we settle all the parameters of the Deep Deterministic Policy Gradient (DDPG) algorithm as well

as the simulation parameters to make a single-arm robot able to deform a soft linear object. We also present simulation results that prove that our control approach is more generalizable than existing approaches since our agent can directly deform the SO from an initial shape to a desired final shape different from those used during training without needing to relearn. At the end of this chapter, we make some preliminary tests and propose a solution to transfer what the agent learned in simulation to a real single-arm robotic agent.

6.1 Problem statement

We address the problem of controlling the deformation of a Deformable Linear Object (DLO) using a robot arm that manipulates it. For simplicity, the robot grasps one end of the object to not have to deal with the gripper positioning problem [210], and the other end is fixed to the ground. The object is represented by a mesh, and we describe its deformation by a set of selected mesh nodes. The objective is to control the arm so that the positions of the selected nodes are driven to prescribed values. The difficulty of this indirect control problem lies in the fact that the dynamical model of the system to be controlled is complex and uncertain. In the previous chapter, we proposed a generalizable architecture to solve this problem based on DRL. The problem setup is illustrated in Figure 6.1.

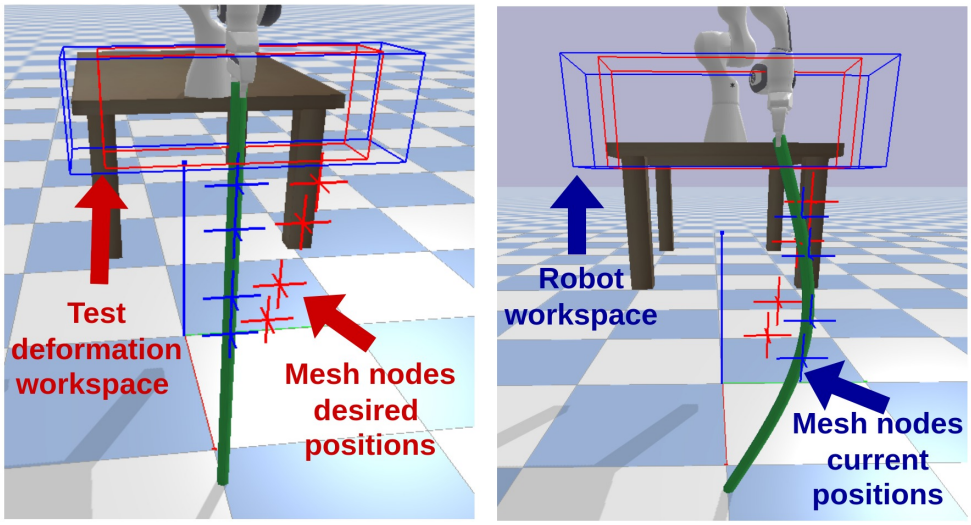


Figure 6.1: The setup we consider, including an illustration of some elements of our methodology. The robot deforms the soft linear object (green) by making the selected mesh nodes (i.e., the blue points) reach the desired corresponding positions (i.e., the red points). The points are marked as crosses. The robot tip position has to remain within the deformation workspace (i.e., the red box) for performing the desired deformation. The deformation workspace used in testing is bigger than the training workspace. The blue box delimits the robot's workspace, i.e., the robot's gripper tip cannot reach a position out of that box due to the robot's articular limits.

We train and evaluate our approach in simulation. Our evaluation is carried out in diverse conditions, and it validates the capability of the proposed approach. The following section

presents all the parameters of our approach, whose values must be set according to the single-arm application.

6.2 Implementation

After having introduced all the necessary DRL components in Section 5.3, we describe in this section how we apply them to address the specific problem scenario considered (Section 6.1). We provide the implementation details including all assigned values for the DDPG and simulation parameters.

6.2.1 Approach overview

Before starting the learning phase, we create a deformation space box within which the robot gripper tip moves to deform the object, and we record the positions of the selected nodes P_d in a database. The reason for using a deformation space box is to record several deformations within a limited space that is reachable by the gripper tip. We have created different databases, each based on a box of different size: the training one, which is smaller, and the testing one, which is larger. All the details about the databases are mentioned in Section 6.2.3. The robot's objective is to manipulate the object so that the current positions of the selected nodes P_c reach the desired positions P_d within a tolerance threshold.

Figure 5.2 gives an overview of our architecture. The action a^k given by the DDPG to the agent (i.e., the robot) is the Cartesian velocity of the gripper tip $a^k \in \mathcal{A} = (V_x, V_y, V_z) \implies a^k \in \mathbb{R}^3$. The action a^k is continuous since each element of the velocity ($V_x, V_y,$ or V_z) can have any value within the interval $[-1, 1]$. Then, the action a^k is integrated according to the timestep (which is equal to 0.06 s) to calculate the new gripper tip position (X_n, Y_n, Z_n). The classical position-based controller available in Bullet moves the arm from its current position (X_c, Y_c, Z_c) to the new one (X_n, Y_n, Z_n). Since the DLO is grasped by the gripper tip, moving the gripper tip will move the mesh nodes a_o^k to some new positions P_c.

The state $s^k \in \mathcal{S}$ is made up of the gripper tip current state $s_g^k \in \mathbb{R}^6$ and the current object shape $s_o^k \in \mathbb{R}^{6m}$ (cf. (6.1)) with m the number of selected mesh nodes. s_{g_t} includes the gripper tip position (X_c, Y_c, Z_c) and velocity (V_x, V_y, V_z). s_o^k is composed of the positions of the selected mesh nodes $P_c \in \mathbb{R}^{3m}$, and their desired positions $P_d \in \mathbb{R}^{3m}$.

$$s^k = (s_g^k, s_o^k) \in \mathcal{S} = (X_c, Y_c, Z_c, V_x, V_y, V_z, P_c, P_d). \tag{6.1}$$

We calculate the reward r^k as the average Euclidean distance D^k between the current positions of the selected mesh nodes P_c and their desired positions P_d (cf. (6.2)). Using subindex j to

denote the position of a single mesh node, we have:

$$r^k = f_u(\{\mathbf{G}\}) = -\overline{D^k(P_c, P_d)} = -\frac{\sum_{j=1}^{m} D^k(P_{c_j}, P_{d_j})}{m}. \tag{6.2}$$

6.2.2 DDPG parameters

The actor, actor target, critic, and critic target Deep Neural Networks (DNNs) have the same architecture: 3 Fully Connected (FC) hidden layers, each of which comprises 256 neurons. We use the Rectified Linear Unit (ReLU) as an activation function. We apply the Tanh function on the actor outputs a^k to ensure that the gripper tip velocities remain in the interval $[-1, 1]$. We add noise to the action a^k using Ornstein-Uhlenbeck noise [203] for the exploration. We initialize the DNNs of the actor and critic with random values as in [203]. The actor target and critic target DNNs weights copy those of the actor and critic DNNs. The ADAM optimizer is used for gradient updates with learning rates of $\alpha_A = 0.0001$ for the actor and $\alpha_C = 0.001$ for the critic. A batch of 128 transitions is randomly sampled from the replay buffer, containing 50000 transitions. We use a constant discount factor $\gamma = 0.99$ and a constant Polyak factor $\tau = 0.01$.

Since we use parallel learning, in each episode (or epoch), 32 agents are trying to achieve a different deformation during 300 transitions. This means that each agent makes 300 actions and passes through 300 different transitions to try to achieve 32 different goals (each agent has a different goal). Each action will have a reward r^k, and each agent will have a global reward equal to the sum of the action reward over the 300 transitions. This leads to having different gradients that are synchronized among the 32 agents, i.e., there will be one final gradient equal to the sum of all the 32 gradients. Then 32 agents networks are updated based on that final gradient so that all these networks keep having the same updated weights. We train the 32 agents during 63 episodes, which is equivalent to $32 * 63 = 2016$ episodes if we use a single agent and do not parallelize the training. The training lasts from 1000 to 1 million episodes in the literature [161, 189]. For training 32 agents, we used 32 CPU cores and the Python library MPI [211]. All the conducted training lasted less than three and a half hours.

6.2.3 Simulation parameters

We use PyBullet as the simulator's physics engine to train our agent. The simulator's physics engine uses the FEM method [212] to simulate the SO, which is cylindrical with a diameter of 5 cm and a height of 1.55 m. The model of our SO is built up from a 3D tetrahedral mesh. That model comprises 200 nodes, 392 tetrahedrons, 789 links, and 396 faces. The SO has the following mechanical parameters: the Young's modulus is equal to 2.5 MPa, the Poisson coefficient is equal to 0.3, the mass is equal to 0.2 Kg, the damping ratio is equal to 0.01, and the friction coefficient is equal to 0.5. These values are selected among the intervals given in [161, 213, 214]

for polyurethane materials which are the most used in the industry to make foam bars. The simulation timestep is equal to 0.003 s. We choose the timestep to integrate the action a^k as equal to $20 * 0.003 = 0.06$ s, i.e., sufficiently larger than the simulation timestep.

We created three databases, each based on a box of different 3D size. The gripper tip moves inside that box to deform the object, and we recorded those deformations to use them as desired positions P_d in the training and the testing phase. Figure 6.2 shows the small, the medium, and the large boxes. The small box size equals 0.15 m on the x-axis, 0.5 m on the y-axis, and 0.25 m on the z-axis. The medium box size is equal to: 0.2 m on the x-axis, 0.6 m on the y-axis, and 0.25 m on the z-axis. The large box size equals 0.2 m on the x-axis, 0.8 m on the y-axis, and 0.3 m on the z-axis. The small box generates the small database, which contains 930 deformations. The medium box is used to generate the medium database, which includes 930 deformations. The large box generates the large database, which has 2651 deformations. We decided to sample the same number of deformations in the small and medium databases to ensure having more new deformations in the medium database. Indeed, since the difference in the 3D size of both small and medium boxes is not big, if we sample many deformations from the medium box, the probability of having the same deformation as using the small box is high. For our first experiment, the small database is used for the training, and both the small and the large databases are used for the testing. For our second experiment, the medium database is used for the training, and both the medium and the large databases are used for the testing.

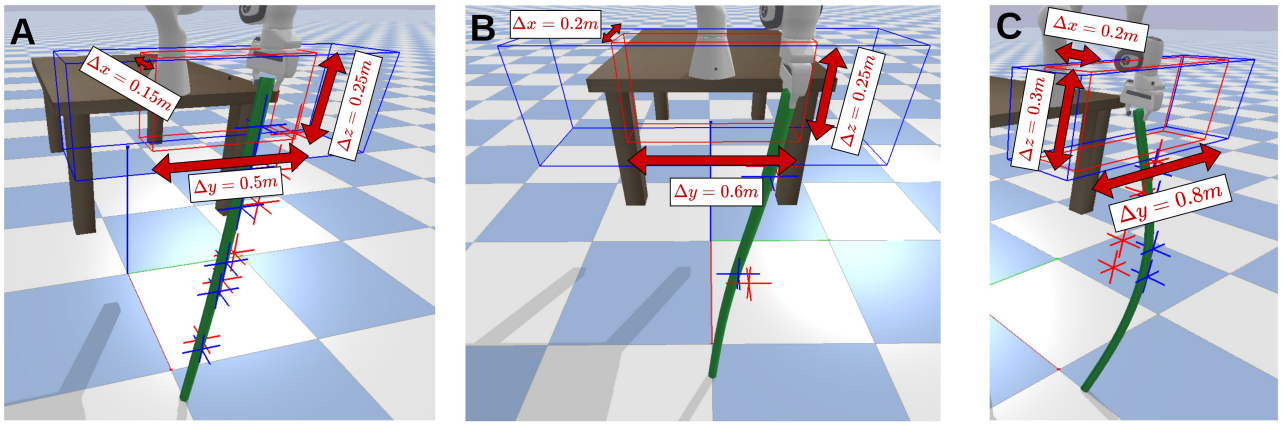


Figure 6.2: 3D boxes of different sizes used for database generation. **(A)** The small box (in red) used to generate the small database. **(B)** The medium box (in red) used to generate the medium database. **(C)** The large box (in red) used to generate the large database.

6.3 Experiments in simulation

This section presents our experimental results in simulation for both conducted experiments. The entire code is available on `https://github.com/MelodieDANIEL/robotic_control_of_DLO_using_DRL`.

6.3.1 First experiment

For the first experiment, we trained the agent using the small database to extract the desired deformations, i.e., the desired positions P_d of the mesh nodes. We have done three trainings to control: two mesh nodes, four mesh nodes, and six mesh nodes. We used an average distance error threshold of 0.05 m. As mentioned in the previous section, the training was parallelized: 32 agents were trained each for 63 episodes, leading to having 2016 episodes in total. During the training, the environment was reinitialized to the initial configuration (the robot and the object returned to their initial position) after each episode. Figure 6.3 shows the average reward obtained by the 32 agents in each episode when controlling two mesh nodes, four mesh nodes, and six mesh nodes. As we can notice from Figure 6.3, there is no need to smooth the learning curves, as in the literature [161, 189]. This is thanks to the stability of the learning due to its parallelization.

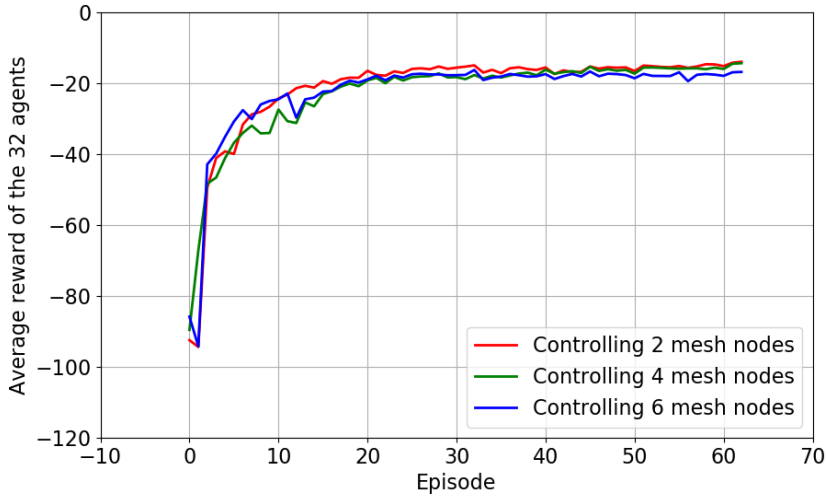


Figure 6.3: Average reward obtained in the first experiment by the 32 agents in each episode when controlling two mesh nodes, four mesh nodes, and six mesh nodes.

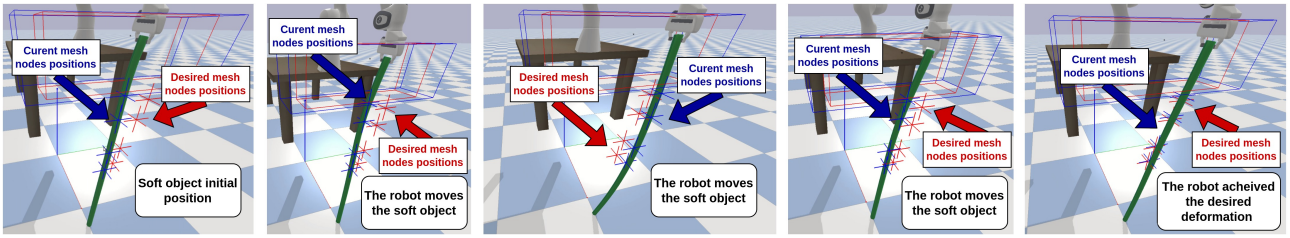


Figure 6.4: Example of the robot deforming the SO: four mesh nodes reach their desired positions with an average distance error threshold of 0.05 m.

