Development of an Intelligent Robotic System for Rehabilitation of Upper Limbs Using a Collaborative Robot

LUCAS DE AZEVEDO FERNANDES

Supervised by

Prof. Dr. José Luis de Sousa Magalhães Lima

Prof. Dr. Alberto Yoshihiro Nakano

Prof. Dr. Paulo Leitão

Bragança

2018-2019

Development of an Intelligent Robotic System for Rehabilitation of Upper Limbs Using a Collaborative Robot

LUCAS DE AZEVEDO FERNANDES

Thesis presented in the School of Technology and Management of the Polytechnic Institute of Bragança to fulfill the requirements of a Master of Science Degree in Industrial Engineering (Electrical Engineering branch).

Supervised by

Prof. Dr. José Luis de Sousa Magalhães Lima

Prof. Dr. Alberto Yoshihiro Nakano

Prof. Dr. Paulo Leitão

Bragança

2018-2019

Acknowledgments

I would first like to thank my advisers of the IPB Prof. Dr. José Luis Lima and Prof. Dr. Paulo Leitão for their support, document reviews, criticisms, discussions and suggestions, contributing directly to the achieved results. Second, I would like to acknowledge Prof. Alberto Nakano, who willingly accepted to be my superviser from the UTFPR in Brazil. I would also like to thank CeDRI researchers Thadeu Brito and Luis Piardi who actively contributed to this work.

I would like to express my most profound gratitude to Silvio Fernandes, my father, Aurea Fernandes, my mother and Laura Fernandes, my sister. Without them not only would my stay in Portugal have been impossible, but, without my parent's joy in my happy moments and unfailing support in difficult times, this present document would have stayed blank.

I would like to thank my friends with whom I shared an apartment for almost an year in Bragança: Leonardo Candido, Matheus Breve and Luis Guilherme. With them I shared countless happy moments that will remain in memory whenever I remember my stay in Portugal, including long discussions about random topics, our theses, past and future travels and numerous stories.

Special thanks to all my friends and colleagues. Since mentioning all of them would be difficult, I would like to thank those I spent the most time with: Thiago Fernandes, Alessa Oliveira, Fernando Ribeiro, Erick Moretti, Nathalia Gonçales, Nadine Martelozo, Letícia Cena and William Molina.

My sincere thanks to Victor Vallim, Matheus Talacio, João Paulo Moura, Leonardo Gasparoto, João Paulo da Silva, Lucas de Souza and Jordan Bologna great friends who even far contributed immensely to this achievement. Finally, I would also like to acknowledge people not mentioned here who, whether directly or indirectly, contributed to the writing of this document.

Resumo

A reabilitação é um processo relevante para a recuperação de disfunções e para uma melhor realização das Atividades de Vida Diária (AVDs) do paciente. Portanto, o desenvolvimento de tecnologias para este campo tem uma importância significativa, pois o aprimoramento da reabilitação pode afetar muitas pessoas.

Este trabalho propõe um sistema robótico para a reabilitação dos membros superiores utilizando um robô colaborativo e um algoritmo de controle inteligente, o que torna a solução robusta e adaptável para cada paciente. O robô UR3 da Universal Robots© foi usado como base para a implementação de dois algoritmos de *Reinforcement Learning* (RL), o SARSA e o Q-learning, aplicados a esse problema de reabilitação. O objetivo deste sistema é fornecer um treinamento de força comum, aplicando uma resistência ao movimento realizado pelo paciente.

Basicamente, esta tese está dividida em duas partes principais. A primeira foi o desenvolvimento de uma simulação composta pelo UR3 e um modelo do corpo humano na plataforma V-REP, que pudesse ser controlado através de uma interface dedicada ou, externamente, através do MATLAB usando os algoritmos de *Reinforcement Learning*. Essa simulação foi criada com uma interface gráfica para visualização e uma interface homem-máquina, para controlar o sistema robótico com o algoritmo RL, construído no MATLAB. Os resultados obtidos com a simulação apresentaram o comportamento esperado do sistema.

A segunda parte foi o experimento do sistema real com um indivíduo saudável. O experimento foi dividido em duas fases: a primeira considerando o treinamento apenas em um eixo e a segunda nos três eixos cartesianos. Os algoritmos utilizados foram os mesmos da simulação, mas, neste caso, foram implementados na linguagem Python. Os resultados são apresentados quer em simulação quer com o robô real e validam a metodologia desenvolvida e aplicada.

Os resultados obtidos com o experimento real do sistema para a apenas um eixo foram comparados com a simulação do modelo do braço humano proposto em outros trabalhos

para validar a metodologia aplicada.

Este trabalho representa uma contribuição importante para o campo da reabilitação, pois apresenta um novo recurso para ajudar terapeutas e pacientes a obter melhores resultados no processo de reabilitação.

Palavras-chave: *Reinforcement Learning*; Reabilitação Robótica; Modelo do Braço Humano; Robôs Colaborativos; Reabilitação dos Membros Superiores.

Abstract

Rehabilitation is a relevant process for the recovery from dysfunctions and improves the realisation of patient's Activities of Daily Living (ADLs). Therefore, the development of technologies for this field has significant importance because the improvement of the rehabilitation can affect many people.

This work proposes a robotic system for the rehabilitation of the upper limbs using a collaborative robot and an intelligent control algorithm that makes the solution robust and adaptable to each patient. The UR3 from Universal Robots© was used to implement two Reinforcement Learning algorithms, the SARSA and Q-learning, applied to this rehabilitation problem. The goal of this system provides a common training force applying resistance on the movement performed by the patient.

This thesis is divided into two main parts. The first one was the development of a simulation composed by the UR3 and a human model in V-REP platform that could be controlled through a dedicated interface or externally through the MATLAB using the self-control algorithms. This simulation was created with a graphical interface for visualisation, and a human-machine interface, to control the robotic system with RL algorithm, built on MATLAB. The results obtained with the simulation presented the expected system behaviour.

The second part was the experiment of the real system with a healthy subject. The experiment was divided in two phases the first considering the training only in one axis and second in the three Cartesian axes. The used algorithms were the same as the simulation, but in this case, they were implemented in Python language. The experiment considering one axis presents satisfactory results, while for the three axes the results were not so good. The obtained results with the real system experiment for one and three axis were compared with the human arm model proposed in other studies to validate the applied methodology.

This work represents an important contribution for the field because presents a new feature to help therapists and patients to get better results in the rehabilitation process.

Keywords: Reinforcement Learning; Robotic Rehabilitation; Human Arm Model; Collaborative Robots; Upper Limbs Rehabilitation.

Contents

List of Tables **xiii**

List of Figures **xv**

Acronyms **xvii**

1 Introduction **1**

1.1 Robotics Physiotherapy and Diseases 2

1.2 Objectives 3

1.3 Thesis Structure 4

2 State-of-the-Art **7**

2.1 Robotics Rehabilitation 7

2.1.1 Collaborative Robots and Rehabilitation 10

2.1.2 Virtual Scenarios for Rehabilitation Robotics 11

2.2 Intelligent Rehabilitation Robotics 12

2.3 Common characteristics and deviations 13

3 Theoretical Background **15**

3.1 Reinforcement Learning 15

3.1.1 Elements of Reinforcement Learning 16

3.1.2 Markov Decision Process (MDP) 17

3.1.3 Monte Carlo Methods 18

3.1.4 Temporal-Difference Learning 20

3.1.5 SARSA Learning and Q-Learning algorithms 20

3.1.6 Policy and Action Choice 22

3.2 Impedance Control for Human arm 24

3.3 Tools used in the development 26

3.3.1 The UR3 collaborative robot . . . 26
3.3.2 Force Sensor . . . 28

4 Development **31**
4.1 Simulated Environment . . . 31
4.1.1 Virtual Scenario . . . 32
4.1.2 Simulation Codes . . . 34
4.1.3 Human Machine Interface (HMI) . . . 36
4.2 Modelling the System as an RL Problem . . . 37
4.2.1 Agent and Environment . . . 37
4.2.2 Actions, Policy and Reward Signal . . . 39
4.3 Communication Between MATLAB and V-REP . . . 41
4.4 Using the Collaborative Robot (UR3) . . . 42
4.4.1 Development of the self-control in Python . . . 45
4.4.2 Force Measurements and Resistance provided by UR3 . . . 47
4.4.3 Force Signal Sampling . . . 50

5 Results and Discussion **53**
5.1 Simulation Results . . . 53
5.2 Experimental Results . . . 58
5.2.1 The one axis experiment . . . 60
5.2.2 The three axes experiment . . . 66
5.3 Model of Human Arm . . . 70

6 Conclusions **73**
6.1 Future Works . . . 74

References **77**

A The Q-learning results for the one axis experiment **A1**

B Other Results from the three axes experiment **B1**

List of Tables

3.1 Nominal Arm Model Parameters [43] 25
3.2 Other features of UR3 [45] 26
3.3 Maximum joint's torque (UR3) [46] 27
3.4 Technical Specification of ROBOTIQ FT-300 [48] 28

4.1 Used MODBUS ports 47

5.1 Command force_mode applied on the two experiments 60

List of Figures

2.1 Shows the REHAROB the first project to use two industrial manipulators to physiotherapy rehabilitation. . . . 9
2.2 Simulation of a Robotic Rehabilitation System [32] . . . 12

3.1 Generic MDP . . . 18
3.2 Generic Q-Matrix (2 states and 2 actions) . . . 21
3.3 Rewards per Episode for a generic RL problem - SARSA and Q-Learning . . . 22
3.4 Human Arm Modeling [43] . . . 25
3.5 Indication of UR3's joints . . . 27
3.6 Force Sensor FT-300 from ROBOTIQ . . . 29

4.1 Models in V-REP platform . . . 32
4.2 UR3 Configuration . . . 33
4.3 The complete virtual scenario with the UR3 and the Manikin Model . . . 34
4.4 An Illustration of the operation of the both simulation scripts . . . 35
4.5 Simulation settings - Tabs 1 and 2 . . . 37
4.6 Configuration Tab for the insertion of the self-control . . . 38
4.7 Two approaches for Q-matrix used in this work . . . 40
4.8 Block Diagram that represets the both algorithm implemented in the rehabilitation 42
4.9 Flowchart of both controlling implemented on V-REP . . . 43
4.10 Explanation of a generic Synchronous mode between V-REP and an external API 44
4.11 System Architecture of proposed approach. . . . 45
4.12 Force_mode command used to apply the Resistance Force in the Movement . . 47
4.13 Developed tool implemented to help the human hand grab in the real environment . . . 48
4.14 UR3 in the real environment . . . 51

5.1 “General Settings” Tab with the values (simulation) 54
5.2 Simulated Movement of human arm 54
5.3 Data related to movement 55
5.4 Resultant Force in Simulation without the Self-Control Algorithm 56
5.5 Resultant Force with the insertion of self control module 57
5.6 Maximum Torque Update of UR3 in some episodes 57
5.7 Average reward of the RL algorithm 58
5.8 Force training with the patient 59
5.9 The velocity of the intelligent system moving along x-axis considering the SARSA algorithm 61
5.10 The position of the UR3’s end-effector along x-axis considering the SARSA algorithm 62
5.11 Comparison between position and resistance force in the movement considering the SARSA algorithm 62
5.12 The measured force by sensor 63
5.13 The measured force by sensor on the x-axis 64
5.14 The rewards assigned to the system by the SARSA algorithm 65
5.15 The UR3’s applied force chosen by the SARSA algorithm 66
5.16 The measured force in X-axis during the test with All axes enabled 67
5.17 The resultant force in XYZ-axes during the test with All axes enabled 68
5.18 The resultant force in XYZ-axes during the test with All axes enabled 69
5.19 The robot force in X-axis during the test with All axes enabled 70
5.20 The human arm force calculated with the arm model 71

A.1 The velocity of the intelligent system moving along x-axis considering the Q-learning algorithm A1
A.2 The position of the UR3’s end-effector along x-axis considering the Q-learning algorithm A2

A.3 Comparison between position and resistance force in the movement considering the Q-learning algorithm A3
A.4 The measured force by sensor - Q-learning algorithm A3
A.5 The measured force by sensor on the x-axis - Q-learning algorithm A4
A.6 The rewards assigned to the system by the Q-learning algorithm A4
A.7 The UR3's applied force chosen by the Q-learning algorithm A5

B.1 The robot force in Y-axis during the test with All axes enabled B1
B.2 The robot force in Z-axis during the test with All axes enabled B2

Acronyms

API	Application Programming Control
GUI	Graphical User Interface
HIL	Harware-in-the-Loop
HMI	Human-Machine Interface
ICF	International Calssification of Functioning, Disability and Health
MDP	Markov Decision Process
MSD	Mass-Spring-Damper
RL	Reinforcement Learning
ROM	Range of Motion
TD	Temporal-Difference Learning
VR	Virtual Reality
WHO	World Health Organization
YLD	Year Lived with Disability

Chapter 1

Introduction

The rehabilitation field is becoming increasingly important because more people need to regain body functions lost in traumas, diseases, and due to ageing. This importance is clearly expressed when it is analysed the increase of the absolute number of Years Lived with Disability (YLDs), between 2005 and 2015, a rise of 17 million has been recorded [1]. This data represents all people suffering from any disability and takes into account the life estimate in each world region. According to a study presented by the World Health Organization (WHO) in [2], the number of YLDs was approximately 119 thousand million in 2015, with almost 1000 million people showing disabling health conditions. About 74% of the YLDs recorded in 2015 are linked to health conditions whose rehabilitation may be beneficial, and this represents about 740 million people worldwide who need these services.

Define a concept of rehabilitation is difficult because the disability represents a diversified set of health conditions. It is classified by *International Classification of Functioning, Disability and Health* (ICF) [3] in three inter-connected areas: impairments, activity limitations and participation restrictions. Impairments are problems in body function or alteration in body structures. Activity limitations are difficulties in executing any activity of daily life. Participation restrictions are problems with the involvement in any area of life (for example: facing discrimination or limited transportation). Therefore the rehabilitation is important principally in the first and second areas and impact in the third secondarily. Thus the rehabilitation definition that will guide this work is introduced by WHO in the *World Report on Disability* [4] that defines as a "set of measures that assist individuals who experience, or are likely to experience, disability to achieve and maintain optimal functioning in interaction with their environments".

To endorse the importance of improving the rehabilitation was also created by WHO [5]

an action plan that presents the objectives that must be taken into account in decisions of Member States, partners and WHO itself, this plan is valid between 2014 and 2021. Among the objectives exposed, are presented concerns about the technologies applied to rehabilitation, called Assistive technologies, specifically the article 35 presents:

> "Assistive technologies are evolving quickly and include any item, piece of equipment or product, whether it is acquired commercially, modified or customised, that is used to increase, maintain or improve the functional capabilities of individuals with disability. [...] Assistive technologies play a significant role in enabling people with disability to function and participate."

Therefore, it is evident that there is a concern for developing new assistive technologies and making them accessible. An assistive technology must be suitable to the user and its environment to increase independence and improve participation in daily activities [4]. Examples of assistive devices are prostheses, orthoses, wheelchairs, hearing aids, ocular devices, talking books and speech synthesisers. The objective of this introduction is to contextualise the main diseases that can lead the patient to dysfunction, the objective of this work, and the thesis structure.

1.1 Robotics Physiotherapy and Diseases

The rehabilitation process requires a different approach depending on the situation that caused the current health condition. Some several conditions and diseases may lead a patient to develop partially or totally dysfunctions. Some of these diseases already received a previous approach using robotics and represent a considerable contribution to the field. In a literature review accomplished by Ruiz [6], was explored which are the main diseases that rehabilitation robots are applied. Among all data presented, the upper limbs are principally assisted when there is the occurrence of Stroke or Parkinson's Disease. The concerns about the post-stroke patients lead to numerous studies and several approaches using the robotics, and it is interesting to explain this disease and why it is so researched.

According to WHO the stroke can be defined as a cardiovascular disease, caused by the interruption of blood flow in the brain [7]. The dysfunction produced by the stroke depends on the location of occurrence, but the main ones are somatosensory deficits, pain, visual deficits, motor deficits, changes in muscle tone and others [8]. The development of technologies to rehabilitation after stroke is significant because, as stated by Mackay and Mensah [7], annually about 15 million people are affected by the disease and approximately 10 million develop dysfunction that complicates the realisation of daily activities. The robotic devices resulting from concern about the stroke are diversified and created for various purposes. The studies related in this work takes into account the approaches for the upper limbs.

Some interesting devices that help post-stroke patients may be presented. One of them helps the physiotherapy of the hand of post-stroke patients is shown in [9]. The device was designed to encapsulate the human hand when the movement happens. It records data referring to force and position, and this information is presented to the therapist and may improve the therapy on the patient's side. A mechanism address to rehabilitation of the shoulder of post-stroke patients is shown in [10]. The proposed system uses six Degrees of Freedom (DOF) and is attached to the shoulder. So, the system also promotes the improvement of rehabilitation because increase the workspace, provide acute alignment between the robot and patient's rotation axes, and does not require additional adjustment.

1.2 Objectives

This section explains the goals for both environments, the simulation and the real. The objectives are divided in topics to facilitate the understand, and are:

- Development of an environment simulation in the software V-REP with the main elements of this approach:
  - Model of the UR3's robotic arm from Universal Robots;
  - Model of the human body taking into account the joints of shoulder and elbow;

- Modelling the robotic rehabilitation physiotherapy using the Reinforcement Learning technique using the following algorithms:
  - Q-Learning: off-policy algorithm;
  - SARSA Learning: On-Policy algorithm;
- Implement the RL algorithms developed in the simulation to test, analyse, and correct some problems before the real robot use.
- Use the collaborative robot (UR3) to implement the developed RL algorithms, training the system and collect the results.
- Compare the obtained results from the real experiment with a model of human arm to validate the approach used.

1.3 Thesis Structure

- Chapter 1: Introduction
  - Contains a contextualisation about the rehabilitation physiotherapy, motivations and the objective of this work.
- Chapter 2: State of the art
  - Contains the literature review about the rehabilitation robots, the simulation in this field, the common characteristics, and deviations between this work and those found in literature.
- Chapter 3: Theoretical background
  - Briefly explains the theoretical background used in the development, the subjects addresses are the Reinforcement Learning, the modelling of the human arm and the explanation about some tools used.
- Chapter 4: Methodology

  - Explains the proposed simulation structure, the real structure, and the implementation of Reinforcement Learning Algorithms to the proposed problem.

- Chapter 5: Results

  - Presentation of the obtained results in all experiments performed, including the simulation. Also the analysis and discussion of the main observed characteristics of the implemented system are presented.

- Chapter 6: Conclusion

  - Contains the conclusions of this works and the recommendations for future works.

Chapter 2

State-of-the-Art

This chapter presents some relevant works that address the assistive devices for physical rehabilitation, specifically for upper limbs through the use of robotics, besides the existing robotic rehabilitation systems in the literature and the different approaches of each work are explained. Also the common characteristics and the deviations of this work in relation to the other is presented.

2.1 Robotics Rehabilitation

The rehabilitation robots are one of the possibilities that the modern robotics provided. There are many types of robots with different purposes that are used to provided helps to specific tasks or improve the physiotherapy of the patients. In 2001, Tejima [11] accomplished a review of several works, classifying the types of rehabilitation robots according to their tasks and separating them in four groups: the augmentative robots, the augmentative mobility, robots for help care-givers, and the therapy robots.