For the testing phase, all the results are calculated for 1000 testing episodes with 30 steps, i.e., the robot can take a maximum of 30 actions to achieve the deformation. We test the three trainings for an average distance error threshold of 0.05 m and 0.03 m. We evaluate them using

	Mesh nodes number	Database	Testing error threshold (m)	Done (%)	Mean average distance error $\pm\sigma$ (m)	Best (m)
With reinitialization	2	Small	0.05	99.5	0.03010 ± 0.00591	0.01156
			0.03	80.2	0.02987 ± 0.00993	0.01156
		Large	0.05	73.2	0.05886 ± 0.02730	0.02724
			0.03	21.1	0.06866 ± 0.03214	0.01862
	4	Small	0.05	99.8	0.04251 ± 0.00631	0.01012
			0.03	88.7	0.02816 ± 0.00758	0.01012
		Large	0.05	97.2	0.04386 ± 0.00840	0.02130
			0.03	46.8	0.04650 ± 0.02306	0.01309
	6	Small	0.05	99.1	0.04473 ± 0.00685	0.01009
			0.03	73.4	0.02796 ± 0.01183	0.01009
		Large	0.05	79.0	0.05144 ± 0.01506	0.02779
			0.03	20.0	0.06415 ± 0.02563	0.01548
Without reinitialization	2	Small	0.05	87.9	0.04530 ± 0.01142	0.01579
			0.03	47.5	0.04107 ± 0.01808	0.01236
		Large	0.05	44.9	0.07411 ± 0.04454	0.01642
			0.03	13.9	0.08038 ± 0.04536	0.02438
	4	Small	0.05	93.7	0.04486 ± 0.01215	0.00569
			0.03	45.2	0.05015 ± 0.02732	0.01399
		Large	0.05	72.8	0.05365 ± 0.02201	0.01506
			0.03	21.0	0.05873 ± 0.02551	0.01343
	6	Small	0.05	36.8	0.08669 ± 0.03817	0.02106
			0.03	15.7	0.07478 ± 0.03371	0.01826
		Large	0.05	64.8	0.05914 ± 0.02447	0.02771
			0.03	15.2	0.06244 ± 0.02655	0.01537

Table 6.1: Results of all the conducted tests for the trainings using the small database.

the small and the large databases to extract the desired deformations. We assess them finally with and without reinitializing the environment. All these results are presented in Table 6.1. In Table 6.1, the column "done" indicates the percentage of the agent's success in achieving the desired deformations. The percentage is calculated on the 1000 episodes with 30 steps. The "Best" column reveals the minimum distance error obtained within the 1000 episodes.

Figure 6.4 presents an example of the robot deforming a SO to reach a new deformation on which the robot was not trained. Other deformations are presented in the video available on `https://youtu.be/MbFCS59ZZ_4`. As we can notice from Table 6.1, in the case that we reinitialize the environment and use the same database and distance error threshold as in training, the agent achieves in the worst-case scenario 99.1 % deformations. If we only change the distance error threshold from 0.05 m to 0.03 m, the agent succeeds in attaining in the

worst-case scenario 73.4 % deformations. If we only change the database to the large one, the agent realizes in the worst-case scenario 73.2 % deformations. For the last test, we do not reinitialize the environment, and we keep the other parameters constant. Specifically, the initial position of the SO in the current episode is the desired one achieved by the robot in the previous episode. Therefore, in order to succeed in this scenario, the agent needs to have learned a stronger, more general policy. In this more challenging scenario, the robot succeeds in making 87.9 % deformations while controlling two mesh nodes, 93.7 % deformations while controlling four mesh nodes, and 36.8 % deformations while controlling six mesh nodes. We can observe that the results for four mesh nodes are better than for two mesh nodes. Our interpretation is that describing the deformation of an object using only two mesh nodes is not precise enough; hence the agent has difficulty generalizing what it has learned during training.

The results prove that our approach is generalizable. We trained the agent using a small deformations database with a constant distance error threshold and reinitializing the environment after each episode. The agent can be more precise in the testing phase than in training, as shown by our tests with a lower distance error threshold. The agent achieves other deformations than those used during training without needing to be retrained. The agent makes the SO reach the desired deformation even if the object position is not reinitialized. Our method presents a limitation when we combine the changes in the testing phase. Sometimes it performs well, such as when we test the four mesh nodes control on the large database without reinitializing the environment: in this case, the robot achieves 72.8 % deformations. Sometimes the test fails, such as when we test the two mesh nodes control on the large database with a distance error threshold of 0.03. The robot achieves only 21.1 % deformations in this case.

6.3.2 Second experiment

For the second experiment, we trained the agent using the medium database to extract the desired deformations, and we tested it using the medium and the large database. All the rest of the experimental configuration remains the same as in the previous experiment. Figure 6.5 presents the average reward obtained in this experiment by the 32 agents in each episode when controlling two mesh nodes, four mesh nodes, and six mesh nodes. All the results of this experiment are presented in Table 6.2.

If we reinitialize the environment and use the same database and distance error threshold as in training, the agent achieves in the worst-case scenario 86.8 % deformations. Let's suppose we only change the distance error threshold to 0.03 m. In that case, the agent succeeds in attaining 38.2 % deformations while controlling two mesh nodes, 69.4 % deformations while controlling four mesh nodes, and 89.1 % deformations while controlling six mesh nodes. If we only change the database to the large one, the agent realizes in the worst-case scenario 62.3 % deformations. For the last test, we did not reinitialize the environment and kept the

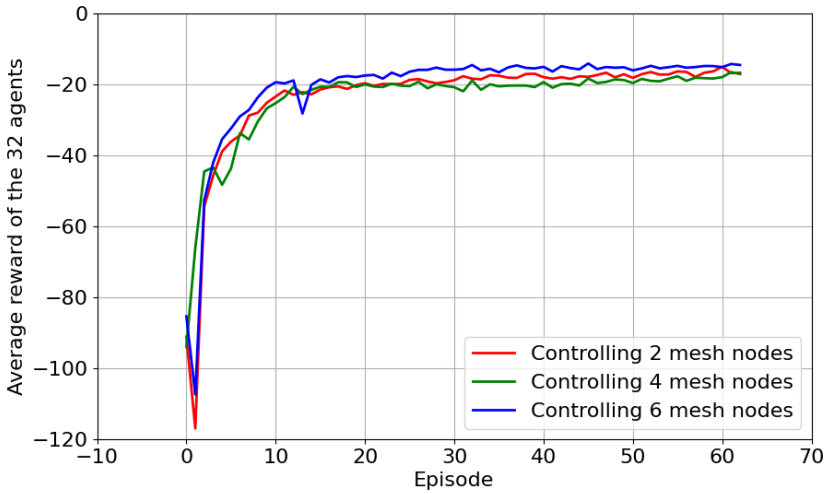


Figure 6.5: Average reward obtained in the second experiment by the 32 agents in each episode when controlling two mesh nodes, four mesh nodes, and six mesh nodes.

other parameters constant. In this more complex scenario, the robot succeeds in making 66.6 % deformations while controlling two mesh nodes, 75.2 % deformations while controlling four mesh nodes, and 74.1 % deformations while controlling six mesh nodes.

We have the same observations as in the first experiment in which we trained the agent using the small deformations database. The results obtained for four mesh nodes are better than for two mesh nodes. These new results keep proving that our control approach is generalizable. We trained the agent using a medium deformations database with a constant distance error threshold and reinitializing the environment after each episode. The agent can be more precise in the testing phase than in training, as shown by our tests with a smaller distance error threshold. The agent achieves other deformations than those used during training without needing to be retrained. The agent makes the SO reach the desired deformation even if the object position is not reinitialized. We can also observe the same limitation as in the previous experiment when we combine the changes in the testing phase. Sometimes it performs well, such as when we test the four mesh nodes control on the large database without reinitializing the environment: in this case, the robot achieves 72.1 % deformations. Sometimes the test fails, such as when we test the two mesh nodes control on the large database with a distance error threshold of 0.03. In this case, the robot achieves only 22.8 % deformations.

In general, we can notice from the conducted experiments that the agents trained using the small deformations database performed better than those trained using the medium deformations database. This is because when training using the small database, the desired goals are in a more restricted region of the 3D space, which allows the agents not to disperse and to learn what is useful directly.

	Mesh nodes number	Database	Testing error threshold (m)	Done (%)	Mean average distance error $\pm\sigma$ (m)	Best (m)
With reinitialization	2	Medium	0.05	86.8	0.04410 ± 0.00926	0.01657
			0.03	38.2	0.04507 ± 0.02302	0.01136
		Large	0.05	62.3	0.06501 ± 0.02863	0.02517
			0.03	22.8	0.06877 ± 0.03025	0.01650
	4	Medium	0.05	98.0	0.04420 ± 0.00816	0.02018
			0.03	69.4	0.02334 ± 0.01449	0.01902
		Large	0.05	86.2	0.05289 ± 0.02050	0.02674
			0.03	37.1	0.05454 ± 0.02739	0.01119
	6	Medium	0.05	96.9	0.04604 ± 0.00990	0.02184
			0.03	89.1	0.02771 ± 0.00620	0.01623
		Large	0.05	83.3	0.05652 ± 0.02529	0.02599
			0.03	36.7	0.05983 ± 0.03295	0.01925
Without reinitialization	2	Medium	0.05	66.6	0.05653 ± 0.02546	0.02072
			0.03	20.7	0.05567 ± 0.02314	0.01715
		Large	0.05	54.1	0.06432 ± 0.02884	0.01366
			0.03	18.1	0.05878 ± 0.02683	0.0938
	4	Medium	0.05	75.2	0.04807 ± 0.01651	0.01221
			0.03	17.5	0.05944 ± 0.02533	0.01304
		Large	0.05	72.1	0.06809 ± 0.03872	0.02494
			0.03	22.6	0.07924 ± 0.03584	0.01572
	6	Medium	0.05	74.1	0.05245 ± 0.01833	0.03283
			0.03	40.8	0.04827 ± 0.02470	0.02164
		Large	0.05	55.4	0.06053 ± 0.02554	0.03055
			0.03	20.2	0.06795 ± 0.03580	0.01656

Table 6.2: Results of all the conducted tests for the trainings using the medium database.

6.4 Sim-to-real transfer

As explained in Chapter 2, simulation environments are used for training the different DRL agents [79] because they offer potentially endless data sources and allay safety concerns with real robots. However, once the models are implemented in real robots, the gap between the simulated and real environments reduces the performance of the learned policies [90]. Therefore, numerous research works are currently focused on bridging this sim-to-real gap and achieving more efficient policy transfer [79]. An excellent approach for successfully transferring from simulation to the real environment is domain randomization [79, 90]. It consists of sampling simulation parameters (such as camera position, light position, textures, etc.) from probability distributions centered at a noisy estimate of the ground truth [90]. As a result, the agent can disregard small changes in the environment, making it more robust to domain changes. Thus, the agent can deal with the sim-to-real transfer [79, 90].

At first, we did a real test without using standard sim-to-real techniques such as domain randomization. We thought it was unnecessary to utilize them because we did not use cameras. Indeed, to localize and track the mesh nodes we control, we utilized a motion capture system that directly returns the 3D position of the markers. In this test, we consider the configuration of controlling four mesh nodes on the small database with a distance error threshold of 0.05 m. We reinitialize the environment at the end of each episode. The robot can well deform the object on the top X-Y planes but does not systematically deform the object correctly when the deformation is severe on the Z-axis. This is because the real SO bends naturally to one side, which is not necessarily the same bending side as in training using the simulator. Figure 6.6 shows a successful deformation on a X-Y plane and Figure 6.7 presents a failed deformation on the Z-axis. Other deformations are presented in the video available on `https://drive.google.com/drive/folders/1LYNDyK1RK6EKwAFX8JrIiWwHqf9fwICV?usp=sharing`.

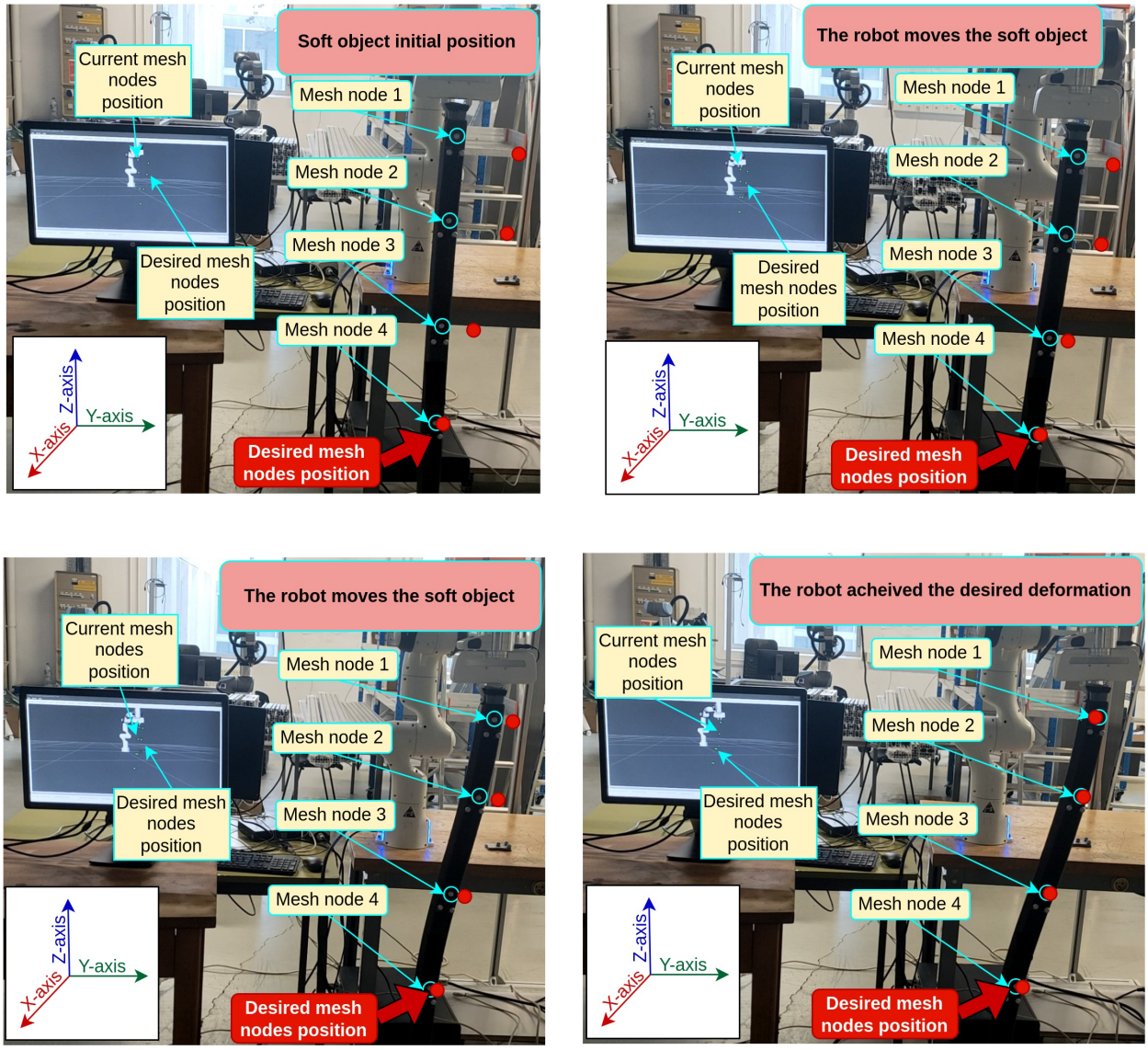


Figure 6.6: Example of a successful test of the robot deforming the SO: four mesh nodes reach their desired positions with an average distance error threshold of 0.05 m.

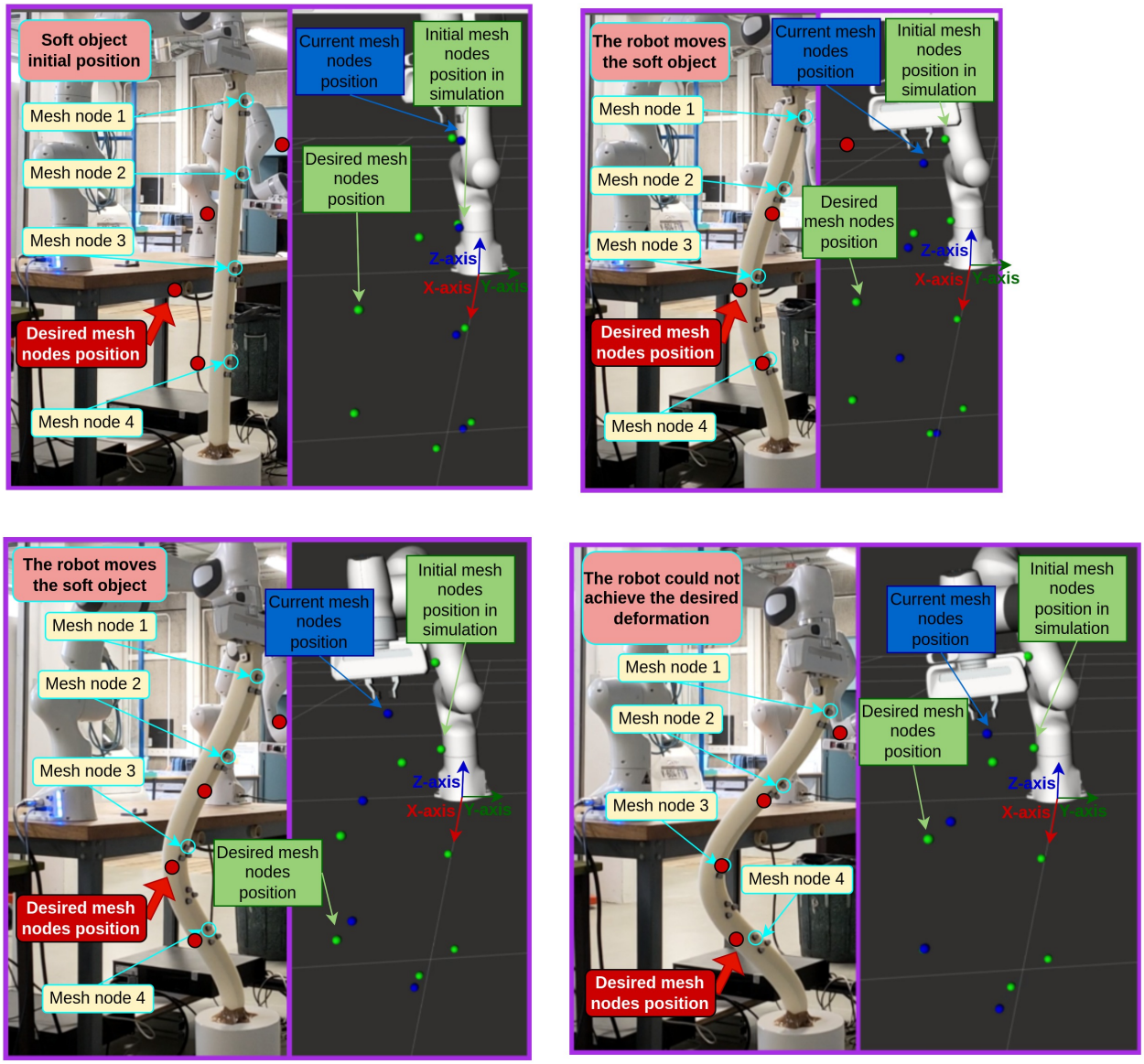


Figure 6.7: Example of a failed test of the robot deforming the SO: four mesh nodes do not reach their desired positions with an average distance error threshold of 0.05 m.

We discovered that the main sim-to-real gap problem is that the direction of the deformation of the SO is not always the same in simulation and reality. There should not be a big difference between the simulation and the reality. Otherwise, what the agent has learned will be useless because the process will not be Markovian anymore. Our interpretation is that it is due to the singularity of the initial position of the SO in the environment. Indeed, when the episode is initialized, the robot grasps the DLO positioned vertically. This position is singular since when the robot press the object vertically, it bends in a particular constant direction in the simulated tests. In contrast, in real experiments, the bending direction will vary from one object to another because of the natural plasticity of the object.

Database	Testing error threshold (m)	Done (%)	Mean average distance error (m)
Small	0.05	97	0.03169
	0.03	93	0.02291
Medium	0.05	96	0.03245
	0.03	91	0.02308
Large	0.05	86	0.05061
	0.03	76	0.04526

Table 6.3: Results of the conducted tests in simulation on the extended approach. These tests are performed for controlling four mesh nodes while reinitializing the environment after each episode.

We propose as a solution to always orient the gripper tip before moving it in the vertical direction to eliminate the singularity. We extend our simulated control approach to include a fixed initial orientation to the gripper. We conducted simulated tests while calculating the results for 100 episodes with 30 steps. Table 6.3 presents these results. Some of these results are shown in the video available on `https://drive.google.com/drive/folders/1LYNDyK1RK6EKwAFX8JrIiWwHqf9fwICV?usp=sharing`. We found that the results of the extended approach are mainly as good as those of the original one. In some cases, the results are even better, especially when the distance error threshold is smaller in the testing phase than in the training phase. For example, we controlled four mesh nodes using an average distance error threshold of 0.03 m in the testing phase (while in the training phase, it was equal to 0.05 m). With the original approach, the robot could achieve 88.7 % of the deformations, and with the extended approach, it can perform 93 % of them. However, the extended approach should also be tested through real experiments to ensure the solution is viable.

6.5 Conclusion

We have assessed through experiments that our control approach based on the DDPG algorithm is generalizable thanks to the fact that we parallelized the learning of the agent. The generalizability of our approach is proven because the agent (i.e., a single robot arm) can deform the SO starting from a different initial position and end up with a different desired shape without having to relearn. We verified this by training the agent on a small or a medium deformations database and testing it on a large deformations database. In the next chapter, we evaluate our control approach on two Panda robotic arms to verify that our approach can deal with dual-arm robots.

Chapter 7

A Dual-Arm Robotic approach for the Deformation of Soft Linear Objects Using Deep Reinforcement Learning

Contents

Chapter 5 introduced a robotic control approach for manipulating Soft Objects (SOs) based on a Deep Reinforcement Learning (DRL) approach. The previous chapter presented simulation results for a single-arm robotic agent that prove that our control approach is more generalizable than existing ones because we parallelized the agent's learning. In this chapter, we settle all the parameters of the Deep Deterministic Policy Gradient (DDPG) algorithm and the simulation parameters to make the agent that is made of two Panda arms able to deform a soft linear object. Our simulation results confirm the approach keeps its generality benefit when applied to

a dual-armed system. Those results are as good as those presented in Chapter 6. They are even, in some cases, better, i.e., when we combine the changes of the parameters in the test phase. This is possible thanks to the new reward function we use. Indeed, the new reward values are based on the maximum Euclidean distance between the current and the desired position of the mesh nodes, whereas in Chapter 6, they were based on the average Euclidean distance.

7.1 Problem statement

We address the problem of controlling the deformation of a Deformable Linear Object (DLO) by a dual-armed system. Indeed, a DRL agent controls two Panda robots that each is grasping one end of the SO. A mesh represents the object, and we describe its deformation by a set of selected mesh nodes. The agent's objective is to make the positions of the selected nodes reach their desired positions by controlling both robotic arms. Manipulating a SO with a dual-armed system is more interesting because the two arms can achieve more complex deformations than the ones performed by only one. The difficulty of this indirect control problem is that the dynamical model of the system to be controlled is complex and uncertain. The problem setup is illustrated in Figure 7.1.

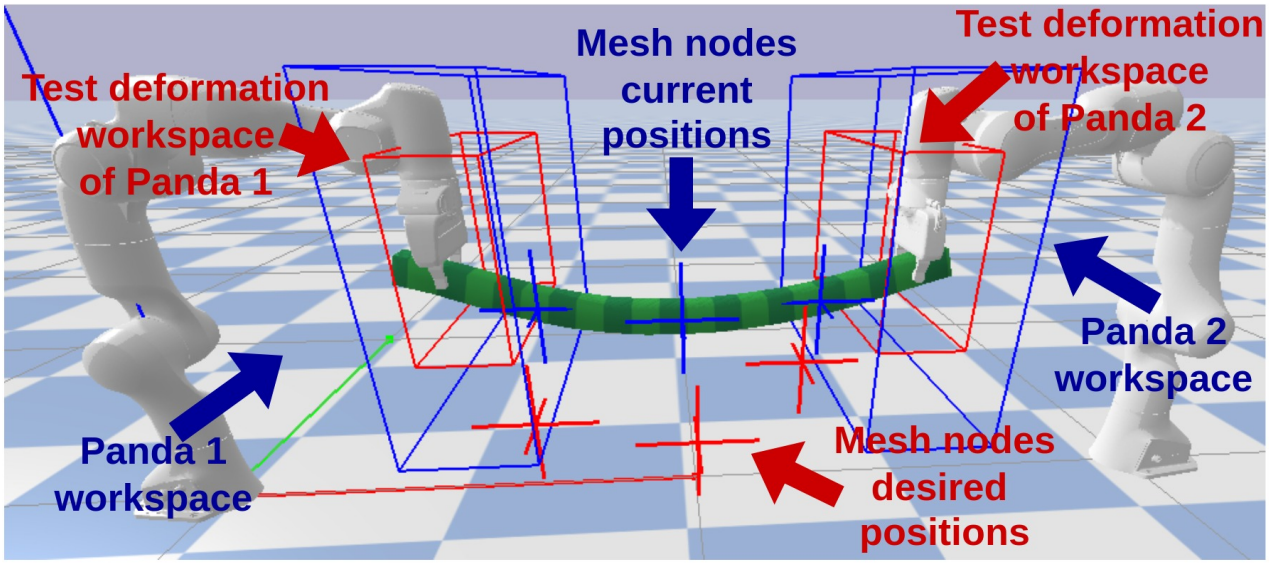


Figure 7.1: The setup we consider, including an illustration of some elements of our methodology. The agent deforms the soft linear object (green) by making the selected mesh nodes (i.e., the blue points) reach the desired corresponding positions (i.e., the red points). The points are marked as crosses. The agent controls both Panda arms to deform the soft object. Each arm tip position must remain within its corresponding deformation workspace (i.e., the red boxes) to perform a desired deformation. Each arm deformation workspace used in testing is bigger than the training workspace. The blue boxes delimit the robots' workspaces, i.e., each robot's gripper tip cannot reach a position out of that box.

We train and evaluate our approach in simulation. Our evaluation is carried out in diverse

conditions, and it validates the capability of the proposed approach. The following section presents all the parameters of our approach, whose values must be set according to the dual-arm application.

7.2 Implementation

After having introduced all the necessary DRL components in Section 5.3, we describe in this section how we apply them to address the specific problem scenario considered (Section 7.1). We provide the implementation details, including all assigned values for the DDPG and simulation parameters.

7.2.1 Approach overview

Before starting the learning phase, we create for each robot a deformation space box within which the robot gripper tip moves to deform the object, and we record the positions of the selected nodes P_d in a database. The reason for using a deformation space box for each robotic arm is to record several deformations within a limited space that is reachable by the gripper tip. We have created different databases, each based on boxes of different size: the training one, which is smaller, and the testing ones, which are larger. All the details about the databases are mentioned in Section 7.2.3. The robots' objective is to manipulate the object so that the current positions of the selected nodes P_c reach the desired positions P_d within a tolerance threshold.

Figure 5.2 gives an overview of our architecture. The action a^k given by the DDPG to the agent (i.e., robot 1 and robot 2) is the Cartesian velocity of the gripper tip of each robot $a^k \in \mathcal{A} = (V_{x_1}, V_{y_1}, V_{z_1}, V_{x_2}, V_{y_2}, V_{z_2}) \implies a^k \in \mathbb{R}^6$. The action a^k is continuous since each element of the velocity ($V_{x_1}, V_{y_1}, V_{z_1}, V_{x_2}, V_{y_2},$ or V_{z_2}) can have any value within the interval $[-1, 1]$. Then, the action a^k is integrated according to the timestep (which is equal to 0.06 s) to calculate the new grippers tip positions $(X_{n_1}, Y_{n_1}, Z_{n_1}, X_{n_2}, Y_{n_2}, Z_{n_2})$. The classical position-based controller available in Bullet moves both arms from their current position $(X_{c_1}, Y_{c_1}, Z_{c_1}, X_{c_2}, Y_{c_2}, Z_{c_2})$ to their new one $(X_{n_1}, Y_{n_1}, Z_{n_1}, X_{n_2}, Y_{n_2}, Z_{n_2})$. Since the DLO is grasped by each gripper tip, moving the grippers tip will move the mesh nodes a_o^k to some new positions P_c.

The state $s^k \in \mathcal{S}$ is made up of each robot gripper tip current state $s_g^k \in \mathbb{R}^{12}$ and the current object shape $s_o^k \in \mathbb{R}^{6m}$ (cf. (7.1)) with m the number of selected mesh nodes. s_g^k includes both robots gripper tip positions $(X_{c_1}, Y_{c_1}, Z_{c_1}, X_{c_2}, Y_{c_2}, Z_{c_2})$ and their velocities $(V_{x_1}, V_{y_1}, V_{z_1}, V_{x_2}, V_{y_2}, V_{z_2})$. s_o^k is composed of the positions of the selected mesh nodes $P_c \in \mathbb{R}^{3m}$, and their desired positions $P_d \in \mathbb{R}^{3m}$.

$$s^k = (s_g^k, s_o^k) \in \mathcal{S} = (X_{c_1}, Y_{c_1}, Z_{c_1}, X_{c_2}, Y_{c_2}, Z_{c_2}, V_{x_1}, V_{y_1}, V_{z_1}, V_{x_2}, V_{y_2}, V_{z_2}, P_c, P_d). \tag{7.1}$$

We calculate the reward r^k as the maximum Euclidean distance D^k between the current positions of the selected mesh nodes P_c and their desired positions P_d (cf. (7.2)).

$$r^k = f_u(\{\mathbf{G}\}) = -Max(D^k(P_c, P_d)). \tag{7.2}$$

7.2.2 DDPG parameters

We kept mainly the DDPG parameters as they were settled in Chapter 6. We only change the number of training episodes. We train the 32 agents during 81 episodes, equivalent to $32 * 81 = 2592$ episodes if we use a single agent and do not parallelize the training. This section is a reminder of all the DDPG parameters.

The actor, actor target, critic, and critic target Deep Neural Networks (DNNs) have the same architecture: 3 Fully Connected (FC) hidden layers, each of which comprises 256 neurons. We use the ReLU as an activation function. We apply the Tanh function on the actor outputs a^k to ensure that the grippers tip velocities remain in the interval $[-1, 1]$. We add noise to the action a^k using Ornstein-Uhlenbeck noise [203] for the exploration. We initialize the DNNs of the actor and critic with random values as in [203]. The actor target and critic target DNNs weights copy those of the actor and critic DNNs. The ADAM optimizer is used for gradient updates with learning rates of $\alpha_A = 0.0001$ for the actor and $\alpha_C = 0.001$ for the critic. A batch of 128 transitions is randomly sampled from the replay buffer, containing 50000 transitions. We use a constant discount factor $\gamma = 0.99$ and a constant Polyak factor $\tau = 0.01$.

Since we use parallel learning, in each episode (or epoch), 32 agents are trying to achieve a different deformation during 300 transitions. This means that each agent makes 300 actions and passes through 300 different transitions to try to achieve 32 different goals (each agent has a different goal). Each action will have a reward r^k, and each agent will have a global reward equal to the sum of the action reward over the 300 transitions. This leads to having different gradients that are synchronized among the 32 agents, i.e., there will be one final gradient equal to the sum of all the 32 gradients. Then 32 agents networks are updated based on that final gradient so that all these networks keep having the same updated weights. We train the 32 agents during 81 episodes, equivalent to $32 * 81 = 2592$ episodes if we use a single agent and do not parallelize the training. The training lasts from 1000 to 1 million episodes in the literature [161, 189]. For training 32 agents, we used 32 CPU cores and the Python library MPI [211].

7.2.3 Simulation parameters

Compared to Chapter 6, we increase the calculation time of the solver that computes the deformation of the simulated SO by adding a waiting time of 30 s to have more precise deformation results. We also change the 3D size of the deformation boxes to adapt them to the new environment configuration. We settle all the values of all the other simulation parameters

to the prescribed values in Chapter 6. We still use PyBullet as the simulator's physics engine to train our agent. The simulator's physics engine uses the FEM method [212] to simulate the SO, and we kept the values of its mechanical parameters as they were in Chapter 6.

We created three databases. Each database was created by moving both arms in their corresponding deformation box to deform the object, and we recorded those deformations to use them as desired positions P_d in the training and the testing phase. The 3D size of the deformation boxes varies from one database to another. Figure 7.2 shows the small, medium, and large boxes that have the same 3D size for both arms. The small box size equals 0.15 m on the x-axis, 0.5 m on the y-axis, and 0.3 m on the z-axis. The medium box size equals 0.15 m on the x-axis, 0.6 m on the y-axis, and 0.4 m on the z-axis. The large box size equals 0.2 m on the x-axis, 0.8 m on the y-axis, and 0.5 m on the z-axis. The large box 3D size is equal to the 3D size of the workspace of the robotic arms. The small boxes generate the small database, which contains 1000 deformations. The medium boxes are used to generate the medium database, which includes 1000 deformations. The large boxes generate the large database, which has 1000 deformations. The small database is used for the training, and all databases (i.e., small, medium, and large databases) are used for the testing.