The augmentative robots are used to improve a specific or multiple tasks on the patient side, as the pick and place of an object or an orthoses that allows the patient perform some task. Devices used to improve the mobility of the patient, as the robotic wheelchair, are from the augmentative mobility group. The robots for help care-givers are those devices that help nurses or care-givers to accomplished a task, as to move the body of a patient. Robots used in occupational or physical therapy compose the fourth group and are the focus of this work.

The therapy robots also can be separated according to operation and support provided to the patient. Badesa, Morales, Sabater-Navarro, *et al.* [12] presented this classification: the assistive and resistive mode.

- In the **assistive mode** the robot causes the movement in the limb of the patient, generating forces that replace or add up to the patient's strength.
- During the active movement of the patient the robot in **resistive mode** stores the data about the performed exercise, as position, force, velocity and other. The robot can also be set to exert different levels of resistance to the motion.

The first concepts of the industrial robotics appear in the late 40's, but only in 1954 a robotic manipulator with memory was patented by George Charles Devol and in 1961 the same started to be used in industrial level for the General Motors. The robotic manipulators are the electro-mechanical structures controlled by a software, usually anthropomorphic, i.e., similar to human arm. The idea of use this robotic systems to rehabilitation begun years late, in the 80's the use of robotics in contact with humans or even near by them was not recommended due their physical dimensions and accident history. Only in 1999, with a research accomplished by Budapest University of Technology and Economics, a common industrial manipulator was used to perform the physiotherapy rehabilitation, this project was called REHAROB [13].

The objective of REHAROB project is help the treatment of shoulder and elbow of hemiparetic patients [14]. The Hemiparesis is a sequel of stroke that some patients may develop depending on the location of the brain injury, however a study conducted by Taub, Miller, Novack, *et al.* [15] in 1993 evidenced that the motion therapy has a positive effect in patients with this health condition. The realisation of this therapy can be quite stressful for the therapist because repetitive movements must be performed several times, thus the application of a robot in there cases represent an improvement for rehabilitation [16]. The REHAROB is a robotic system composed by two ABB industrial robots, the IRB 140 connected to upper arm and the IRB 1400H connected to lower arm, and this system is shown in the Figure 2.1. The mechanical design is presented in [14].

The exercises conducted by the robotic system are result of a research about conventional movements performed in this type of physiotherapy and were found 45 motions which are respectively presented in [17]. The validation and the verification of the efficiency was

Figure 2.1: The REHAROB system connected to upper and lower limb of a patient (Modified from Toth, Fazekas, Arz, *et al.* [16]).

conducted by a clinical test presented in [16]. The clinical tests consisted of two groups, the first with four healthy persons and the second with eight patients. Each patient received an accumulated 30 minutes of robotic physiotherapy during 20 consecutive working days, totalizing 7200 minutes of robotic operation in the whole group of patients plus 3600 minutes of operation during the system preparation. The results demonstrated an improvement in the patients on: the range of motion, the spasticity of the antigravity, the muscle strength, and the Functional Independence Measure.

Currently, there are many types of robotic systems commercially applied to rehabilitation, some of these examples are the devices provided by Hocoma©, the ARMEO® robots [18]. These robots are already used in therapy clinics around the world and some studies shown the contributions to the rehabilitation of the patients, as the presented in [19], [20] and [21]. A list of other devices used for upper limb rehabilitation is shown in [22], are displayed the characteristics of each system and for which part of the limb is designed.

2.1.1 Collaborative Robots and Rehabilitation

The interaction between human and robots, where both can be in the same room with a high level of safety could be impossible in the 80's. However, currently, it is perfectly possible due to emergence of collaborative robots (cobots). The idea of use cobots is nothing new, in 1996 Colgate, Edward, Peshkin, *et al.* [23] in the paper *Cobots: Robots for collaboration with human operators* presented a system with this purpose.

At the industry level, the cobots already took their places. This type of robots are used for several tasks as pick and place, welding and other precision tasks. The examples of this industrial robots are: the UR3 from Universal Robots© [24], the LBR iiwa from KUKA© [25] and the YuMi from ABB© [26]. The UR3 was used to develop this work.

Some authors use cobots to rehabilitation purposes. One of these approaches was accomplished by Papaleo, Zollo, Spedaliere, *et al.* [27]. In their paper, a cobot of KUKA with 7 DOF which adapts to patient needs in real-time was used. The patient is attached to end-effector of the robot, so the system applies an assistive force when the patient cannot complete certain trajectory or a resistive force when moving away from it. A experimental validation was conducted taking into account healthy behaviour and a simulated post-stroke behaviour when the patients intentionally fail in the motion. The results of the system are interesting because the robot acquires the patient data about the movement and use to modifying the task complexity.

Another project that uses cobot-based approach to achieve rehabilitation is the Universal RoboTrainer [28] [29]. This system uses a Universal Robot to train disabled upper limbs, and the idea is based on studies that prove the effectiveness of repetitive movements in the treatment of dysfunctions. Some tests with this device have been performed. One of them addresses the confidence in the medical human-robot iterations shown in [30], and presents the main issues in that approach on the patient side, as signs of distress in the exercises, concern and expectation about the movements attached to a robot arm, and others.

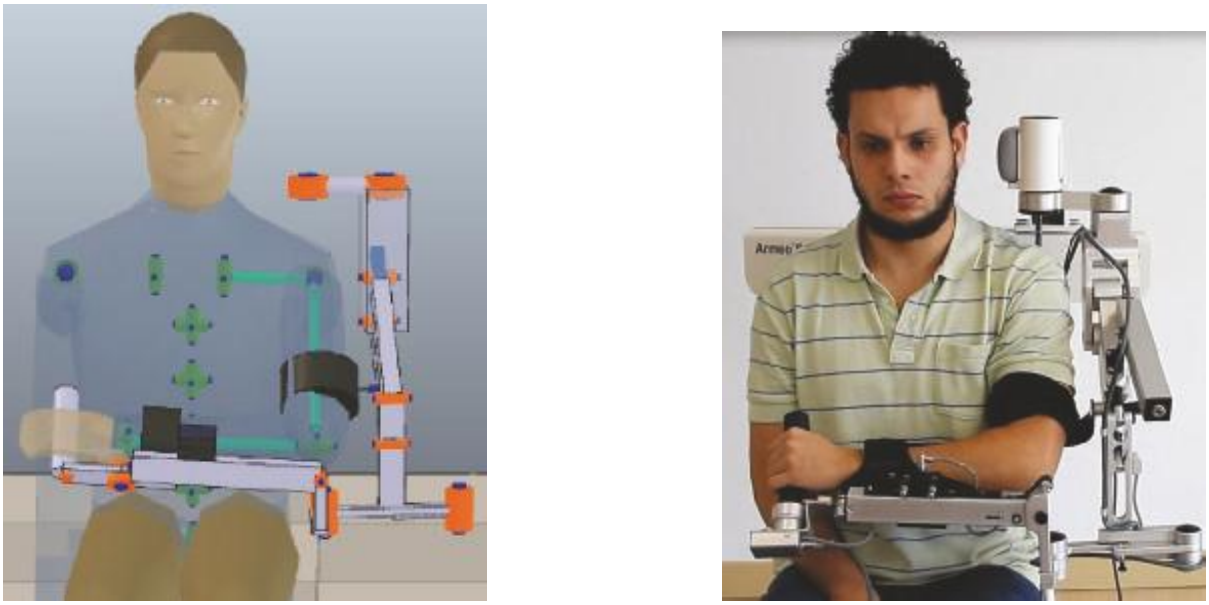
2.1.2 Virtual Scenarios for Rehabilitation Robotics

Another tool employed to rehabilitation is the virtual scenarios that are used with several purposes. In literature many works have been found using this approach to create better systems, provide visual feedback to the patient or test devices by simulating the environment. Rehabilitation robots are potentially problematic because works with the human contact and, for this reason, requires a high level of safety. Therefore, the clinical test for assessment of new technologies is sometimes not possible and consequentially delay the development of the project, so the simulation also have the purpose to accelerate this system where there is the human contact.

A simulation of rehabilitation environment with a model of the human limb can be seeing in [31]. It presents the design and simulation of an exoskeleton with eight DOF to upper limb rehabilitation. The model of the upper limb was developed in Opensim, the design and the mechanical structure was constructed in Solidworks, and the MATLAB was used to perform the dynamic simulation through the Simmechanics. This dynamic simulation helped to model a predictive dynamics problems in the motion of the elbow considering the torque square and the minimum execution time for the desired rotation.

A model of a upper limb integrated in a virtual reality (VR) can be seeing in [32], the proposed work establish a method to estimate the posture of the human limb attached to the exoskeleton. The joint angle measurements and the constraints of exoskeleton are used to estimate the human limb joint angle in virtual reality. The robot-based system used was the Armeo Spring exoskeleton, a standard robotic system commercially used in rehabilitation field. The simulation was performed in V-REP platform and signals were measured from ARMEO through software and hardware provided by outsources. The simulation can be seen in Figure 2.2, where Figure 2.2a shows the environment simulation and Figure 2.2b presents the system that was simulated.

Du, Sun, Su, *et al.* [33] presents other approach that uses a VR to perform the rehabilitation with the robotic system, in this case the system was developed in ROS/Gazebo and presented an immersive VR display. When the patient perform the movements, the robotic system



(a) Simulation on V-REP

(b) Robotic System with Human

Figure 2.2: Simulation of a Robotic Rehabilitation System [32]

acquire the data and transmits to ROS/Gazebo, thereby patient can see in real time your performance inside the simulated environment. Therefore, the patients could gain intuitive feelings about arm's movement through visual feedback. The virtual scene is a standard kitchen with various objects and a human body, this scene represents some ADLs.