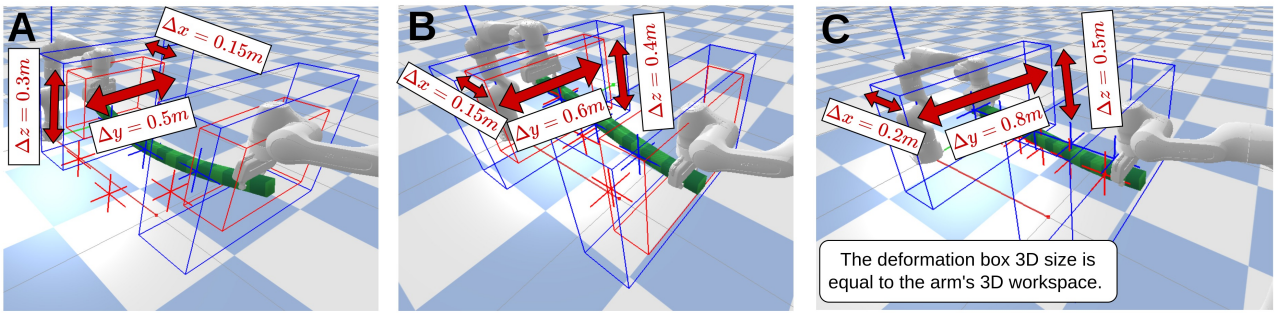


Figure 7.2: 3D boxes of different sizes are used for database generation. Each robot has its own deformation box. **(A)** The small boxes (in red) are used to generate the small database. **(B)** The medium boxes (in red) are used to generate the medium database. **(C)** The large boxes (in red) are used to generate the large database.

7.3 Experiments

This section presents our experimental results in simulation for all conducted experiments. Since training the agent using a small database and testing it using a larger one gave the best results in Chapter 6, we followed the same procedure in this chapter.

We trained the agent (i.e., both Panda arms) using the small database to extract the desired deformations, i.e., the desired positions P_d of the mesh nodes. We have done two trainings to control: three mesh nodes, and five mesh nodes. We used a maximum distance error threshold of 0.05 m. As mentioned in the previous section, the training was parallelized: 32 agents were trained each for 81 episodes, leading to having 2592 episodes in total. During the training, the

environment was reset to the initial configuration (both robots and the object returned to their initial position) after each episode. Figure 7.3 shows the average reward obtained by the 32 agents in each episode when controlling three mesh nodes, and five mesh nodes. As we can notice from Figure 7.3, there is no need to smooth the learning curves, as in the literature [161, 189]. This is thanks to the stability of the learning due to its parallelization.

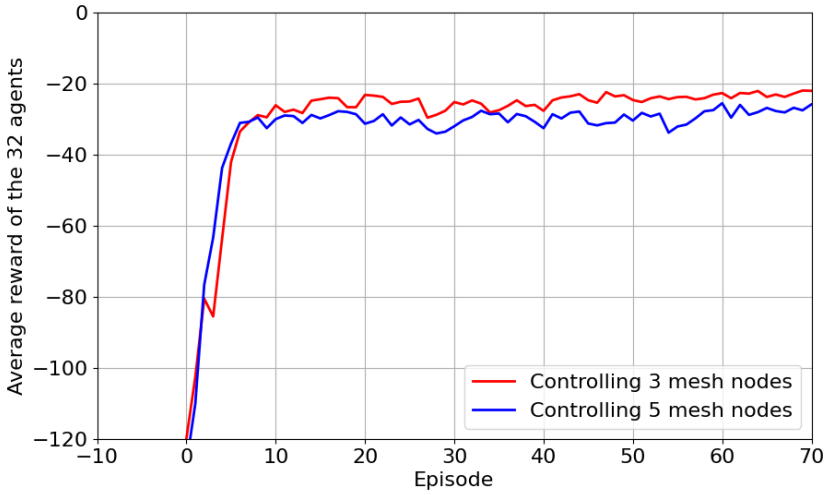


Figure 7.3: Average reward obtained by the 32 agents in each episode when controlling three mesh nodes and five mesh nodes.

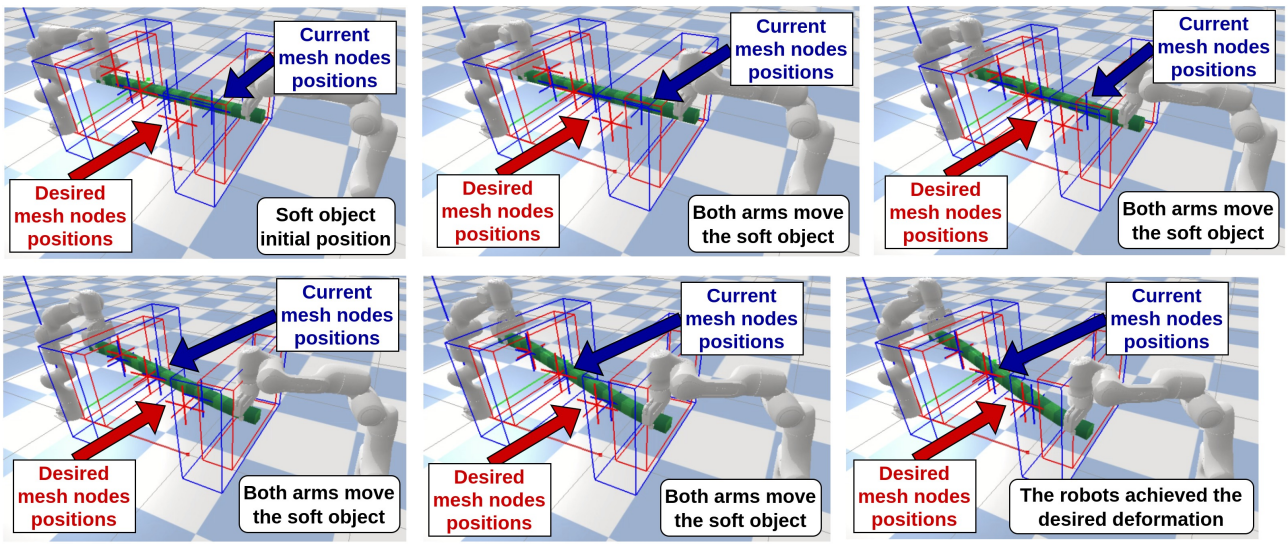


Figure 7.4: Example of the robot deforming the SO: three mesh nodes reach their desired positions with a maximum distance error threshold of 0.05 m.

For the testing phase, all the results are calculated for 1000 testing episodes with 30 steps, i.e., the agent can take a maximum of 30 actions to achieve the deformation. We test the two trainings for a maximum distance error threshold of 0.05 m and 0.03 m. We evaluate them using the small, the medium, and the large databases to extract the desired deformations. We assess

them finally with and without reinitializing the environment. All these results are presented in Table 7.1. In Table 7.1, the column "done" indicates the percentage of the agent's success to achieve the desired deformations. The percentage is calculated on the 1000 episodes with 30 steps. The "Best" column reveals the minimum distance error obtained within the 1000 episodes.

	Mesh nodes number	Database	Testing error threshold (m)	Done (%)	Mean average distance error $\pm\sigma$ (m)	Best (m)
With reinitialization	3	Small	0.05	91.7	0.03859 ± 0.01166	0.01113
			0.03	73.6	0.03021 ± 0.01334	0.01052
		Medium	0.05	97.2	0.03822 ± 0.01101	0.00998
			0.03	77.6	0.02860 ± 0.01214	0.01009
		Large	0.05	84.1	0.04270 ± 0.01438	0.01254
			0.03	58.2	0.03568 ± 0.01738	0.00891
	5	Small	0.05	67.3	0.04939 ± 0.01863	0.01189
			0.03	34.8	0.04656 ± 0.02078	0.01286
		Medium	0.05	72.1	0.04964 ± 0.01787	0.01733
			0.03	37.4	0.04401 ± 0.02202	0.010973
		Large	0.05	60.1	0.05718 ± 0.02636	0.02584
			0.03	33.7	0.05365 ± 0.02934	0.01356
Without reinitialization	3	Small	0.05	84.2	0.04584 ± 0.02475	0.01190
			0.03	67.3	0.03604 ± 0.02491	0.01104
		Medium	0.05	90.2	0.04433 ± 0.02866	0.01325
			0.03	62.3	0.03243 ± 0.02566	0.01120
		Large	0.05	70.8	0.06176 ± 0.03656	0.02119
			0.03	52.1	0.04543 ± 0.03835	0.01405
	5	Small	0.05	65.3	0.04978 ± 0.01887	0.01293
			0.03	31.3	0.04784 ± 0.02247	0.01222
		Medium	0.05	69.2	0.05027 ± 0.02060	0.02142
			0.03	35.9	0.04691 ± 0.02286	0.00820
		Large	0.05	59.4	0.05780 ± 0.02666	0.02094
			0.03	29.8	0.05510 ± 0.02972	0.01266

Table 7.1: Results of all the conducted tests for the trainings using the small database.

Figure 7.4 presents an example of the dual-arm agent deforming a SO to reach a new deformation on which the agent was not trained. Other deformations are presented in the video available on `https://drive.google.com/drive/folders/15bbEugspTJfKj6urdxYxZlEC5PuQlps3?usp=sharing`. As we can notice from Table 7.1, for controlling three mesh nodes, the results are mostly as good as the results achieved by the single arm while controlling two, four, or six mesh nodes (cf. Table 6.1). Sometimes there is a slight reduction in performance, such as when we reinitialize the environment and use the

same database and distance error threshold as in training, the single-arm agent achieves in the worst-case scenario 99.1 % deformations. In contrast, when controlling three mesh nodes, the dual-arm agent performs 91.7 % deformations. However, the dual-arm agent performs better when combining the testing parameters changes. For example, when the single-arm agent controls two mesh nodes on the large database with a distance error threshold of 0.03. In this case, the robot achieves only 21.1 % deformations. In contrast, the dual-arm agent performs 58.2 % deformations when controlling three mesh nodes and achieves 33.7 % deformations when controlling five mesh nodes.

The results prove that our approach keeps its generalizability whether we test it on a single-arm or dual-arm agent. We trained the agent using a small deformations database with a constant distance error threshold and reinitializing the environment after each episode. The agent can be more precise in the testing phase than in training, as shown by our tests with a lower distance error threshold. The agent achieves other deformations than those used during training without needing to be retrained. The agent makes the SO reach the desired deformation even if the object position is not reinitialized. The results presented in Table 7.1 show some weakness when the dual-arm agent controls five mesh nodes. There is a significant reduction in the performance compared to when the dual-arm agent controls three mesh nodes. Our interpretation is that this is because controlling more mesh nodes is getting more complex. We can notice from Figure 7.3 that the rewards received by the dual-arm agent when controlling five mesh nodes are smaller than when controlling three mesh nodes. The solution to improve the results is to train the dual-arm agent longer (in terms of the number of episodes) when it controls more mesh nodes.

7.4 Conclusion

In this chapter, we tested our control approach on a dual-armed system. We changed the reward function and calculated it as the maximum Euclidean distance between the current positions of the selected mesh nodes and their desired positions. We also left the simulator to calculate the deformation of the DLO for a longer time. We made these modifications to increase the dual-arm agent's performance when combining the changes in the testing phase parameters. Indeed, when the single-arm agent was controlling, for example, two mesh nodes on the large database with a distance error threshold of 0.03, it achieved only 21.1 % deformations. In contrast, the dual-arm agent performs 58.2 % deformations when controlling three mesh nodes and achieves 33.7 % deformations when controlling five mesh nodes. The agent can deform the SO starting from a different initial position and end up with a different desired shape without relearning. We verified this by training the agent on a small deformations database and testing it on larger ones.

Chapter 8

General conclusion and Perspectives

Contents

8.1 Summary and conclusion

The rapid growth in demand for service robot applications in the home or industries has raised interest in Human-Robot Collaboration (HRC) during the past several years. Therefore, it is crucial to make robots endowed with pertinent abilities. Several effective robots with sophisticated proprioception sensing and actuation control have been created. Nevertheless, these robots still lack sufficient autonomy to work effectively with people.

8.1.1 HRC classifications

Much research has been conducted to improve the autonomy of robots. We have reviewed the most relevant of them in Part I. In Chapter 1, we started by presenting how HRCs can be classified. Robots' decisional autonomy can be increased by classifying the HRCs based on their similitudes. One more autonomous robot can take the place of two robots that should have comparable aptitudes and carry out similar tasks.

8.1.2 Main components required to set up a HRC

Another way to enhance the robots' decisional autonomy is to improve one of the main components required to set up a HRC: perception, Decision-Making (DM), motion execution, and evaluation. In the remaining of Chapter 1, we introduced them briefly. The perception comprises speech recognition, object detection, location recognition, posture recognition, and human intention detection. Trajectory planning includes determining the optimal trajectory for carrying out the robot's action. A suitable controller is necessary for low-level control because it enables the robot to move precisely into the desired position. The DM allows the robot to select the best actions to make by taking into account the task, the environment, the previous actions of the agents (humans and other robots), and the performance metrics that the agents are trying to maximize to optimize collaboration. The HRC evaluation is typically based on performance metrics to determine how beneficial the collaboration is for human agents. We can consider as performance metrics the task duration, human posture, task accuracy, robot velocity, robot's dexterity, etc.

8.1.3 Robot's DM process

In this thesis, we aim to improve the HRC by maximizing the benefits of collaboration for human agents. Enhancing the robot's perception and control approaches is one way to do this. The other is to improve the robot's DM process by taking performance metrics into account. We favor using the second way because it enables us to optimize collaboration at a higher level by taking into account various metrics even more because the robot's perception and control may be integrated into the robot's DM process. DM techniques are used to make robots able to adapt themselves to humans while accomplishing a task in collaboration. A DM process is made of three main parts:

- A **DM method:** models the relationship between the agents, the actions, the environment, and the task.
- A **DM strategy:** is the policy of choosing actions based on the value of their reward calculated by the reward function.
- A **reward function (or utility function):** calculates a reward for each action.

In chapter 2, we reviewed the DM methods, DM strategies, and utility functions used within the robot's DM process in the literature to place our contributions with respect to them.

8.1.4 Our framework for optimizing the HRC

This thesis aims to increase the performance of HRC through the DM process of the robots. A collaboration comprises human and robot agents working together to accomplish the task. The objective of previous work in the area of HRC is to adapt the robot to human behavior to perform a task without taking into account how well the interaction unfolds. Some previous studies provide specialized frameworks to enhance the HRC performance by taking into account fixed performance metrics that are difficult to be modified within their architectures. Unlike the state-of-the-art frameworks, the one we developed and presented in Chapter 3 can easily handle changing performance metrics from one case scenario to another. Our framework treats HRC as a constrained optimization problem with a utility function (or reward function) that is divided into three major components. Firstly, a reward assesses the performance of the collaboration, and it is the only part that is altered when modifying the performance metrics. It allows for control over how the interaction proceeds and ensures that the robots' behaviors will be directly adjusted to those of the human agents involved in the collaboration. Secondly, a constraint specifies how to complete the task. Thirdly, a set containing the robots' physical abilities. Accordingly, the utility function may be designed to enhance the robot's dexterity in manipulating objects.

8.1.5 First application of our framework

Firstly, we tested this framework through an assembly task. For this application, the human agent was collaborating with the Nao robot. This robot has great manipulation limitations (its motions are inaccurate) that can hardly be enhanced. That is why we focused the reward function on improving the collaboration performance by changing the performance metrics considered in it without trying to enhance the robot's manipulation dexterity. We were optimizing the collaboration based on several performance metrics, such as the time to completion of the task and the number of human errors. We were doing that independently of the human agent behavior.

We carried out experiments in real and simulation. By using our proposed utility function instead of the state-of-the-art utility function, we improved the experiment time up to 66.7 %. Theoretically, this improvement can reach a value close to 100 %. We also got a percentage of human errors reduction up to 50.6 % by considering the predicted probability that the human makes errors for optimizing the time to completion. From these conducted experiments, even though we were able to optimize the collaboration, the improvement was restrained because the abilities of collaborative robots are very limited, especially Nao. It was unable to do the pick

and place task adequately. To further improve the optimization of the HRC performance, we need to increase the robot's manipulation dexterity.

8.1.6 Second application of our framework

Secondly, we focused on enhancing the robot's manipulation dexterity while it performed a more complex task, such as a Soft Object (SO) manipulation task. To the best of our knowledge, no work involving the co-manipulation of a SO by a human-robot collaborative team has been performed using Deep Reinforcement Learning (DRL). Our thesis perspective lies in testing our framework on such applications. Chapter 5 presents the background on SOs manipulation (especially the deformable linear objects) and the DRL approaches we used.

Since this application is highly challenging, we divided it into four steps. The first one is to make the robot able to deform the SO thanks to a DRL approach. In Chapter 6, we tested our DRL control approach on a single robotic arm. The results prove that our approach can generalize the initial and final desired deformation of the SO more easily than the existing ones in the literature. The second step consists of applying our control approach to a dual-arm robotic agent that deforms the SO. The conducted tests in Chapter 7 prove that the approach can generalize the initial and final desired deformation as well as when it was tested on a single-arm robot.