2.2 Intelligent Rehabilitation Robotics

A rehabilitation robot can be controlled in many ways depending on the work environment, the performed tasks, and how the contact with the patient will be. The problem of the several existing control types is the adaptation to each patient. This adaptation generates a huge wasted time to set up the parameters and the system to the new person that interact with it. So, develop adaptive robots to deal with the patients in a dynamic manner is very important to the field [34].

Some authors address their papers to adaptive controls and developing interesting systems, is the case of the Krebs, Palazzolo, Dipietro, *et al.* [35] where is proposed a performance-based progressive robot therapy. The work uses the MIT-MANUS, and the objective is to

deliver an optimal therapy for each patient. The system uses a game when the patient receives feedback through a display according to the movement performed. This game is a series of point-to-point moves that the patient achieves with a determined level of robot assistance. While the movement is going on, the robot acquires the EMG, speed, and time signals to evaluate the patient's performance. Thus, according to this, change the robot's parameters for a new interaction. The performance values analysed are the ability to initiate a movement, move from a starting position to the target, aim its movement along the target axis and reach the target position. When a patient executes a right movement, the system decreases its stiffness, providing less guidance to the patient and challenging him to improve furthermore.

The study conducted by Kahn, Rymer, and Reinkensmeyer [36] presents another force oriented adaptive control to assist the chronic stroke patient. The developed algorithm is implemented in an ARM-Guide device that consists of a servo motor controlling a position of the patient's arm. The signals of measured speed to determine an assistive or resistive force for the next movement are used. The measured velocity is compared with the desired value for this variable, when this velocity is smaller than the desired one, the patient is classified with an injury and the system provide an assistive force to complete the next movement. Otherwise, a resistive force is applied, and the patient is classified as a healthy person. The system was tested during eight weeks using this type of control with one hemiparetic individual, and the results were: improvement of the reach distance in exercises, higher velocity after the therapy sessions, and consequently shorter times to complete the functional tasks.

The control used in this work is based on Reinforcement Learning (RL) approach, this type of algorithms uses environment feedback to update its parameters providing an adaptive control making this method an innovative approach.

2.3 Common characteristics and deviations

The main goal of this project is to develop a robotic system to assist the rehabilitation of patients with upper limb dysfunction, principally in the shoulder. It is emphasized that the system is built to meet the needs of patients that are capable to perform the limb's movement,

however with a deficient strength and not in the whole Range of Motion (ROM).

The operation of the proposed structure is classified as resistive mode [12], this means that the robot will exert a resistance force and will acquire the data from the patient's performance. When the patient perform the exercise repetitively the upper limbs are able to recover the lost strength, as shown in [16]. The data acquired from this system on the proposed exercises are the performed strength of the arm (in three dimensional axes and the resultant) and the variations of it, the execution velocity of the tasks and the ROM. All information are available to the therapist.

This system is based in the collaborative robot (UR3) from Universal Robots© and the control of the force applied on the movement is provided through a self-control algorithm based in the RL approach. The self-control algorithms is developed using MATLAB to communicate with the dynamic simulation, and using Python when the actual robot is used. It is noticed in [16] that a long time is wasted in the set up of the system for each patient and exercise, this is avoided by using an autonomous control algorithm because it is adaptable to patient needs (force) on a free trajectory, i.e., the therapist may indicate any path to the patient without having to configure the movement on the robot and, when there is a realisation of the motion, the autonomous algorithm gives the resistance force independently of the exercise's directions.

The self-control algorithm needs a training period before the human contact, besides is necessary to predict the dynamic problems of the system to ensure patient safety, similarly to the tests accomplished by [31]. Therefore, a simulated environment is proposed to perform the system tests and acquire data to analyse and evaluate it. The simulation is performed through V-REP from Coppelia Robotics. The main elements are a human body model, simulating principally the shoulder joints, and a UR3 model. All simulated dynamic parts of the cobot were already implemented (dynamic model provided by Coppelia Robotics). However, the human arm representation were developed through the combination of linear joints and provided a simplified way to training it.

Chapter 3

Theoretical Background

This chapter presents the theoretical background involved in the development of the proposed system. The introduced knowledge refers to the control implemented, the kinematic iterations between the robot and human arm, and the used tools to develop this work.

3.1 Reinforcement Learning

In this work the Reinforcement Learning (RL) algorithms to control the robotic system were explored. These algorithms use feedback from the environment to adapt and update its parameters. The RL is a type of unsupervised machine learning technique because the algorithms do not use a database or previous knowledge to training, but the signals measured from the environment are used to award rewards, which will determine the correct behaviours of the structure. Therefore, it is important to know the concept of these types of algorithms. The study presented in this section was based on the references [37], [38], [39], and [40].

Interact with the environment is the capacity of every living being. These interactions provided important biofeedback because these signals are used to improve skills. As an example, a human child uses an RL technique when learns to balance, walk or waves its arms without an explicit teacher. This technique may also be used by machines to improve their behaviours in dynamic situations because it provides an adaptive control referenced on environment. The main challenge in working with this concept is knowing how to get feedback from the environment, which of these signals are essential, which actions to take, and how to differentiate right and wrong actions. All these concerns should be taken into account before making a decision.

In a decision process, an agent within a determined environment must take a correct

decision to choose a specific action in a set of possible actions, in order to move from a state to another and learn which actions in those states return a better reward. It is necessary to know the main elements that compose an RL technique to explore this concept.

3.1.1 Elements of Reinforcement Learning

The RL is formed by two main elements, the agent and the environment. To explain these, it will be used as an example a game of chess. In this game the agent is the player and the environment is the chess game. The player must decide on a set of possible action, that represents all the movements of all pieces, which one will be taken. Depending on the action the game will move from a specific state to another, these states represent the possible new configurations of the pieces in game. These states are dynamically changed until there is a winner.

Another elements also compose an RL, and are described as: the policy, the reward signal, the value function, and optionally a model of the environment.

The policy defines how an agent will learn, i.e., the actions will be taken with a purpose, in the chess game the policy is compared with the strategy taken by the player to pursue a positive result. The reward signals defines the goal of the RL problem and depending on the strategy of the player, if the player wants take the maximum pieces of the opponent the reward signal must be modelled differently of when the player strategy is take the opponent's strongest pieces [37]. The value function related the optimal criteria with policies, in other words, this function evaluate how good it is for the agent perform a determined action in that specific state, this concept is difficult to analyse in a chess game because is subjective, but one way to look at how this function works would be to assess whether the player is winning or not the game [38]. The model of the environment represents a way to predict the next state and next reward given an action taken, this model will not always exist, in the example this model could be a set of probabilities to predict the next opponent's move.

The described elements compose a generic RL and the concepts presented will guide this work. This type of algorithm is constructed to make a decision in a dynamic process and to

adapt the next choice to the environment, therefore it is important to know which concepts make up this operation. To study RL problems, the Markov Decision Process (MDP) is used [37].

3.1.2 Markov Decision Process (MDP)

An MDP is a mathematical framework for modelling the environment in a decision making, i.e, when a decision-maker is about to actuate in a specific environment some elements must be taken into account, such as the current state of the system, the next desired state, the actions that can be taken and the probabilities of these actions leading to the desired place. All these processes and elements can be described mathematically through the MDP framework. The formulations presented here were explained by Szepesvári [39]. The MDP is defined as a triplet $M = (S, A, P_0)$, where S represents the set of states, A is the set of actions, and P_0 is the transition probabilities. The P_0 presents a value to each state-action pair $(s, a)\ \epsilon\ S\ \times\ A$. Thus the probability of a system moving from a state s_1 to some other state s_2 given an action a chosen in s_1 state is:

$$P(s_1, a, s_2) = P_0(\{y\} \times \mathbb{R} \mid s_1, a) \tag{3.1}$$

Therefore, the same action taken in the same state does not always result in the same next state. Also, it is necessary to formulate the immediate reward function, which gives the expected immediate reward when an action a is chosen in s_1 state, taken into account the probabilities mentioned above:

$$r(s_1, a) = \mathbb{E}\left[R_{(s_1,a)}\right] \tag{3.2}$$

The term $\mathbb{E}\left[R_{(s_1,a)}\right]$ represents the search for the reward in the data set $R_{(s_1,a)}$, the problem is that this data is not always available in table form, and sometimes this value of immediate reward is obtained from the environment. Another significant value is the final reward R_{final}, or the reward awarded when is reached the goal, and this is obtained only one time. To better

illustrate the Figure 3.1 presents an MDP process graphically.

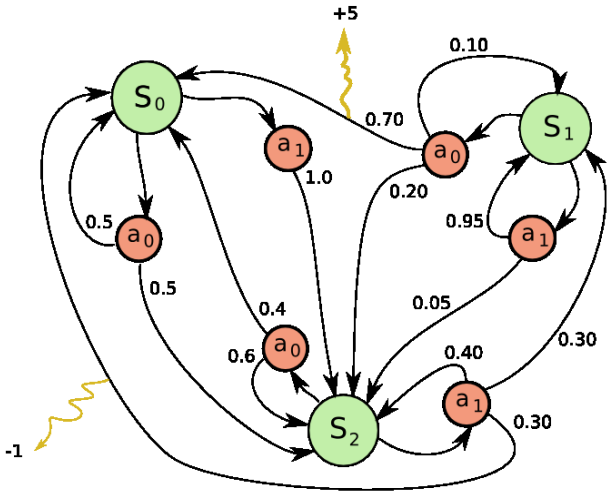


Figure 3.1: A generic MDP with three states [41]

Figure 3.1 shows three states s_0, s_1 and s_2, in each state two actions can be taken a_0 and a_1. The black arrows represent the transition probabilities and the yellow arrows the rewards that can be obtained. In that configuration, the decision-maker can start in any state, and the goal is reached on the +5 yellow arrow. So, if the decision-maker starts in state s_1, it should choose the action a_0 to get a chance to reach the goal. If it starts in another state, it must first moves to state s_1. This Figure does not show the immediate reward, but the final one. This immediate reward can be imagined as a positive value awarded if the agent approaches +5 yellow arrow.