The next section introduces how we plan to manage the last two steps in our future work. The last two steps consist of replacing one of the robotic arms with a human agent, then performing tests in real, and optimizing the collaboration based on some performance metrics.

8.2 Future work

Our perspectives are to accomplish the last two steps required to test our framework on a co-manipulation task involving deformable objects. For this, we have to test our DRL control approach in real for the single-arm and the dual-arm robot. Then, we should adjust our DRL agent to make it able to consider the human agent's behavior. Finally, we will have to test the DRL agent in real while it performs that task with a human agent.

8.2.1 Sim-to-real transfer for the single-arm and the dual-arm robot

As stated in Chapter 6, our main problem with achieving a sim-to-real transfer is that the direction of the deformation of the SO is different in reality and simulation. The disparity between the simulation and reality should be slight. Otherwise, the process will no longer be Markovian, making the agent's learning incorrect. According to our understanding, it is caused by the singularity of the SO initial position in the environment. In fact, when the episode is initialized, the robot grasps the SO that is vertically positioned. This position is singular

because, in the simulated tests, the object bends in a specific constant direction when the robot presses it vertically. In contrast, in real experiments, different objects will bend in different directions because of the object's inherent plasticity.

To avoid singular positions, we suggest orienting the gripper in some direction before moving the gripper tip vertically. We put the gripper tip in a predetermined initial orientation when the episode is initialized. We ran simulated tests and discovered that by adding this extension, the results are as good as those of the original environment. Sometimes the outcomes are even better, particularly when the distance error threshold is lower during testing than during training. For instance, we controlled four mesh nodes during the testing phase with an average distance error threshold of 0.03 m (while in the training phase, it was equal to 0.05 m). Using the extension, the robot could execute 93 % of the deformations while it only achieved 88.7 % of the deformations using the original approach. To ensure this solution is valid, this extension should also be tested in real. We are currently setting up the real platform to perform those tests. A video of some successful first tests is available on `https://drive.google.com/drive/folders/17LCKQBX_TKwwtWVKPWQ1EPDyY16N6HFo?usp=sharing`. We plan to evaluate this solution further, integrate the results into a paper and submit it to an international journal.

8.2.2 Collaborating with the human agent

To perform this step and make the robot collaborates with the human agent, the human agent's behavior should be included in the simulator. Since predicting a human agent's behavior is challenging, we can use a multi-agent solution instead. Some studies have focused on Multi-Agent Deep Reinforcement Learning (MADRL) [164–166] approaches to cope with this problem. The MADRL approaches deal with multiple agents that share the same environment [165] but behave independently. The idea behind these approaches is that each agent has to know the actions of the others and their impact on the environment to collaborate with them [166]. To apply this solution to our HRC application involving the manipulation of SO, we will have to consider as individual agents both Panda arms. The first Panda arm will represent the robot, and the second one will represent the human. Then, the human agent behavior will be approximated by an individual learning robotic agent. The use of these techniques could be valid for our application, but to confirm that, we will have to test them.

8.2.3 Evaluating the DRL agent through a real co-manipulation task

At last, we will need to integrate some performance metrics in the reward function to optimize the HRC performance and evaluate if the optimization is higher than the one obtained within our first application (i.e., the assembly task). As performance metrics of the co-manipulation task, we can consider the following metrics: the time to achieve the deformation, the accuracy of the reached deformation, the robot's dexterity, and the robot's velocity to adapt to the human.

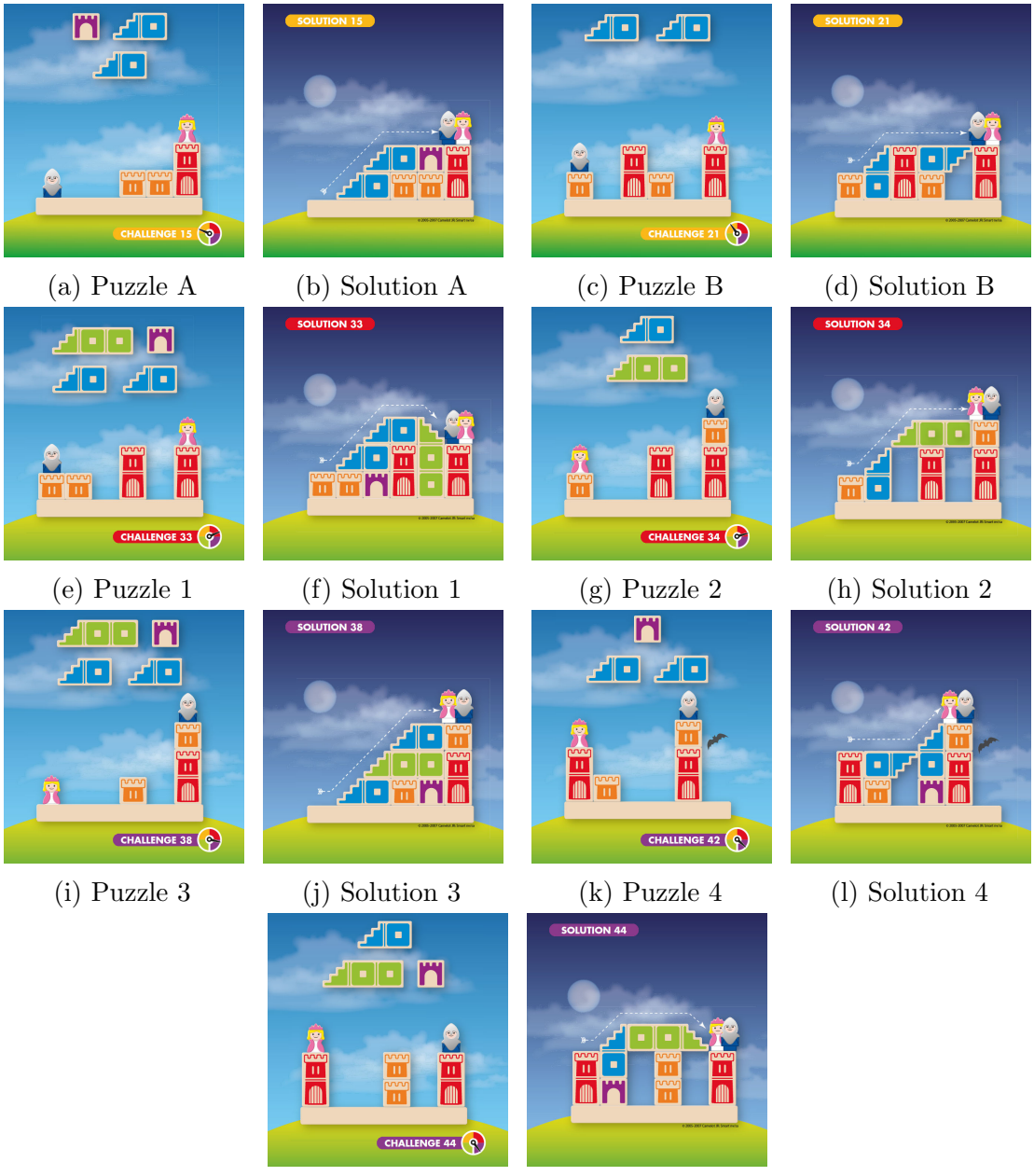
Part IV

Appendices

Appendix A

Puzzles from the Camelot Jr. game

In this appendix, we present all the puzzles from the Camelot Jr. game that we used in Chapter 4. Puzzles A and B are those we used to make sure that the human participants understood the gameplay.



(a) Puzzle A (b) Solution A (c) Puzzle B (d) Solution B

(e) Puzzle 1 (f) Solution 1 (g) Puzzle 2 (h) Solution 2

(i) Puzzle 3 (j) Solution 3 (k) Puzzle 4 (l) Solution 4

(m) Puzzle 5 (n) Solution 5

Figure A.1: Puzzles from the Camelot Jr. game used in Chapter 4.

Appendix B

Simulation results of Section 4.6.4

Table B.1 presents the numerical values of the percentage of time improvement and the reduction of the number of human errors for all the figures presented in Section 4.6.4.

	Percentage of time improvement		Percentage of human errors reduction	
I_1, I_2, and I_3 values	Figure 4.7	Figure 4.8	Figure 4.9	Figure 4.10
$I_1 = 0$, $I_2 = 0.1$, and $I_3 = 0.9$	0.0	0.0	7.09412331431	4.37067607062
$I_1 = 0$, $I_2 = 0.2$, and $I_3 = 0.8$	0.0	0.0	12.6336858252	7.8020796966
$I_1 = 0$, $I_2 = 0.3$, and $I_3 = 0.7$	0.0	0.0	16.556882222	10.4704561576
$I_1 = 0$, $I_2 = 0.4$, and $I_3 = 0.6$	0.0	0.0	19.2303631915	13.3638679771
$I_1 = 0$, $I_2 = 0.5$, and $I_3 = 0.5$	0.0	0.0	22.3643444888	15.4993421616
$I_1 = 0$, $I_2 = 0.6$, and $I_3 = 0.4$	0.0	0.0	24.8542177267	17.9408050283
$I_1 = 0$, $I_2 = 0.7$, and $I_3 = 0.3$	0.0	0.0	28.4374851075	19.6044166944
$I_1 = 0$, $I_2 = 0.8$, and $I_3 = 0.2$	0.0	0.0	31.2453571429	22.1412554113
$I_1 = 0$, $I_2 = 0.9$, and $I_3 = 0.1$	0.0	0.0	29.2866666667	22.7585714286
$I_1 = 0$, $I_2 = 1$, and $I_3 = 0$	0.0	0.0	0.0	0.0
$I_1 = 0.1$, $I_2 = 0$, and $I_3 = 0.9$	0.0	0.0	17.5302972429	7.33482163197
$I_1 = 0.1$, $I_2 = 0.1$, and $I_3 = 0.8$	0.0	0.0	19.2143318311	10.5479301203
$I_1 = 0.1$, $I_2 = 0.2$, and $I_3 = 0.7$	0.0	0.0	21.3374721068	14.0052043345
$I_1 = 0.1$, $I_2 = 0.3$, and $I_3 = 0.6$	0.0	0.0	23.5786203979	16.5280674286
$I_1 = 0.1$, $I_2 = 0.4$, and $I_3 = 0.5$	0.0	0.0	26.7249188197	18.5858323843
$I_1 = 0.1$, $I_2 = 0.5$, and $I_3 = 0.4$	0.0	0.0	27.5937682456	20.9918095099
$I_1 = 0.1$, $I_2 = 0.6$, and $I_3 = 0.3$	0.0	0.0	28.9937657713	22.1913226588
$I_1 = 0.1$, $I_2 = 0.7$, and $I_3 = 0.2$	0.0	0.0	32.0019444444	22.9278030303
$I_1 = 0.1$, $I_2 = 0.8$, and $I_3 = 0.1$	0.0	0.0	34.205	23.5061904762
$I_1 = 0.1$, $I_2 = 0.9$, and $I_3 = 0$	0.646808142428	0.441773745339	0.0	0.0
$I_1 = 0.2$, $I_2 = 0$, and $I_3 = 0.8$	0.0	0.0	25.4460206798	12.376411145
$I_1 = 0.2$, $I_2 = 0.1$, and $I_3 = 0.7$	0.0	0.0	25.6904326925	15.6962303954
$I_1 = 0.2$, $I_2 = 0.2$, and $I_3 = 0.6$	0.0	0.0	27.5106709889	18.315465336
$I_1 = 0.2$, $I_2 = 0.3$, and $I_3 = 0.5$	0.0	0.0	30.0794412809	20.4125719276
$I_1 = 0.2$, $I_2 = 0.4$, and $I_3 = 0.4$	0.0	0.0	30.8272285354	22.0221180209
$I_1 = 0.2$, $I_2 = 0.5$, and $I_3 = 0.3$	0.0	0.0	31.9211207311	24.9664681615
$I_1 = 0.2$, $I_2 = 0.6$, and $I_3 = 0.2$	0.0	0.0	34.119047619	24.4444642857
$I_1 = 0.2$, $I_2 = 0.7$, and $I_3 = 0.1$	0.765528401311	0.0	0.0	27.6111904762
$I_1 = 0.2$, $I_2 = 0.8$, and $I_3 = 0$	2.4788012545	1.80659141302		0.0
$I_1 = 0.3$, $I_2 = 0$, and $I_3 = 0.7$	0.0	1.91789786313	30.2540826341	16.6733556418
$I_1 = 0.3$, $I_2 = 0.1$, and $I_3 = 0.6$	0.0	0.0	30.7411696561	19.7313028361
$I_1 = 0.3$, $I_2 = 0.2$, and $I_3 = 0.5$	0.0	0.0	31.3673661689	21.5925174216
$I_1 = 0.3$, $I_2 = 0.3$, and $I_3 = 0.4$	0.0	0.0	33.7247655123	23.7318043068
$I_1 = 0.3$, $I_2 = 0.4$, and $I_3 = 0.3$	0.0	0.0	34.3449001924	23.9794936545
$I_1 = 0.3$, $I_2 = 0.5$, and $I_3 = 0.2$	0.0	0.0	36.4016269841	26.2835119048
$I_1 = 0.3$, $I_2 = 0.6$, and $I_3 = 0.1$	2.77628815301	1.46088929863	36.7869047619	24.7346428571
$I_1 = 0.3$, $I_2 = 0.7$, and $I_3 = 0$	5.6292792232	4.03759880367	0.0	0.0
$I_1 = 0.4$, $I_2 = 0$, and $I_3 = 0.6$	0.0	0.0	36.3891268668	20.3970746112
$I_1 = 0.4$, $I_2 = 0.1$, and $I_3 = 0.5$	0.0	0.0	35.2710778111	22.4513601676
$I_1 = 0.4$, $I_2 = 0.2$, and $I_3 = 0.4$	0.0	0.0	35.8305131674	25.1917275086
$I_1 = 0.4$, $I_2 = 0.3$, and $I_3 = 0.3$	0.0	0.0	36.4988095238	26.7846236171
$I_1 = 0.4$, $I_2 = 0.4$, and $I_3 = 0.2$	2.97937356761	0.0	37.8678571429	26.1883928571
$I_1 = 0.4$, $I_2 = 0.5$, and $I_3 = 0.1$	6.8298290148	3.67128494973	35.0916666667	21.646547619
$I_1 = 0.4$, $I_2 = 0.6$, and $I_3 = 0$	10.0581040567	6.91767350379	0.0	0.0
$I_1 = 0.5$, $I_2 = 0$, and $I_3 = 0.5$	0.0	0.0	38.5427888223	22.5377313961
$I_1 = 0.5$, $I_2 = 0.1$, and $I_3 = 0.4$	0.0	0.0	39.8677200577	25.6673357198
$I_1 = 0.5$, $I_2 = 0.2$, and $I_3 = 0.3$	2.5889362939	0.0	40.2805687831	28.5753607504
$I_1 = 0.5$, $I_2 = 0.3$, and $I_3 = 0.2$	7.56507185318	2.4071413430	38.0336309524	21.5879166667
$I_1 = 0.5$, $I_2 = 0.4$, and $I_3 = 0.1$	12.187798206	7.11099379702	41.565	22.1403571429
$I_1 = 0.5$, $I_2 = 0.5$, and $I_3 = 0$	15.7060720797	11.0874137267	0.0	0.0
$I_1 = 0.6$, $I_2 = 0$, and $I_3 = 0.4$	0.0	0.0	41.0730555556	25.1399181374
$I_1 = 0.6$, $I_2 = 0.1$, and $I_3 = 0.3$	8.17205250781	0.0	42.012965368	28.6511640212
$I_1 = 0.6$, $I_2 = 0.2$, and $I_3 = 0.2$	14.1780122742	5.64235894103	43.109047619	21.2425054113
$I_1 = 0.6$, $I_2 = 0.3$, and $I_3 = 0.1$	18.8995795602	11.1593671564	45.3266666667	24.2171428571
$I_1 = 0.6$, $I_2 = 0.4$, and $I_3 = 0$	23.2260934025	15.568227852	0.0	0.0
$I_1 = 0.7$, $I_2 = 0$, and $I_3 = 0.3$	17.2439908187	3.33567251462	44.296547619	18.6905624931
$I_1 = 0.7$, $I_2 = 0.1$, and $I_3 = 0.2$	22.4988118634	10.0605063426	45.0121428571	20.8104816017
$I_1 = 0.7$, $I_2 = 0.2$, and $I_3 = 0.1$	27.4951742932	15.4641925539	43.6866666667	21.2117857143
$I_1 = 0.7$, $I_2 = 0.3$, and $I_3 = 0$	31.8854548846	21.023549533	0.0	0.0
$I_1 = 0.8$, $I_2 = 0$, and $I_3 = 0.2$	33.0590564877	14.5036859249	44.6876190476	22.0960714286
$I_1 = 0.8$, $I_2 = 0.1$, and $I_3 = 0.1$	37.9880475163	20.7778656126	45.2483333333	22.3210714286
$I_1 = 0.8$, $I_2 = 0.2$, and $I_3 = 0$	41.9390153589	26.6215414675	0.0	0.0
$I_1 = 0.9$, $I_2 = 0$, and $I_3 = 0.1$	49.8353835563	26.317921026	50.5583333333	21.8939285714
$I_1 = 0.9$, $I_2 = 0.1$, and $I_3 = 0$	53.3069306931	33.3469782673	0.0	0.0
$I_1 = 1$, $I_2 = 0$, and $I_3 = 0$	66.6666666667	40.0	0.0	0.0

Table B.1: Time improvement percentage and human errors reduction percentage obtained for all the figures presented in Section 4.6.4.

Bibliography

[1] R. Rosenberg-Kima, Y. Koren, M. Yachini, and G. Gordon, "Human-robot-collaboration (HRC): social robots as teaching assistants for training activities in small groups," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI), Daegu, South Korea, 11-14 March 2019*, pp. 522–523, IEEE, 2019.

[2] S. M. Fakhrhosseini, D. Lettinga, E. Vasey, Z. Zheng, M. Jeon, C. H. Park, and A. M. Howard, "Both "look and feel" matter: Essential factors for robotic companionship," in *26th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN 2017, Lisbon, Portugal, 28 August - 1 September 2017*, pp. 150–155, IEEE, 2017.

[3] H. Cao, R. Simut, N. Desmet, A. D. Beir, G. V. de Perre, B. Vanderborght, and J. Vanderfaeillie, "Robot-assisted joint attention: A comparative study between children with autism spectrum disorder and typically developing children in interaction with NAO," *IEEE Access*, vol. 8, pp. 223325–223334, 2020.

[4] M. Assad-Uz-Zaman, M. Rasedul Islam, S. Miah, and M. H. Rahman, "Nao robot for cooperative rehabilitation training," *Journal of Rehabilitation and Assistive Technologies Engineering*, vol. 6, p. 2055668319862151, 2019.

[5] R. Gervasi, L. Mastrogiacomo, and F. Franceschini, "A conceptual framework to evaluate human-robot collaboration," *The International Journal of Advanced Manufacturing Technology*, vol. 108, no. 3, pp. 841–865, 2020.

[6] J. DelPreto and D. Rus, "Sharing the load: Human-robot team lifting using muscle activity," in *2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20-24 May 2019*, pp. 7906–7912, IEEE, 2019.

[7] C. Brosque, E. Galbally, O. Khatib, and M. Fischer, "Human-robot collaboration in construction: Opportunities and challenges," in *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 20-24 June 2020*, pp. 1–8, IEEE, 2020.

[8] L. Peternel, N. Tsagarakis, D. Caldwell, and A. Ajoudani, "Robot adaptation to human physical fatigue in human-robot co-manipulation," *Autonomous Robots*, vol. 42, no. 5, pp. 1011–1021, 2018.

[9] A. Ghadirzadeh, J. Bütepage, A. Maki, D. Kragic, and M. Björkman, "A sensorimotor reinforcement learning framework for physical human-robot interaction," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9-14 October 2016*, pp. 2682–2688, IEEE, 2016.

[10] A. Steinfeld, T. Fong, D. B. Kaber, M. Lewis, J. Scholtz, A. C. Schultz, and M. A. Goodrich, "Common metrics for human-robot interaction," in *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction (HRI), Salt Lake City, Utah, USA, 2-3 March 2006*, pp. 33–40, ACM, 2006.

[11] J. Bütepage and D. Kragic, "Human-robot collaboration: From psychology to social robotics," *CoRR*, vol. abs/1705.10146, 2017.

[12] J. Nelles, S. T. Kwee-Meier, and A. Mertens, "Evaluation metrics regarding human well-being and system performance in human-robot interaction – A literature review," in *Proceedings of the 20th Congress of the International Ergonomics Association (IEA 2018), Florence, Italy, 26-30 August 2018* (S. Bagnara, R. Tartaglia, S. Albolino, T. Alexander, and Y. Fujita, eds.), (Cham), pp. 124–135, Springer International Publishing, 2019.

[13] B. Chandrasekaran and J. M. Conrad, "Human-robot collaboration: A survey," in *SoutheastCon 2015, Fort Lauderdale, FL, USA, 9-12 April 2015*, pp. 1–8, IEEE, 2015.

[14] N. M. Seel, ed., *Encyclopedia of the Sciences of Learning*. Boston: Springer US, 2012.

[15] I. Maurtua, A. Ibarguren, J. Kildal, L. Susperregi, and B. Sierra, "Human-robot collaboration in industrial applications: Safety, interaction and trust," *International Journal of Advanced Robotic Systems*, vol. 14, no. 4, p. 1729881417716010, 2017.

[16] A. Saggese, M. Vento, and V. Vigilante, "Miviabot: A cognitive robot for smart museum," in *Computer Analysis of Images and Patterns (CAIP), Salerno, Italy, 2-6 September 2019* (M. Vento and G. Percannella, eds.), (Cham), pp. 15–25, Springer International Publishing, 2019.

[17] S. Nikolaidis, D. Hsu, and S. Srinivasa, "Human-robot mutual adaptation in collaborative tasks: Models and experiments," *The International Journal of Robotics Research*, vol. 36, no. 5-7, pp. 618–634, 2017.

[18] T. B. Sheridan, "Human-robot interaction: Status and challenges," *Human factors*, vol. 58, no. 4, pp. 525–532, 2016.

[19] T. Moulières-Seban, D. Bitonneau, J. Salotti, J. Thibault, and B. Claverie, "Human factors issues for the design of a cobotic system," in *Advances in Human Factors in Robots and Unmanned Systems, Los Angeles, California, USA, 1721 July 2017* (P. Savage-Knepshield and J. Chen, eds.), (Cham), pp. 375–385, Springer International Publishing, 2017.

[20] H. Yanco and J. Drury, "Classifying human-robot interaction: an updated taxonomy," in *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583), The Hague, Netherlands, 10-13 October 2004*, vol. 3, pp. 2841–2846, IEEE, 2004.

[21] R. Pérula-Martínez, *Autonomous decision-making for socially interactive robots.* PhD thesis, Universidad Carlos III de Madrid. Departamento de Ingeniera de Sistemas y Automtica, 2017.

[22] M. H. D. Zakaria, S. Lengagne, J. A. C. Ramón, and Y. Mezouar, "General framework for the optimization of the human-robot collaboration decision-making process through the ability to change performance metrics," *Frontiers in Robotics and AI*, vol. 8, 2021.

[23] G. Durantin, S. Heath, and J. Wiles, "Social moments: A perspective on interaction for social robotics," *Frontiers in Robotics and AI*, vol. 4, p. 24, 2017.

[24] J. Nowak, P. Fraisse, A. Cherubini, and J. Daurès, "Assistance to older adults with comfortable robot-to-human handovers," in *2022 IEEE International Conference on Advanced Robotics and Its Social Impacts (ARSO), Long Beach, CA, USA, 28-30 May 2022*, pp. 1–6, IEEE, 2022.

[25] V. Wagner-Hartl, T. Gehring, J. Kopp, R. Link, A. Machill, D. Pottin, A. Zitz, and V. E. Gunser, "Who would let a robot take care of them? - gender and age differences," in *HCI International 2020 - Posters, Copenhagen, Denmark, 19-24 July 2020* (C. Stephanidis and M. Antona, eds.), (Cham), pp. 196–202, Springer International Publishing, 2020.

[26] C. Clabaugh, K. Mahajan, S. Jain, R. Pakkar, D. Becerra, Z. Shi, E. Deng, R. Lee, G. Ragusa, and M. Matarić, "Long-term personalization of an in-home socially assistive robot for children with autism spectrum disorders," *Frontiers in Robotics and AI*, vol. 6, p. 110, 2019.

[27] O. Nocentini, L. Fiorini, G. Acerbi, A. Sorrentino, G. Mancioppi, and F. Cavallo, "A survey of behavioral models for social robots," *Robotics*, vol. 8, no. 3, p. 54, 2019.

[28] A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib, "Progress and prospects of the human-robot collaboration," *Autonomous Robots*, vol. 42, no. 5, pp. 957–975, 2018.

[29] M. Flad, J. Otten, S. Schwab, and S. Hohmann, "Steering driver assistance system: A systematic cooperative shared control design approach," in *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), San Diego, CA, USA, 5-8 October 2014*, pp. 3585–3592, IEEE, 2014.

[30] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos, "Revisiting active perception," *Autonomous Robots*, vol. 42, no. 2, pp. 177–196, 2018.

[31] P. Haazebroek, S. van Dantzig, and B. Hommel, "A computational model of perception and action for cognitive robotics," *Cognitive Processing*, vol. 12, no. 4, pp. 355–365, 2011.

[32] J. Mainprice, R. Hayne, and D. Berenson, "Predicting human reaching motion in collaborative tasks using inverse optimal control and iterative re-planning," in *2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26-30 May 2015*, pp. 885–892, IEEE, 2015.

[33] J. Fan, P. Zheng, and S. Li, "Vision-based holistic scene understanding towards proactive human-robot collaboration," *Robotics and Computer-Integrated Manufacturing*, vol. 75, p. 102304, 2022.

[34] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27-30 June 2016*, pp. 5297–5307, IEEE, 2016.

[35] M. S. Bartlett, G. Littlewort, I. R. Fasel, and J. R. Movellan, "Real time face detection and facial expression recognition: Development and applications to human computer interaction," in *2003 Conference on Computer Vision and Pattern Recognition Workshop, Madison, Wisconsin, USA, 16-22 June 2003*, p. 53, IEEE Computer Society, 2003.

[36] S. Wang, D. Tao, and J. Yang, "Relative attribute SVM + learning for age estimation," *IEEE Transactions on Cybernatics*, vol. 46, no. 3, pp. 827–839, 2016.

[37] D. P. Losey, C. G. McDonald, E. Battaglia, and M. K. O'Malley, "A review of intent detection, arbitration, and communication aspects of shared control for physical humanrobot interaction," *Applied Mechanics Reviews*, vol. 70, no. 1, 2018.

[38] F. Alonso-Martín, M. Malfaz, J. a. Sequeira, J. F. Gorostiza, and M. A. Salichs, "A multimodal emotion detection system during human-robot interaction," *Sensors*, vol. 13, no. 11, pp. 15549–15581, 2013.

[39] D. Mukherjee, K. Gupta, L. H. Chang, and H. Najjaran, "A survey of robot learning strategies for human-robot collaboration in industrial settings," *Robotics and Computer-Integrated Manufacturing*, vol. 73, p. 102231, 2022.

[40] R. A. Calvo and S. D'Mello, "Affect detection: An interdisciplinary review of models, methods, and their applications," *IEEE Transactions on Affective Computing*, vol. 1, no. 1, pp. 18–37, 2010.

[41] O. Negulescu, "Using a decision making process model in strategic management," *Review of General Management*, vol. 19, no. 1, 2014.

[42] J. Fülöp, "Introduction to decision making methods," in *BDEI-3 workshop, Washington, USA, 13-15 December 2004*, pp. 1–15, 2005.

[43] N. Jarrassé, V. Sanguineti, and E. Burdet, "Slaves no longer: Review on role assignment for human-robot joint motor action," *Adaptive Behavior*, vol. 22, no. 1, pp. 70–82, 2014.

[44] M. Faroni, M. Beschi, A. Visioli, and N. Pedrocchi, "A real-time trajectory planning method for enhanced path-tracking performance of serial manipulators," *Mechanism and Machine Theory*, vol. 156, p. 104152, 2021.

[45] A. De Santis, B. Siciliano, A. De Luca, and A. Bicchi, "An atlas of physical human-robot interaction," *Mechanism and Machine Theory*, vol. 43, no. 3, pp. 253–270, 2008.

[46] V. Duchaine and C. M. Gosselin, "General model of human-robot cooperation using a novel velocity based variable impedance control," in *Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'07), Tsukuba, Japan, 22-24 March 2007*, pp. 446–451, IEEE, 2007.

[47] G. Hoffman, "Evaluating fluency in human-robot collaboration," *IEEE Transactions on Human-Machine Systems*, vol. 49, no. 3, pp. 209–218, 2019.

[48] R. Gervasi, L. Mastrogiacomo, and F. Franceschini, "A conceptual framework to evaluate human-robot collaboration," *The International Journal of Advanced Manufacturing Technology*, vol. 108, pp. 841–865, 2020.

[49] M. T. J. Spaan, T. S. Veiga, and P. U. Lima, "Decision-theoretic planning under uncertainty with information rewards for active cooperative perception," *Autonomous Agents and Multi-Agent Systems*, vol. 29, no. 6, pp. 1157–1185, 2014.

[50] M. Chen, S. Nikolaidis, H. Soh, D. Hsu, and S. Srinivasa, "Trust-aware decision making for human-robot collaboration: Model learning and planning," *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 9, no. 2, pp. 1–23, 2020.

[51] J. Mainprice and D. Berenson, "Human-robot collaborative manipulation planning using early prediction of human motion," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3-7 November 2013*, pp. 299–306, IEEE, 2013.

[52] M. U. Gutmann, J. Corander, *et al.*, "Bayesian optimization for likelihood-free inference of simulator-based statistical models," *Journal of Machine Learning Research*, 2016.

[53] L. Roveda, M. Magni, M. Cantoni, D. Piga, and G. Bucca, "Human-robot collaboration in sensorless assembly task learning enhanced by uncertainties adaptation via bayesian optimization," *Robotics and Autonomous Systems*, vol. 136, p. 103711, 2021.

[54] D. L. Poole and A. K. Mackworth, *Artificial Intelligence: foundations of computational agents*. Cambridge University Press, 2010.

[55] A. Howes, X. Chen, A. Acharya, and R. L. Lewis, "Interaction as an emergent property of a partially observable markov decision process," in *Computational Interaction*, pp. 287–310, Oxford University Press, 2018.

[56] A. Lederer, A. J. O. Conejo, K. Maier, W. Xiao, J. Umlauft, and S. Hirche, "Gaussian process-based real-time learning for safety critical applications," in *Proceedings of the 38th International Conference on Machine Learning (ICML), Virtual Event, 18-24 July 2021* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 6055–6064, PMLR, 2021.

[57] A. A. Chehade and A. A. Hussein, "A collaborative gaussian process regression model for transfer learning of capacity trends between li-ion battery cells," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 9542–9552, 2020.

[58] J. Mockus, "Application of bayesian approach to numerical methods of global and stochastic optimization," *Journal of Global Optimization*, vol. 4, pp. 347–365, 1994.

[59] T. Denoeux, "Decision-making with belief functions: A review," *International Journal of Approximate Reasoning*, vol. 109, pp. 87–110, 2019.

[60] T. Zhou and J. P. Wachs, "Early prediction for physical human robot collaboration in the operating room," *Autonomous Robots*, vol. 42, no. 5, pp. 977–995, 2018.

[61] X. Yu, W. He, Y. Li, C. Xue, J. Li, J. Zou, and C. Yang, "Bayesian estimation of human impedance and motion intention for humanrobot collaboration," *IEEE Transactions on Cybernetics*, vol. 51, no. 4, pp. 1822–1834, 2021.

[62] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[63] S. Wang, T. Zhou, and J. Bilmes, "Bias also matters: Bias attribution for deep neural network explanation," in *Proceedings of the 36th International Conference on Machine Learning (ICML), Long Beach, California, USA, 9-15 June 2019* (K. Chaudhuri and

R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 6659–6667, PMLR, 2019.

[64] C. Bircanoğlu and N. Arca, "A comparison of activation functions in artificial neural networks," in *26th Signal Processing and Communications Applications Conference (SIU), Izmir, Turkey, 2-5 May 2018*, pp. 1–4, IEEE, 2018.