The modelling of the environment is the first step to implement an RL algorithm. Other elements that need the modelling inside the framework of the RL problem is: the policy that will guide system learning, and how the updates of the RL algorithm will be performed.

3.1.3 Monte Carlo Methods

Almost all RL algorithms involve estimating value functions, which determines how good it is for the agent to be in some state. In each decision, the agent expects future rewards that will define this notion of "how good", so learning methods for estimating the value functions and

discovery optimal policies is important [37]. A learning procedure that uses the experience from the contact with the environment to create the knowledge are the Monte Carlos methods. This procedure is a way of solving the RL problem based on averaging sample results.

Learning the state-value function for a given policy to predict the returns obtained is the core of the RL problem. Let's suppose the goal of estimating the value of $V_\pi(s_1)$, which represents the expected return starting from a state s_1 under the policy π. A set of episodes obtained following π and passing through s_1 is used to find out the function. Each episode consists of the set of interactions with the environment that the agent perform, starting at state s_x and finishing at s_y. Each occurrence of state s_1 in an episode is called a visit to s_1. Sutton and Barto [37] presents pseudo-code that illustrates a simple and generic estimation of the value $V_\pi(s_1)$ is presented in Algorithm 1.

Algorithm 1 Monte Carlo Method

```
1:  function STATE-VALUE FUNCTION
2:      Initialise:
3:          π ← policy to be evaluated
4:          V ← an arbitrary state value function
5:          Returns(s_1) ← an empty list, for all s ϵ S
6:      Repeat forever:
7:          Generate an episode using π
8:          For each state s_1 appearing in the episode:
9:              G ← return following the first occurrence of s_1
10:             Append G to Returns(s_1)
11:             V(s_1) ← average(Returns(s_1))
12: end function
```

The code represents a simple way to evaluate a state s_1 under the policy π. This approach requires a reward model linked to the environment for the rewards to be awarded because there is no modelling that links rewards to states, i.e., it is not possible to award a reward without knowing if this is the right way. However, the goal of an RL is evaluating all state-action pairs using the same structure of the Monte Carlo methods, in other words, will be granted to all state-action pair a value calculated through the defined policy and using the rewards acquired.

3.1.4 Temporal-Difference Learning

The Temporal-Difference (TD) Learning is a method to solve the RL problems, that merges the directly learning from experience with the constant update of the values learned, without waiting for a final outcome [37]. Thus, a reward model linked to the environment is no longer necessary because the award needs to wait until the next time step. Thereby, the update rule for the state value ($V_\pi(s_1)$) is:

$$V(s_1) \leftarrow V(s_1) + \alpha(\left[R_1 + \gamma V(s_2) - V(s_1)\right]) \tag{3.3}$$

In Equation (3.3) the value of the state s_1 is updated taking into account the value of reward acquired in s_2 plus the value of the same state. The constants α and γ are named learning rates and both are values in the range [0,1]. As explained the same action in the same state may lead to two different results, this means that the update of state values must be done with caution, thus if the value of γ is close to one, the update based on the next state will have a huge impact. The α is used to ensure that $V(s_1)$ growth is not so accelerate that it takes the system to a non optimal response, i.e., if the value of α is close to one the system runs the risk of understanding that any path that seems right is the better one [37].

3.1.5 SARSA Learning and Q-Learning algorithms

All knowledge presented above guides the idea of an algorithm to solve an RL problem. However, to a complete solution, it is still necessary to use the state-action pair. The Equation (3.3) is again used, but in this case, the value is updated inside a matrix, the Q-matrix. This matrix correlates the pair-values and provides an excellent method to solve the problems.

The first control algorithm explained is the SARSA Learning, this algorithm uses all concepts seen and provided a value function very close to Equation (3.3). So, the update of a pair (s_1, a_1) is accomplished taking into account the state s_2 achieved after performing a_1, the action a_2 chosen in the state s_2 and the reward R_1 awarded after the action a_1, as shown by Equation (3.4). The algorithm is known as SARSA because the configuration of the value function, (s_1,a_1,R_1,s_2,a_2).

$$Q(s_1, a_1) \leftarrow Q(s_1, a_1) + \alpha([R_1 + \gamma Q(s_2, a_2) - Q(s_1, a_1)]) \tag{3.4}$$

Notice that this is an on-policy algorithm, that is, the policy of choosing an action is taken into account in the Equation (3.4). The SARSA algorithm is shown in [39]. There are also the off-policy structures, which uses the same concept described, but do not take into account the action choice. Q-Learning is an example of this type of algorithm. The difference from SARSA is the structure of the state-action pair update. So, the value function uses the state achieve s_2 and the reward R_1 to assign the value to the (s_1, a_1) pair. Thus, the configuration of Q-Learning is (s_1, a_1, R_1, s_2). The Equation (3.5) shown this value function.

$$Q(s_1, a_1) \leftarrow Q(s_1, a_1) + \alpha([R_1 + \gamma max(Q(s_2, :)) - Q(s_1, a_1)]) \tag{3.5}$$

In Equation (3.5) the update disregard the action a_2, and uses the maximum state-action value $max(Q(s_2, :))$ for the state s_2. An example of Q-matrix for both algorithms is presented in Figure 3.2. As can be seen, the states are related to the matrix lines and the actions are linked to the columns.

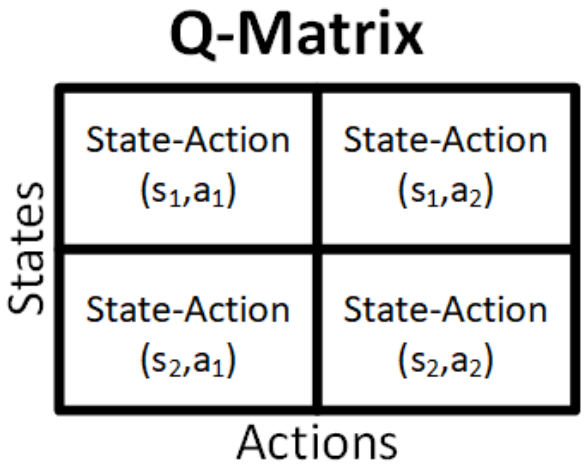


Figure 3.2: A generic Q-matrix with 2 states and 2 actions.

Some features are used to evaluate the performance of an RL algorithm. As the goal of the agent is known, the optimal behaviour will be the convergence to the goal in as few steps

as possible. This optimal number of steps will only be known when enough episodes have passed. Another way to evaluate the agent's behaviour is the number of rewards per episode. This number should increase as the agent learns, and is shown in the Figure 3.3 which presents these values for a generic RL problem. Other ways to assess whether the agent is making the right decision is using environment parameters as the rewards acquired per episode.

Figure 3.3: Rewards per Episode for a generic RL problem - SARSA and Q-Learning [37]

.

Know each algorithm and its value functions represented in Equations (3.4) and (3.5) is essential and makes it possible to study how actions should be chosen. This is an essential part of the knowledge, because the correct choice of the action to achieve a goal is the main objective of this algorithm.

3.1.6 Policy and Action Choice

As seen in Section 3.1.5 both Q-Learning and SARSA Learning present a Q-matrix that correlates the state-action values with the reward acquired by that pair, these values are a way to evaluate the desired behaviour of a system. In these algorithms, the Q-matrix values are also used to perform the choice of the next action.

In a simple RL problem, a policy to guide the action choice may be the greedy policy. Basically, there is a search for the best action in the current state through an analysis of the Q-matrix values, for example, the Figure 3.2 presents a generic Q-matrix, if the system is in

state s_1, the greedy policy will search which one action has the most significant value, a_1 or a_2, and based on this will decide the next action. The choice through the greedy policy is demonstrated in pseudo-algorithm 2, considering a state s_x.

Algorithm 2 Greedy Policy

```
1: function ACTION CHOICE
2:     π_g ← maximum element of Q(s_x,:)
3:     Index ← the indexes of the matrix for π_g
4:     Action ← the second term of Index (represent the better action)
5: end function
```

This is a simple way to determine the next action, but is not the better way. In RL problems the agent faces a conflict between exploration and exploitation, this means that the agent besides choosing the better action must also explore new possibilities to avoid getting stuck in a behaviour that seems correct but not the optimal. Therefore, in most cases, a modified greedy policy called ε-greedy policy is used. Basically, in this case, a random value in the range [0,1] is generated and compared with the ε value. The ε must be a constant close to 0, in this case is used $\varepsilon = 0.1$. If the random number generated is larger than ε the greedy policy is chosen, else the next action is also chosen randomly. The pseudo-algorithm 3 shows the action choice structure based on the ε-greedy policy.

Algorithm 3 ε-Greedy Policy

```
1: function ACTION CHOICE
2:     ε ← 0.1
3:     Generates a random number between 0 and 1
4:     if random >= ε then
5:         Choose the greedy policy
6:     else
7:         Choose a randomly action
8:     end if
9: end function
```

The use of this technique ensures that there will always be an exploration, and the agent will learn all possible ways to complete a specific task. With all the knowledge presented in this section is possible to implement these algorithms to a rehabilitation problem.

3.2 Impedance Control for Human arm

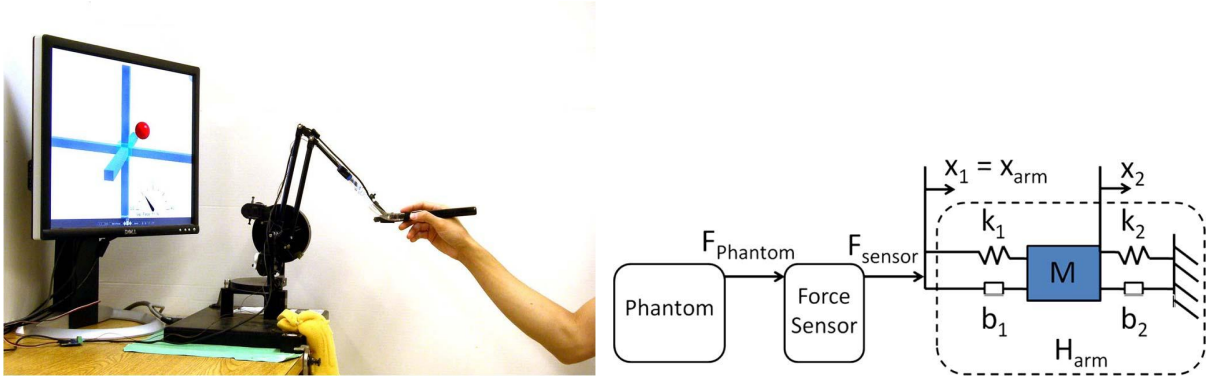
It is relevant to understand how is the behaviour of the human arm and its response in the movement execution, to accomplish a complete study about the rehabilitation of the upper limbs. Some authors have already studied this subject and created some models to simulate the human arm responses in a dynamic environment. In an investigation conducted by Ajoudani [42] the human arm may be modelled as a Mass-Spring-Damper (MSD) system in its end-effector, this means that the human arm can be described as a function of Equation (3.6)

$$F_{ext} = M_{hm}\Delta\ddot{\chi} + D_{hm}\Delta\dot{\chi} + K_{hm}\Delta\chi \tag{3.6}$$

This equation relates an external force F_{ext} with the parameters of the arm's movement in an equilibrium situation. M_{hm}, D_{hm}, and K_{hm} represents the virtual matrices of mass, damping and stiffness of the system, the values of $\Delta\ddot{\chi}$, $\Delta\dot{\chi}$ and $\Delta\chi$ are the variations in acceleration, velocity and position of the human arm. The main challenge in using this model to simulate the arm's response is knowing the parameters that represent the human limb: M_{hm}, D_{hm}, and K_{hm}. Usually, these values are found, taking into account some physical tests with an individual representing the set of people targeted by the study.

Fu and Cavusoglu [43] conducted a study that aimed to know these values of the human arm. In this paper, a model of the human arm is also considered using the MSD approach, but in this model, the human limb is described as two sets of spring-dampers and one mass. The experiments to define the human arm function and the values that represent the limb was performed with the Phantom Premium 1.5a, a haptic device that measures the end-effector force. The system used to acquire theses values and the model is presented in Figure 3.4.

As can be seen in Figure 3.4b the model is more complex that the accomplished in [42]. Therefore, a new transfer function H_{arm} must be found to represent this system. The Equation (3.7) presents this new human arm function, correlating the F_{sensor} (force measured at the end-effector) as input and position of the arm X_{arm} as an output. The Equation 3.7 is in Laplace notation. With this transfer function the experiments were conducted in order to found the following terms b_1, b_2, k_1, k_2 and M that represent the human limb values.



(a) System based on Haptic Device **(b)** Model of Human Limb

Figure 3.4: Human Arm Modeling [43]

$$H_{arm}(s) = \frac{X_{arm}(s)}{F_{sensor}(s)} = \frac{Ms^2 + (b_1 + b_2)s + k_1 + k_2}{b_1 Ms^3 + (b_1 b_2 + k_1 M)s^2 + (b_2 k_1 + b_1 k_2)s + k_1 k_2} \tag{3.7}$$

The accomplished experiment considered 135 subjects using the system represented in Figure 3.4a and presents the model parameters for the three axes (X, Y, and Z) which are presented in the Table 3.1.

Table 3.1: Nominal Arm Model Parameters [43]

Axis	*M (Kg)*	k_1 *(N/m)*	k_2 *(N/m)*	b_1 *(N*s/m)*	b_2 *(N*s/m)*
X-axis	0.2179	379.5	78.75	1.8390	4.645
Y-axis	0.2692	552.4	105.3	3.6090	6.430
Z-axis	0.2041	769.9	271.7	0.7764	18.060

These values about the human limb mentioned in Table 3.1 together with the Equation (3.7) are enough to simulate the human arm response. However, Yang, Wang, Tong, *et al.* [44] show that the model of the human limb can be further simplified in the rehabilitation field, since the movements performed by the patients are slow and extend for long periods of time, contradicting the classical perturbation approach presented in [42] and [43]. Therefore, this approach considers only the spring factor and the position to describe the human limb response. The Equation (3.8) shows the simplification.

$$F_{ext} = k(x_d - x) \tag{3.8}$$

The F_{ext} represents the vector (3x1) of desired force in the directions X, Y, Z, k is the matrix (3×3) of system's stiffness, and the term $x_d - x$ is the displacement of the end-effector.

The Equation (3.8) shows a way to describe the human arm behaviour depending on the position and the stiffness of the arm, and this approach will be used to compare the results obtained with the system that will be proposed in this work. The factor k that describes the human arm parameters only was studied for healthy subjects in previous works, so that there is no data for disabled people.

3.3 Tools used in the development

This section presents some tools used in the development of the intelligent system for robotics rehabilitation, principally the physical elements of the system, since these are the main parts that most limit the potential of the proposed structure. Therefore, it will be addressed the collaborative robot and the force sensor.

3.3.1 The UR3 collaborative robot

The UR3 is a collaborative robotic arm from Universal Robots, and it is shown in Figure 3.5. This robot has a static base, six revolution joints that grant a movement with 6-DOF, and 3 kg of payload [45]. Some other essential features for this work are presented in Table 3.2.

Notice that in the Figure 3.5 each joint is shown by the letters A, B, C, D, E, and F, that respectively represents the Base, Shoulder, Elbow, Wrist 1, Wrist 2, Wrist 3.

Table 3.2: Other features of UR3 [45]

Feature	*Value*
Reach:	500 mm
Joint ranges:	+/- 360º Infinite rotation on end joint
Speed:	360 degrees/sec (Wrist joints) 180 degrees/sec (Other joints)
Communication:	TCP/IP 100 Mbit: IEEE 802.3u, 100BASE-TX Ethernet socket & Modbus TCP
Programming:	Polyscope graphical user interface

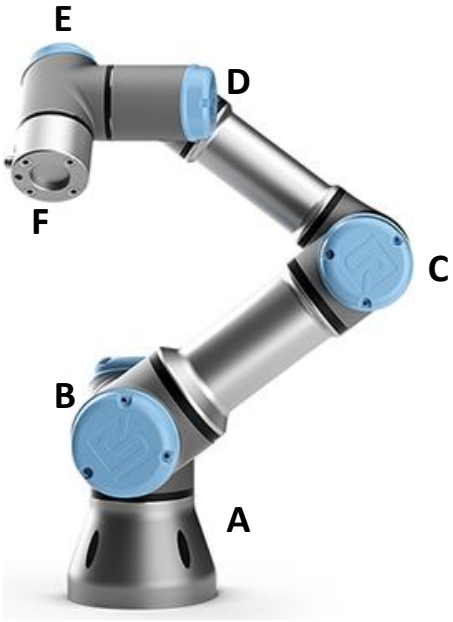


Figure 3.5: Design of UR3 and its respective joints

.

As can be seen, the robot range is 500 mm (depending on the initial position), this value is the first limitation for the system proposed, because there are cases where the range of the human arm is more extensive than the robot.

Each joint of the robot has a range of +/- 360º, so the UR3 can reach any position and angulation, as long as global limitations are respected. As stated, the robot has six joints, and not all are equal, they have different sizes, and this influences the maximum strength they are capable to execute [46]. The UR3 is formed by three different joint's sizes and, consequently, three distinct joint's torque. The Table 3.3 presents these parameters.

Table 3.3: Maximum joint's torque (UR3) [46]

Joint	*Size*	*Maximum Torque*
Wrist 3	Size 0	12 Nm
Wrist 2	Size 0	12 Nm
Wrist 1	Size 0	12 Nm
Elbow	Size 1	28 Nm
Shoulder	Size 2	56 Nm
Base	Size 2	56 Nm

The value of maximum torque of the joints presented is enough for this development because these limitations will not overcome when motion is being referenced in the tool attached to the UR3.

To develop ordinary industrial tasks, such as pick and place, robot programming is performed through a graphical user interface (GUI) called teach pendant. This interface is simplified to the user, presenting the essential functions and fast development of programming. In this work, besides the programming in Polyscope it will be insert a self-control algorithm developed in Python. The communication between the UR3 and this algorithm is performed through a MODBUS protocol, that will be addressed later.

At the end-effector of the UR3, there are two essential devices, the force sensor and a tool. The force sensor will be presented in Section 3.3.2. The tool is a 3D printed device attached to the force sensor, and the contact between the human and the robot happens through it. All movements, forces, positions and other variables will be referenced at the 3D printed device.

3.3.2 Force Sensor

The main device to catch the environment feedback is the force torque sensor FT 300 from ROBOTIQ. The UR3 is also capable of measuring the forces on its end-effector, however this measurement is much less accurate and therefore insufficient for our needs. The accuracy for this force measurement according to the manufacture is 3.5N [47].

The FT 300 sensor is easily connectable to the UR3 and capt the values of force and torque in the three Cartesian axes and orientation [48]. The relevant specification for this work are presented in Table 3.4.

Table 3.4: Technical Specification of ROBOTIQ FT-300 [48]

Specifications	*Fx - Fy - Fz*	*Mx - My - Mz*
Measuring Range	+/- 300 N	+/- 30 Nm
Signal Noise	0.1 N	0.005 Nm
Recommended threshold for contact detection	1 N	0.02 Nm
Data output rate	100 Hz	

This sensor is suitable to the development because the measures of the human force do

not exceed the maximum of the sensor. The value of the signal noise must be taken into account as the robot cannot initiate a movement if the read value is smaller than the noise. In fact, all values that not exceed the threshold of 1 N should not be considered as a contact. The output rate of the data exerts great influence in the data acquire, this means that the sensor only is able to capture 100 measures of force per second, this value will be later used to calculate the force sample size. The Figure 3.6 presents this force sensor.