[65] J. M. Górriz, J. Ramírez, A. Ortíz, F. J. Martínez-Murcia, F. Segovia, J. Suckling, M. Leming, Y.-D. Zhang, J. R. Álvarez-Sánchez, G. Bologna, P. Bonomini, F. E. Casado, D. Charte, F. Charte, R. Contreras, A. Cuesta-Infante, R. J. Duro, A. Fernández-Caballero, E. Fernández-Jover, P. Góme-Vilda, M. Grana, F. Herrera, R. Iglesias, A. Lekova, J. de Lope, E. López-Rubio, R. Martínez-Tomás, M. A. Molina-Cabello, A. S. Montemayor, P. Novais, D. Palacios-Alonso, J. J. Pantrigo, B. R. Payne, F. de la Paz López, M. A. Pinninghoff, M. Rincón, J. Santos, K. Thurnhofer-Hemsi, A. Tsanas, R. Varela, and J. M. Ferrández, "Artificial intelligence within the interplay between natural and artificial computation: Advances in data science, trends and applications," *Neurocomputing*, vol. 410, pp. 237–270, 2020.

[66] B. Mahesh, "Machine learning algorithms - A review," *International Journal of Science and Research (IJSR)*, vol. 9, pp. 381–386, 2020.

[67] T. Hastie, R. Tibshirani, and J. Friedman, *Overview of Supervised Learning*, pp. 9–41. New York, NY: Springer New York, 2009.

[68] L. Rokach and O. Maimon, *Supervised Learning*, pp. 133–147. Boston, MA: Springer US, 2010.

[69] D. Shukla, Ö. Erkent, and J. H. Piater, "Supervised learning of gesture-action associations for human-robot collaboration," in *12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), Washington, DC, USA, 30 May - 3 June 2017*, pp. 778–783, IEEE Computer Society, 2017.

[70] B. Rajoub, "Chapter 3 - supervised and unsupervised learning," in *Biomedical Signal Processing and Artificial Intelligence in Healthcare*, Developments in Biomedical Engineering and Bioelectronics, pp. 51–89, Academic Press, 2020.

[71] Z. Ghahramani, "Unsupervised learning," in *Advanced Lectures on Machine Learning, ML Summer Schools 2003, Canberra, Australia, 2-14 February 2003, Tübingen, Germany, 4-16 August 2003, Revised Lectures*, vol. 3176 of *Lecture Notes in Computer Science*, pp. 72–112, Springer, 2003.

[72] R. Luo, R. Hayne, and D. Berenson, "Unsupervised early prediction of human reaching for human-robot collaboration in shared workspaces," *Autonomous Robots*, vol. 42, no. 3,

pp. 631–648, 2018.

[73] R. Zhang, Q. Lv, J. Li, J. Bao, T. Liu, and S. Liu, "A reinforcement learning method for human-robot collaboration in assembly tasks," *Robotics and Computer-Integrated Manufacturing*, vol. 73, p. 102227, 2022.

[74] L. Roveda, J. Maskani, P. Franceschi, A. Abdi, F. Braghin, L. M. Tosatti, and N. Pedrocchi, "Model-based reinforcement learning variable impedance control for human-robot collaboration," *Journal of Intelligent & Robotic Systems*, vol. 100, no. 2, pp. 417–433, 2020.

[75] H. Bhavsar and A. Ganatra, "A comparative study of training algorithms for supervised machine learning," *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 2, no. 4, pp. 2231–2307, 2012.

[76] H. U. Dike, Y. Zhou, K. K. Deveerasetty, and Q. Wu, "Unsupervised learning based on artificial neural network: A review," in *IEEE International Conference on Cyborg and Bionic Systems (CBS), Shenzhen, China, 25-27 October 2018*, pp. 322–327, IEEE, 2018.

[77] M. A. Wiering and M. van Otterlo, *Reinforcement Learning*, vol. 12 of *Adaptation, Learning, and Optimization*. Springer, 2012.

[78] A. Taylor, I. Dusparic, E. G. López, S. Clarke, and V. Cahill, "Accelerating learning in multi-objective systems through transfer learning," in *2014 International Joint Conference on Neural Networks (IJCNN), Beijing, China, 6-11 July 2014*, pp. 2298–2305, IEEE, 2014.

[79] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, Australia, 1-4 December 2020*, pp. 737–744, IEEE, 2020.

[80] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[81] S. Li, P. Zheng, J. Fan, and L. Wang, "Toward proactive humanrobot collaborative assembly: A multimodal transfer-learning-enabled action prediction approach," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 8, pp. 8579–8588, 2022.

[82] J. Hua, L. Zeng, G. Li, and Z. Ju, "Learning for a robot: Deep reinforcement learning, imitation learning, transfer learning," *Sensors*, vol. 21, no. 4, p. 1278, 2021.

[83] R. Liu, F. Nageotte, P. Zanne, M. de Mathelin, and B. Dresp-Langley, "Deep reinforcement learning for the control of robotic manipulation: A focussed mini-review," *Robotics*, vol. 10, no. 1, p. 22, 2021.

[84] Y. Gu, A. Thobbi, and W. Sheng, "Human-robot collaborative manipulation through imitation and reinforcement learning," in *2011 IEEE International Conference on Information and Automation, Shenzhen, China, 6-8 June 2011*, pp. 151–156, IEEE, 2011.

[85] R. M. Kretchmar, "Parallel reinforcement learning," in *The 6th World Conference on Systemics, Cybernetics, and Informatics, Orlando, Florida, USA, 14-18 July 2002*, 2002.

[86] A. V. Clemente, H. N. C. Martínez, and A. Chandra, "Efficient parallel methods for deep reinforcement learning," *CoRR*, vol. abs/1705.04862, 2017.

[87] M. Hani Daniel Zakaria, M. Aranda, L. Lequièvre, S. Lengagne, J. A. Corrales Ramón, and Y. Mezouar, "Robotic control of the deformation of soft linear objects using deep reinforcement learning," in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE), Mexico City, Mexico, 20-24 August 2022*, 2022.

[88] T. Yu, J. Huang, and Q. Chang, "Optimizing task scheduling in human-robot collaboration with deep multi-agent reinforcement learning," *Journal of Manufacturing Systems*, vol. 60, pp. 487–499, 2021.

[89] E. Salvato, G. Fenu, E. Medvet, and F. A. Pellegrino, "Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning," *IEEE Access*, vol. 9, pp. 153171–153187, 2021.

[90] J. Matas, S. James, and A. J. Davison, "Sim-to-real reinforcement learning for deformable object manipulation," in *2nd Annual Conference on Robot Learning (CoRL), Zürich, Switzerland, 29-31 October 2018, Proceedings*, vol. 87 of *Proceedings of Machine Learning Research*, pp. 734–743, PMLR, 2018.

[91] D. Zhang, Z. Wu, J. Chen, R. Zhu, A. Munawar, B. Xiao, Y. Guan, H. Su, W. Hong, Y. Guo, G. S. Fischer, B. Lo, and G. Yang, "Human-robot shared control for surgical robot based on context-aware sim-to-real adaptation," in *2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23-27 May 2022*, pp. 7694–7700, IEEE, 2022.

[92] N. Agarwal, A. Sondhi, K. Chopra, and G. Singh, "Transfer learning: Survey and classification," in *Smart Innovations in Communication and Computational Sciences* (S. Tiwari, M. C. Trivedi, K. K. Mishra, A. Misra, K. K. Kumar, and E. Suryani, eds.), (Singapore), pp. 145–155, Springer Singapore, 2021.

[93] X. Zhu and Y. Du, "A parallel approach to advantage actor critic in deep reinforcement learning," in *Algorithms and Architectures for Parallel Processing* (S. Wen, A. Zomaya, and L. T. Yang, eds.), (Cham), pp. 320–327, Springer International Publishing, 2020.

[94] N. Geetha and P. Sekar, "Graph theory matrix approach a qualitative decision making tool," *Materials Today: Proceedings*, vol. 4, no. 8, pp. 7741–7749, 2017. International Conference on Advancements in Aeromechanical Materials for Manufacturing (ICAAMM-2016): Organized by MLR Institute of Technology, Hyderabad, Telangana, India.

[95] C. Di Marino, A. Rega, F. Vitolo, S. Patalano, and A. Lanzotti, "A new approach to the anthropocentric design of human-robot collaborative environments," *Acta Imeko*, vol. 9, no. 4, pp. 80–87, 2020.

[96] C. A. Kamhoua, C. D. Kiekintveld, F. Fang, and Q. Zhu, *Game theory and machine learning for cyber security*. Hoboken, NJ, USA: Wiley, 2021.

[97] V. Conitzer and T. Sandholm, "Computing the optimal strategy to commit to," in *Proceedings 7th ACM Conference on Electronic Commerce (EC-2006), Ann Arbor, Michigan, USA, 11-15 June 2006* (J. Feigenbaum, J. C. Chuang, and D. M. Pennock, eds.), pp. 82–90, ACM, 2006.

[98] K. Leyton-Brown and Y. Shoham, "Essentials of game theory: A concise multidisciplinary introduction," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 2, no. 1, pp. 1–88, 2008.

[99] N. Jarrassé, T. Charalambous, and E. Burdet, "A framework to describe, analyze and generate interactive motor behaviors," *PLOS ONE*, vol. 7, no. 11, pp. 1–13, 2012.

[100] Y. Li, G. Carboni, F. Gonzalez, D. Campolo, and E. Burdet, "Differential game theory for versatile physical human-robot interaction," *Nature Machine Intelligence*, vol. 1, no. 1, pp. 36–43, 2019.

[101] V. Gabler, T. Stahl, G. Huber, O. Oguz, and D. Wollherr, "A game-theoretic approach for adaptive action selection in close proximity human-robot-collaboration," in *2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May - 3 June 2017*, pp. 2897–2903, IEEE, 2017.

[102] M. Aruldoss, T. M. Lakshmi, and V. P. Venkatesan, "A survey on multi criteria decision making methods and its applications," *American Journal of Information Systems*, vol. 1, no. 1, pp. 31–43, 2013.

[103] S. Heydaryan, J. Suaza Bedolla, and G. Belingardi, "Safety design and development of a human-robot collaboration assembly process in the automotive industry," *Applied Sciences*, vol. 8, no. 3, p. 344, 2018.

[104] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, "Chapter 10 - metaheuristic algorithms: A comprehensive review," in *Computational Intelligence for Multimedia Big*

Data on the Cloud with Engineering Applications (A. K. Sangaiah, M. Sheng, and Z. Zhang, eds.), Intelligent Data-Centric Systems, pp. 185–231, Academic Press, 2018.

[105] K. Sörensen and F. Glover, "Metaheuristics," *Encyclopedia of Operations Research and Management Science*, vol. 62, pp. 960–970, 2013.

[106] İ. H. Boyacı, "A new approach for determination of enzyme kinetic constants using response surface methodology," *Biochemical Engineering Journal*, vol. 25, no. 1, pp. 55–62, 2005.

[107] A. Y. Aydar, "Utilization of response surface methodology in optimization of extraction of plant materials," in *Statistical Approaches With Emphasis on Design of Experiments Applied to Chemical Processes*, pp. 157–169, InTech London, UK, 2018.

[108] B. Koç and F. Kaymak-Ertekin, "Response surface methodology and food processing applications," *GIDA-Journal of Food*, vol. 35, no. 1, pp. 63–70, 2010.

[109] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, "Chapter 10 - metaheuristic algorithms: A comprehensive review," in *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications* (A. K. Sangaiah, M. Sheng, and Z. Zhang, eds.), Intelligent Data-Centric Systems, pp. 185–231, Academic Press, 2018.

[110] Z. A. Çil, Z. Li, S. Mete, and E. Özceylan, "Mathematical model and bee algorithms for mixed-model assembly line balancing problem with physical human-robot collaboration," *Applied Soft Computing*, vol. 93, p. 106394, 2020.

[111] A. Nourmohammadi, M. Fathi, and A. H. Ng, "Balancing and scheduling assembly lines with human-robot collaboration tasks," *Computers & Operations Research*, vol. 140, p. 105674, 2022.

[112] A. I. Khuri and S. Mukhopadhyay, "Response surface methodology," *WIREs Computational Statistics*, vol. 2, no. 2, pp. 128–149, 2010.

[113] Y. Park, Y. Choi, H. S. Yang, and Y. Seo, "Response surface learning for face misalignment correction," in *2014 Ninth International Conference on Broadband and Wireless Computing, Communication and Applications, Guangdong, China, 8-10 November 2014*, pp. 549–553, IEEE, 2014.

[114] X. Xing, J. Xia, D. Huang, and Y. Li, "Path learning in human-robot collaboration tasks using iterative learning methods," *IEEE Transactions on Control Systems Technology*, vol. 30, no. 5, pp. 1946–1959, 2022.

[115] S. Bouwman, G. A. Bloemhof, J. F. L. van Casteren, and B. Taks, "Advantages of probabilistic system analysis," in *CIRED 2005 - 18th International Conference and Exhibition on Electricity Distribution, Turin, Italy, 6-9 June 2005*, pp. 1–4, IEEE, 2005.

[116] B. Kamiński, M. Jakubczyk, and P. Szufel, "A framework for sensitivity analysis of decision trees," *Central European Journal of Operations Research*, vol. 26, no. 1, pp. 135–159, 2018.

[117] M. Velasquez and P. T. Hester, "An analysis of multi-criteria decision making methods," *International Journal of Operations Research*, vol. 10, no. 2, pp. 56–66, 2013.

[118] H. Rommelfanger, "The advantages of fuzzy optimization models in practical use," *Fuzzy Optimization and Decision Making*, vol. 3, no. 4, pp. 295–309, 2004.

[119] L. Xiujuan and S. Zhongke, "Overview of multi-objective optimization methods," *Journal of Systems Engineering and Electronics*, vol. 15, no. 2, pp. 142–146, 2004.

[120] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and Systems Magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[121] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, 2018.

[122] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, no. 2, pp. 241–258, 2020.

[123] J. Nagi, H. Q. Ngo, L. M. Gambardella, and G. A. D. Caro, "Wisdom of the swarm for cooperative decision-making in human-swarm interaction," in *IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26-30 May 2015*, pp. 1802–1808, IEEE, 2015.

[124] F. Chao, Y. Sun, Z. Wang, G. Yao, Z. Zhu, C. Zhou, Q. Meng, and M. Jiang, "A reduced classifier ensemble approach to human gesture classification for robotic chinese handwriting," in *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Beijing, China, 6-11 July 2014*, pp. 1720–1727, IEEE, 2014.

[125] L. A. T. Cox, Jr., "Game theory and risk analysis," *Risk Analysis*, vol. 29, no. 8, pp. 1062–1068, 2009.

[126] D. M. Kreps, *Nash Equilibrium*, pp. 167–177. London: Palgrave Macmillan UK, 1989.

[127] Y. Zhou, Y. Peng, W. Li, and D. T. Pham, "Stackelberg model-based human-robot collaboration in removing screws for product remanufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 77, p. 102370, 2022.

[128] L. Fei, J. Xia, Y. Feng, and L. Liu, "An electre-based multiple criteria decision making method for supplier selection using dempster-shafer theory," *IEEE Access*, vol. 7, pp. 84701–84716, 2019.

[129] Z. Wu and G. Abdul-Nour, "Comparison of multi-criteria group decision-making methods for urban sewer network plan selection," *CivilEng*, vol. 1, no. 1, pp. 26–48, 2020.

[130] Ö. Şimşek, *Lexicographic decision rule*. UK United Kingdom: Oxford University Press, Mar. 2021.

[131] J. Benediktsson and P. Swain, "Consensus theoretic classification methods," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 4, pp. 688–704, 1992.

[132] G. P. Huber, "Multi-attribute utility models: A review of field and field-like studies," *Management Science*, vol. 20, no. 10, pp. 1393–1402, 1974.

[133] J. Reinhardt, A. Pereira, D. Beckert, and K. Bengler, "Dominance and movement cues of robot motion: A user study on trust and predictability," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB, Canada, 5-8 October 2017*, IEEE, 2017.

[134] S. Lo, B. Fernandez, P. Stone, and A. L. Thomaz, "Towards safe motion planning in human workspaces: A robust multi-agent approach," in *2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May - 5 June 2021*, pp. 7929–7935, IEEE, 2021.

[135] J. Deng, G. Pang, Z. Zhang, Z. Pang, H. Yang, and G. Yang, "cgan based facial expression recognition for human-robot interaction," *IEEE Access*, vol. 7, pp. 9848–9859, 2019.

[136] W. Xu, Q. Tang, J. Liu, Z. Liu, Z. Zhou, and D. T. Pham, "Disassembly sequence planning using discrete bees algorithm for human-robot collaboration in remanufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 62, p. 101860, 2020.

[137] S. Bansal, J. Xu, A. M. Howard, and C. Isbell, "A bayesian framework for nash equilibrium inference in human-robot parallel play," in *Robotics: Science and Systems XVI, Virtual Event, Corvalis, Oregon, USA, 12-16 July 2020* (M. Toussaint, A. Bicchi, and T. Hermans, eds.), 2020.