Figure 3.6: FT-300 Force Sensor from ROBOTIQ [45]

.

Chapter 4

Development

In this chapter will be presented the solutions adopted to develop the simulation, the RL algorithms and the real system. In Section 4.1 the simulated environment to test the system with the self-control is presented. Section 4.2 shows the modelling of the proposed system using the techniques of Reinforcement Learning. Section 4.3 offers the communication between softwares to accomplish the simulation with the self-control provided by RL algorithms. Finally, Section 4.4 shows the implementation of the algorithms in the collaborative robot (UR3), and the communication between the self-control and the environment.

4.1 Simulated Environment

This section shows the development of the elements and codes composing the simulated environment used to test the insertion of the RL algorithms to the rehabilitation field.

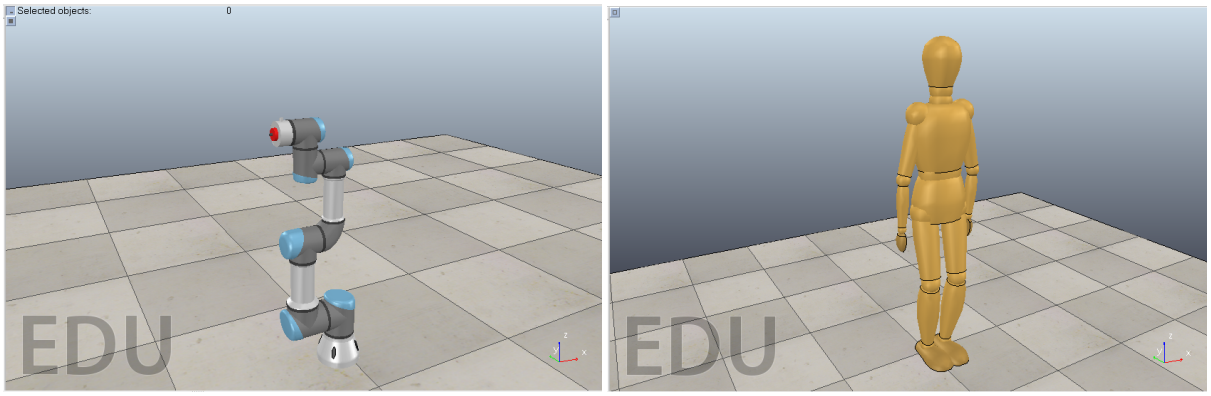
The proposed system was made in V-REP platform. V-REP is a simulation software that integrates a robot simulator with a development environment and can simulate robotic arms, mobile robots, sensors, tools and many different scenes. The Integrated Development Environment (IDE) is programmed in LUA language. V-REP also presents tools for communications with remote Application Programming Interface (API) as MATLAB [49]. Another important model in V-REP is the human body which can be simulated walking, actuating legs and arms [50].

The simulation has three essential parts: The virtual scenario with the robot and the manikin, the codes that control the simulation, and the human-machine interface. Basically, the components used in the simulation are made by mechanical parts and joints. The dynamic properties of mechanical parts that can be changed in the simulation are mainly mass, inertial

moment, and texture. The joints can also be configured, but its dynamic properties refer to movement, as maximum torque, target velocity and position control — all these features in V-REP results in a simulation very close to the real environment.

4.1.1 Virtual Scenario

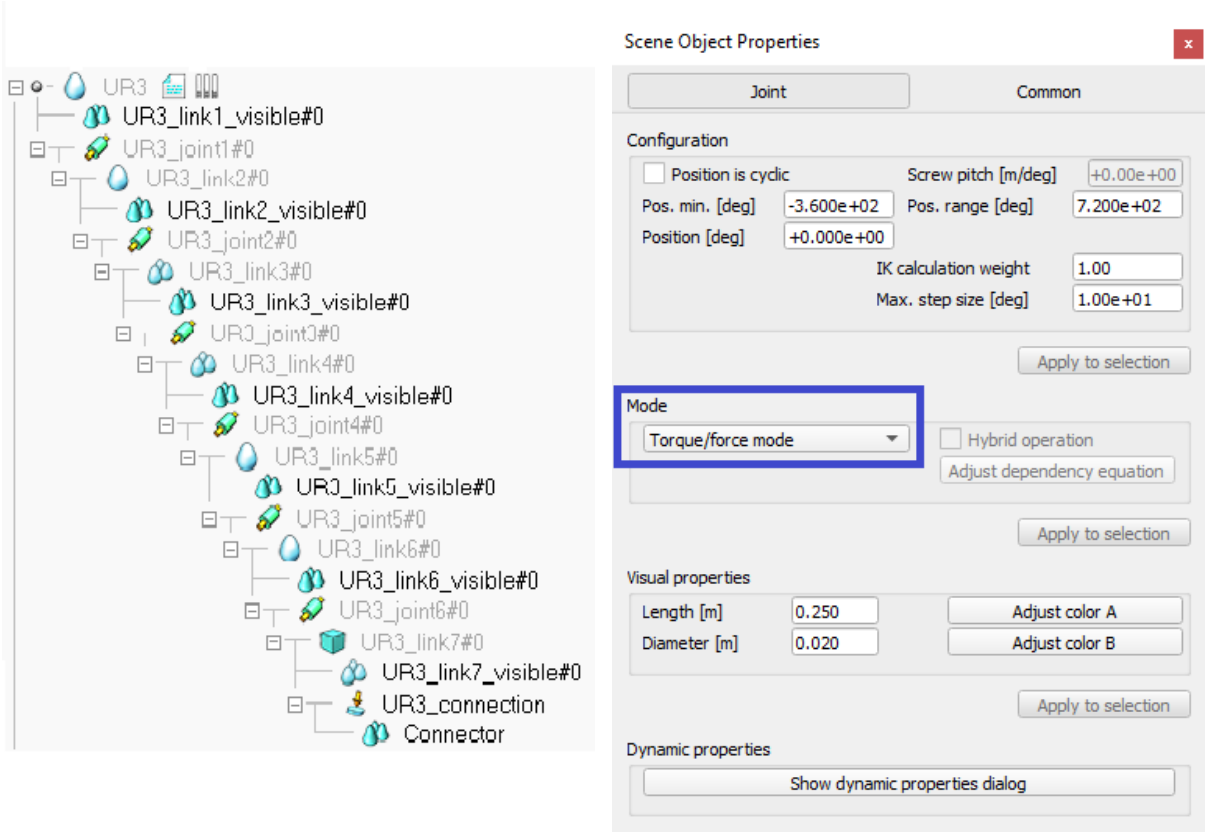
The virtual scenario was created to present the graphic part, where it is possible to visualise the models used to simulate the robot and the human. In this scenario, it is possible to relay all the dynamic configurations of the elements as joints and structural parts. The Figure 4.1a and 4.1b show the collaborative robot, and the human body structure inserted in the simulated environment. Both models are provided by V-REP platform.



(a) UR3 model **(b)** Manikin model

Figure 4.1: Models in V-REP platform

The simulated robot is the UR3 from Universal Robotics©. It has six joints that can be simulated separately and some mechanical parts representing the dynamic structure. The Figure 4.2a shows the active elements for the simulated UR3 model. Within the V-REP platform, it is possible to modify the properties of each structural element, however for the robot it is only necessary the modifications in the joints functioning, since the mechanical parts already had the same characteristics of the real robot. The functioning mode of the joint, configured in Torque/force Mode, was the only modified property, the configuration tab of joint's dynamic properties is presented in Figure 4.2b, with the blue box representing the configuration performed.



(a) Structural Elements of UR3 **(b)** Dynamic Properties Tab of a UR3's joint

Figure 4.2: UR3 Configuration

At the end-effector of the UR3 is attached a force sensor that will show the exact force of the human arm during the entire simulation. Notice that this sensor is represented in Figure 4.2a by the *UR3_connection*. The element called *Connector* has the function of connecting the UR3 to the human arm and was created through a CAD design, it has specific properties (Mass value near zero) so that the simulation is not influenced.

The used Manikin is a particular case because only the human arm is used, then, the essential joints for this approach are those that represent the shoulder. The dynamic properties resemble those made on the robot joints so that the functioning will be in the Torque/force mode. For the Manikin, the structural parts must be configured to approximate the simulation to a real case. Therefore, the data used comes from a study accomplished by Ribeiro, Estivalet, and Loss [51] where is presented a model to estimate the force and the torque during the abduction of the shoulder. The experiment consists in the execution of the abduction

movement by the patient and the measure of the performed force through a dynamometer. The considered patient in this experience is a man with 1.78 m and 85 kg. In this case study, the maximum torque measured was 41 Nm. In the simulation, the system was prepared to receive these values. Thus, the dynamic property configured in the Manikin was the weight to approximate the simulation to the real case.

Finally, the Figure 4.3 shows the complete virtual scenario with all pre-configured elements. The human arm is connected to the robot through the connector, and, because of the joint's settings, when the human arm starts a move, the UR3 will follow the movement.