[138] N. Li, D. W. Oyler, M. Zhang, Y. Yildiz, I. Kolmanovsky, and A. R. Girard, "Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 5, pp. 1782–1797, 2018.

[139] X. Na and D. J. Cole, "Game-theoretic modeling of the steering interaction between a human driver and a vehicle collision avoidance controller," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 1, pp. 25–38, 2015.

[140] R. Gervasi, L. Mastrogiacomo, D. A. Maisano, D. Antonelli, and F. Franceschini, "A structured methodology to support human–robot collaboration configuration choice," *Production Engineering*, vol. 16, no. 4, pp. 435–451, 2022.

[141] S. Parsa and M. Saadat, "Human-robot collaboration disassembly planning for end-of-life product disassembly process," *Robotics and Computer-Integrated Manufacturing*, vol. 71, p. 102170, 2021.

[142] B. Abbasi, E. Noohi, S. Parastegari, and M. Zefran, "Understanding of object manipulation actions using human multi-modal sensory data," *CoRR*, vol. abs/1905.07012, 2019.

[143] T. Bressen, "Consensus decision making," *The change handbook: The definitive resource on today's best methods for engaging whole systems*, vol. 495, pp. 212–217, 2007.

[144] C. Shi, M. Shiomi, T. Kanda, H. Ishiguro, and N. Hagita, "Measuring communication participation to initiate conversation in human-robot interaction," *International Journal of Social Robotics*, vol. 7, no. 5, pp. 889–910, 2015.

[145] R. Sukkerd, R. Simmons, and D. Garlan, "Toward explainable multi-objective probabilistic planning," in *2018 IEEE/ACM 4th International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS), Gothenburg, Sweden, 27 May - 3 June 2018*, pp. 19–25, IEEE, 2018.

[146] M. Kwon, M. Li, A. Bucquet, and D. Sadigh, "Influencing leading and following in human-robot teams," in *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, 22-26 June 2019*, 2019.

[147] S. Nikolaidis, S. Nath, A. D. Procaccia, and S. Srinivasa, "Game-theoretic modeling of human adaptation in human-robot collaboration," in *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction (HRI), Vienna, Austria, 6-9 March 2017*, pp. 323–331, ACM, 2017.

[148] M. Chen, S. Nikolaidis, H. Soh, D. Hsu, and S. Srinivasa, "Trust-aware decision making for human-robot collaboration: Model learning and planning," *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 9, no. 2, pp. 1–23, 2020.

[149] A. A. Malik and A. Bilberg, "Complexity-based task allocation in human-robot collaborative assembly," *Industrial Robot*, vol. 46, no. 4, pp. 471–180, 2019.

[150] A. Cherubini, R. Passama, A. Crosnier, A. Lasnier, and P. Fraisse, "Collaborative manufacturing with physical human-robot interaction," *Robotics and Computer-Integrated Manufacturing*, vol. 40, pp. 1–13, 2016.

[151] A. Sharkawy, C. Papakonstantinou, V. Papakostopoulos, V. C. Moulianitis, and N. A. Aspragathos, "Task location for high performance human-robot collaboration," *Journal of Intelligent & Robotic Systems*, vol. 100, no. 1, pp. 183–202, 2020.

[152] R. Weitschat and H. Aschemann, "Safe and efficient human-robot collaboration part II: Optimal generalized human-in-the-loop real-time motion generation," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3781–3788, 2018.

[153] A. Fishman, C. Paxton, W. Yang, D. Fox, B. Boots, and N. Ratliff, "Collaborative interaction models for optimized human-robot teamwork," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020 - 24 January 2021*, pp. 11221–11228, IEEE, 2020.

[154] A. Tanevska, F. Rea, G. Sandini, L. Caamero, and A. Sciutti, "A socially adaptable framework for human-robot interaction," *Frontiers in Robotics and AI*, vol. 7, p. 121, 2020.

[155] A. Tabrez and B. Hayes, "Improving human-robot interaction through explainable reinforcement learning," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI), Daegu, Korea (South), 11-14 March 2019*, pp. 751–753, IEEE, 2019.

[156] Z. Liu, Q. Liu, W. Xu, Z. Zhou, and D. T. Pham, "Human-robot collaborative manufacturing using cooperative game: Framework and implementation," *Procedia CIRP*, vol. 72, pp. 87–92, 2018. 51st CIRP Conference on Manufacturing Systems.

[157] N. J. Delleman and J. Dul, "International standards on working postures and movements ISO 11226 and EN 1005-4," *Ergonomics*, vol. 50, no. 11, pp. 1809–1819, 2007.

[158] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, "Bridging the gap between value and policy based reinforcement learning," in *Advances in Neural Information Processing Systems 30: 2017 Annual Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4-9 December 2017*, pp. 2775–2785, Curran Associates, Inc., 2017.

[159] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[160] H. Yin, A. Varava, and D. Kragic, "Modeling, learning, perception, and control methods for deformable object manipulation," *Science Robotics*, vol. 6, no. 54, p. 8803, 2021.

[161] R. Laezza and Y. Karayiannidis, "Learning shape control of elastoplastic deformable linear objects," in *2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May - 5 June 2021*, pp. 4438–4444, IEEE, 2021.

[162] M. Shetab-Bushehri, M. Aranda, Y. Mezouar, and E. Özgür, "As-rigid-as-possible shape servoing," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3898–3905, 2022.

[163] D. Kruse, R. J. Radke, and J. T. Wen, "Human-robot collaborative handling of highly deformable materials," in *2017 American Control Conference (ACC), Seattle, WA, USA, 24-26 May 2017*, pp. 1511–1516, IEEE, 2017.

[164] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: A survey," *Artificial Intelligence Review*, vol. 55, no. 2, pp. 895–943, 2022.

[165] W. Du and S. Ding, "A survey on multi-agent deep reinforcement learning: From the perspective of challenges and applications," *Artificial Intelligence Review*, vol. 54, no. 5, pp. 3215–3238, 2021.

[166] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *CoRR*, vol. abs/1511.08779, 2015.

[167] J. Sanchez, J. A. Corrales Ramón, B. Bouzgarrou, and Y. Mezouar, "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: A survey," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 688–716, 2018.

[168] J. Zhu, B. Navarro, P. Fraisse, A. Crosnier, and A. Cherubini, "Dual-arm robotic manipulation of flexible cables," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1-5 October 2018*, pp. 479–484, IEEE, 2018.

[169] R. Lagneau, A. Krupa, and M. Marchal, "Automatic shape control of deformable wires based on model-free visual servoing," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5252–5259, 2020.

[170] Y. Gao, Z. Chen, M. Fang, Y. Liu, and X. Li, "Development of an autonomous soldering robot for USB wires," in *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Boston, MA, USA, 6-9 July 2020*, pp. 1196–1201, IEEE, 2020.

[171] T. Botterill, S. Paulin, R. Green, S. Williams, J. Lin, V. Saxton, S. Mills, X. Chen, and S. Corbett-Davies, "A robot system for pruning grape vines," *Journal of Field Robotics*, vol. 34, no. 6, pp. 1100–1122, 2017.

[172] O. Aghajanzadeh, M. Aranda, J. A. Corrales Ramón, C. Cariou, R. Lenain, and Y. Mezouar, "Adaptive deformation control for elastic linear objects," *Frontiers in Robotics and AI*, vol. 9, 2022.

[173] R. C. Jackson and M. C. Cavusoglu, "Needle path planning for autonomous robotic surgical suturing," in *2013 IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 6-10 May 2013*, pp. 1669–1675, IEEE, 2013.

[174] M. Saha and P. Isto, "Manipulation planning for deformable linear objects," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1141–1150, 2007.

[175] H. Han, G. Paul, and T. Matsubara, "Model-based reinforcement learning approach for deformable linear object manipulation," in *2017 13th IEEE Conference on Automation Science and Engineering (CASE), Xi'an, China, 20-23 August 2017*, pp. 750–755, 2017.

[176] J. Zhu, A. Cherubini, C. Dune, D. Navarro-Alarcon, F. Alambeigi, D. Berenson, F. Ficuciello, K. Harada, J. Kober, X. Li, J. Pan, W. Yuan, and M. Gienger, "Challenges and outlook in robotic manipulation of deformable objects," *IEEE Robotics & Automation Magazine*, vol. 29, no. 3, pp. 67–77, 2022.

[177] T. Bretl and Z. McCarthy, "Quasi-static manipulation of a Kirchhoff elastic rod based on a geometric analysis of equilibrium configurations," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 48–68, 2014.

[178] M. Mukadam and A. B. andTimothy Bretl, "Quasi-static manipulation of a planar elastic rod using multiple robotic grippers," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Chicago, IL, USA, 14-18 September 2014*, pp. 55–60, IEEE, 2014.

[179] P. Zhou, J. Zhu, S. Huo, and D. Navarro-Alarcon, "Lasesom: A latent and semantic representation framework for soft object manipulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5381–5388, 2021.

[180] N. Lv, J. Liu, and Y. Jia, "Dynamic modeling and control of deformable linear objects for single-arm and dual-arm robot manipulations," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2341–2353, 2022.

[181] A. Sintov, S. Macenski, A. Borum, and T. Bretl, "Motion planning for dual-arm manipulation of elastic rods," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6065–6072, 2020.

[182] Y. Li, Y. Wang, Y. Yue, D. Xu, M. Case, S.-F. Chang, E. Grinspun, and P. K. Allen, "Model-driven feedforward prediction for manipulation of deformable objects," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1621–1638, 2018.

[183] J. Qi, G. Ma, J. Zhu, P. Zhou, Y. Lyu, H. Zhang, and D. Navarro-Alarcon, "Contour moments based manipulation of composite rigid-deformable objects with finite time model estimation and shape/position control," *IEEE/ASME Transactions on Mechatronics*, pp. 1–12, 2021.

[184] A. Koessler, N. R. Filella, B. C. Bouzgarrou, L. Lequièvre, and J. A. C. Ramón, "An efficient approach to closed-loop shape control of deformable objects using finite element models," in *2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May - 5 June 2021*, pp. 1637–1643, IEEE, 2021.

[185] O. Aghajanzadeh, G. Picard, J. A. Corrales Ramón, C. Cariou, R. Lenain, and Y. Mezouar, "Optimal Deformation Control Framework for Elastic Linear Objects," in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE), Mexico City, Mexico, 20-24 August 2022*, IEEE, 2022.

[186] O. Aghajanzadeh, M. Aranda, G. López-Nicolás, R. Lenain, and Y. Mezouar, "An offline geometric model for controlling the shape of elastic linear objects," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 2327 October 2022*, IEEE, 2022.

[187] S. Caro, C. Chevallereau, and A. Remus, "Manipulating deformable objects with a dual-arm robot," in *Proceedings of the 2nd International Conference on Robotics, Computer Vision and Intelligent Systems (ROBOVIS), Online Streaming, 27-28 October 2021* (P. Galambos and E. Kayacan, eds.), pp. 48–56, SCITEPRESS, 2021.

[188] D. Navarro-Alarcon, H. M. Yip, Z. Wang, Y. Liu, F. Zhong, T. Zhang, and P. Li, "Automatic 3-D manipulation of soft objects by robotic arms with an adaptive deformation model," *IEEE Transactions on Robotics*, vol. 32, no. 2, pp. 429–441, 2016.

[189] R. Jangir, G. Alenyà, and C. Torras, "Dynamic cloth manipulation with deep reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May - 31 August 2020*, pp. 4630–4636, IEEE, 2020.

[190] Z. Hou, Z. Li, C. Hsu, K. Zhang, and J. Xu, "Fuzzy logic-driven variable time-scale prediction-based reinforcement learning for robotic multiple peg-in-hole assembly," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 1, pp. 218–229, 2022.

[191] L. Leyendecker, M. Schmitz, H. A. Zhou, V. Samsonov, M. Rittstieg, and D. Lütticke, "Deep reinforcement learning for robotic control in high-dexterity assembly tasks - a reward curriculum approach," in *2021 Fifth IEEE International Conference on Robotic Computing (IRC), Taichung, Taiwan, 15-17 November 2021*, pp. 35–42, IEEE, 2021.

[192] A. Verleysen, T. Holvoet, R. Proesmans, C. Den Haese, and F. wyffels, "Simpler learning of robotic manipulation of clothing by utilizing DIY smart textile technology," *Applied Sciences*, vol. 10, no. 12, 2020.

[193] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel, "Learning to manipulate deformable objects without demonstrations," in *Robotics: Science and Systems XVI, Virtual Event / Corvalis, Oregon, USA, 12-16 July 2020* (M. Toussaint, A. Bicchi, and T. Hermans, eds.), 2020.

[194] M. Vecerík, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. A. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *CoRR*, vol. abs/1707.08817, 2017.

[195] K. Suzuki, M. Kanamura, Y. Suga, H. Mori, and T. Ogata, "In-air knotting of rope using dual-arm robot based on deep learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September - 1 October 2021*, pp. 6724–6731, IEEE, 2021.

[196] L. Pecyna, S. Dong, and S. Luo, "Visual-tactile multimodality for following deformable linear objects using reinforcement learning," *CoRR*, vol. abs/2204.00117, 2022.

[197] D. Seita, P. Florence, J. Tompson, E. Coumans, V. Sindhwani, K. Goldberg, and A. Zeng, "Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks," in *IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May - 5 June 2021*, pp. 4568–4575, IEEE, 2021.

[198] X. Lin, Y. Wang, J. Olkin, and D. Held, "Softgym: Benchmarking deep reinforcement learning for deformable object manipulation," in *2020 4th Conference on Robot Learning (CoRL), Virtual Event, Cambridge, MA, USA, 16-18 November 2020*, vol. 155 of *Proceedings of Machine Learning Research*, pp. 432–448, PMLR, 2020.

[199] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," *CoRR*, vol. abs/1606.01540, 2016.

[200] E. Curto and H. Araújo, "3D reconstruction of deformable objects from RGB-D cameras: An omnidirectional inward-facing multi-camera system," in *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2021, Volume 4: VISAPP, Online Streaming, 8-10 February 2021* (G. M. Farinella, P. Radeva, J. Braz, and K. Bouatouch, eds.), pp. 544–551, SCITEPRESS, 2021.

[201] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),*

Vilamoura, Algarve, Portugal, 7-12 October 2012, pp. 5026–5033, IEEE, 2012.

[202] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning." `http://pybullet.org`, 2016–2021.

[203] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *2016 4th International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2-4 May 2016, Conference Track Proceedings*, 2016.

[204] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[205] R. Liu and J. Zou, "The effects of memory replay in reinforcement learning," in *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, USA, 2-5 October 2018*, pp. 478–485, IEEE, 2018.

[206] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *2015 3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7-9 May 2015, Conference Track Proceedings*, 2015.

[207] A. Sehgal, H. La, S. Louis, and H. Nguyen, "Deep reinforcement learning using genetic algorithm for parameter optimization," in *2019 Third IEEE International Conference on Robotic Computing (IRC), Naples, Italy, 25-27 February 2019*, pp. 596–601, IEEE, 2019.

[208] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, V. Kumar, and W. Zaremba, "Multi-goal reinforcement learning: Challenging robotics environments and request for research," *CoRR*, vol. abs/1802.09464, 2018.

[209] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *2016 Proceedings of the 33nd International Conference on Machine Learning (ICML), New York City, NY, USA, 19-24 June 2016*, vol. 48 of *JMLR Workshop and Conference Proceedings*, pp. 1928–1937, JMLR.org, 2016.

[210] I. Cuiral-Zueco, G. López-Nicolás, and H. Araújo, "Gripper positioning for object deformation tasks," in *2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23-27 May 2022*, pp. 963–969, IEEE, 2022.

[211] L. Dalcín and Y. L. Fang, "mpi4py: Status update after 12 years of development," *Computing in Science Engineering*, vol. 23, no. 4, pp. 47–54, 2021.

[212] O. C. Zienkiewicz and P. Morice, *The finite element method in engineering science*, vol. 1977. MCgraw-hill London, 1971.

[213] T. Chakraborty, M. Larcher, and N. Gebbeken, "Performance of tunnel lining materials under internal blast loading," *International Journal of Protective Structures*, vol. 5, no. 1, pp. 83–96, 2014.

[214] M. Çolakoğlu, "Damping and vibration analysis of polyethylene fiber composite under varied temperature," *Turkish Journal of Engineering and Environmental Sciences*, vol. 30, no. 6, pp. 351–357, 2006.