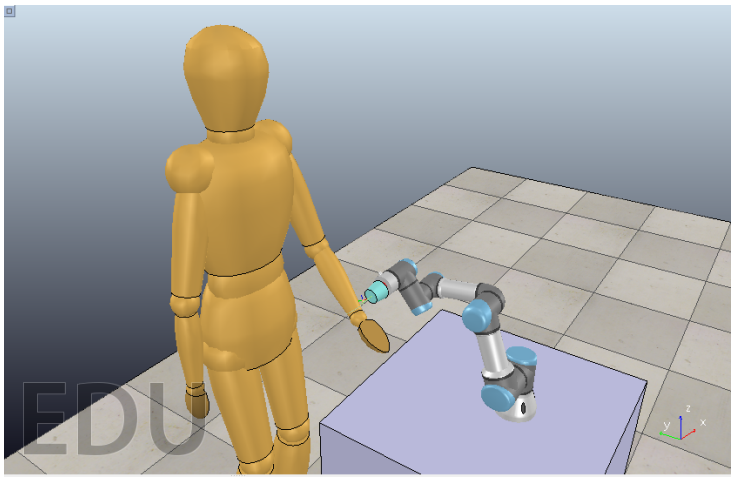


Figure 4.3: Complete virtual scenario with all elements

.

4.1.2 Simulation Codes

The codes were developed in LUA language and were divided into two main scripts, one called by threaded child script and other called by non-threaded script. The main difference between both scripts is the mode of each on runs during the simulation. The non-threaded child script runs just once after the start of the simulation and when it is called by other function, such as the press of buttons. In this script, we find the codes referring to the development of the human-machine interface (HMI) and all functions linked to it. The threaded script runs in loop during the entire simulation, and this script contains the codes about the movement controls, the measured force, the messages that must be sent to another script, and the API

connection.

The Figure 4.4 may be used to illustrate the operation and function of both scripts. It is worth remembering that the simulation is performed through small steps (100ms). Shorter step times could be used, but computational expense would be very high. In the illustration, a user inserts the configurations through the HMI, this data is read by the non-threaded child, sent to threaded child script, and these settings finally are used in the simulated elements. In each simulation step, the elements present a response from the environment that returns via scripts and is displayed in the HMI.

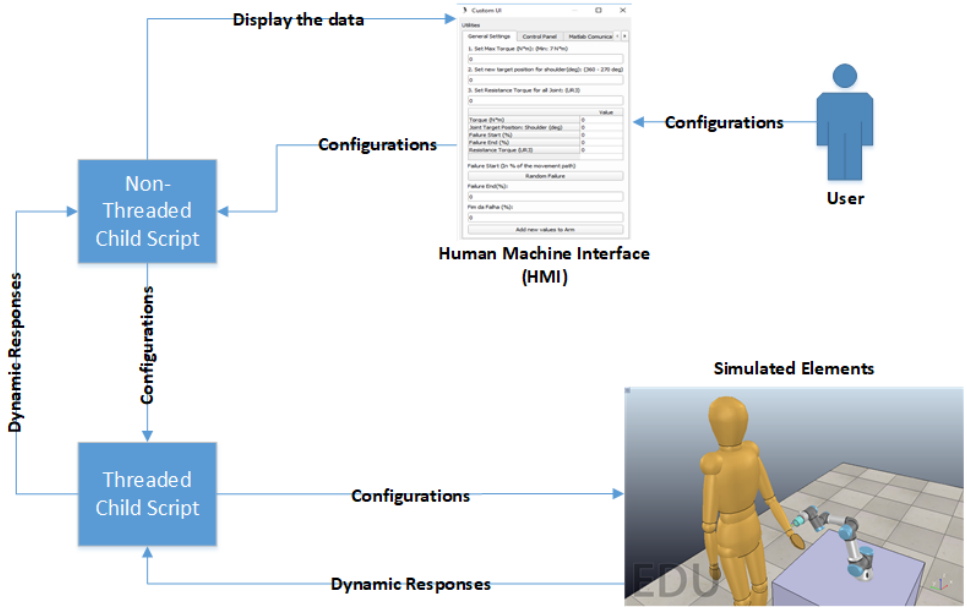


Figure 4.4: Operation of non-threaded and threaded child script

.

The operation shown in Figure 4.4 only represents the simulation being executed using the parameters that are set by the user, and in this mode, the robot resistance to the human movement is changed according to the wishes of them. However, the use of simulation has a second purpose, which is to test the RL algorithms, this latter function can be accessed by the user through the HMI and will be presented in Section 4.3.

4.1.3 Human Machine Interface (HMI)

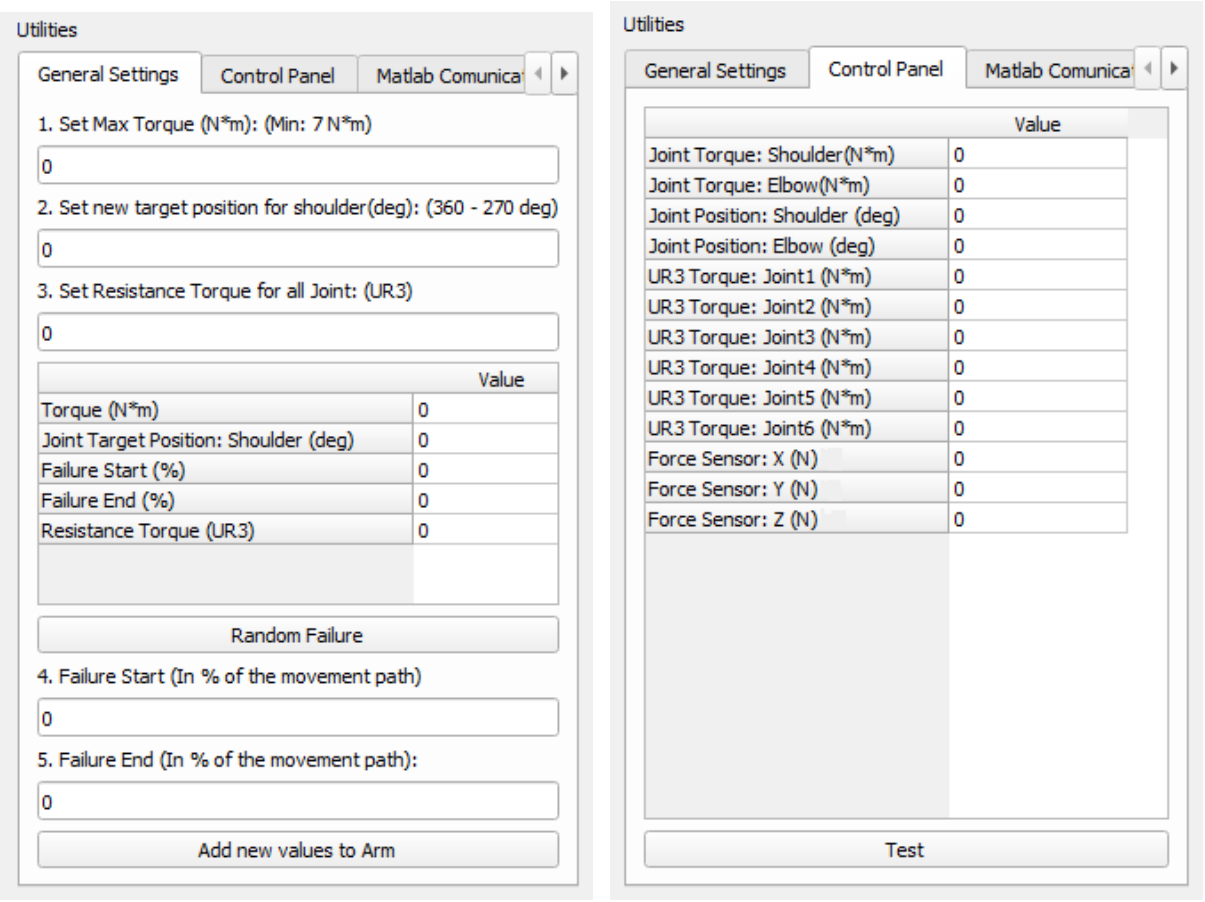
The HMI was configured in XML. Its functions refer to the configuration of the main parameters in the simulation, also to display the feedback from the environment, and to insert the self-control algorithm.

The HMI was divided into three tabs. The first and second tabs are used when the user gives the configurations about the movement. However, the third is used to configure the communication with the external API (MATLAB) and insertion of the RL algorithm. In Figure 4.5a and 4.5b are shown the tabs "General Settings" and "Control Panel", sequentially. The first tab configures the main parameters linked to human arm movement (1 and 2), the resistance torque of the UR3 (3), and the possible variation of the patient force, which can be random (button "random failures") or given by the user (4 and 5). When the button "Add New Values to Arm" is pressed, the simulation starts and the table is filled with the configured parameters.

The second tab, called "Control Panel", presents some results of the movement as torque of the robot's joints and torque of the human arm, and the data measured by the force sensor. The button "Test" is used to perform a test in the system with the following parameters: Human arm torque (30 N.m), shoulder target position (300 degrees), UR3's resistance torque (2 N.m), and without failures in the movement.

The third tab called "MATLAB Communication" is responsible by the communication with the external API, and is presented in Figure 4.6, basically when the button "A" is pressed the simulator can send and receive signals from MATLAB. If the button "B" is activated, the V-REP send a request to the external API to initiate the self-control module. The table showed in tab only presents the information about the communication. This operation will be better commented in the Section 4.3.

With the functioning of the simulation explained and its dynamic properties performed, it is possible to start developing the autonomous control algorithm that will be tested in the simulation to control the resistance force that the UR3 will provide to the movement.



(a) "General Settings" Tab **(b)** "Control Panel" Tab

Figure 4.5: Simulation settings - Tabs 1 and 2

4.2 Modelling the System as an RL Problem

As stated in Section 3.1, a modelling of the problem must be accomplished to enable the insertion of the RL algorithm. For this, it is essential to know the parts that make up the system. Notice that this modelling is used for both simulation and operation in a real environment.

4.2.1 Agent and Environment

The first element to consider is the agent that will interact with the environment, the UR3 collaborative robot. The model of the robot is presented in Figure 3.5 and its limitations are shown in Section 3.3.1. The essential feedback measured from the environment is the force value collected through the force sensor. Besides, can also be acquired the data about position

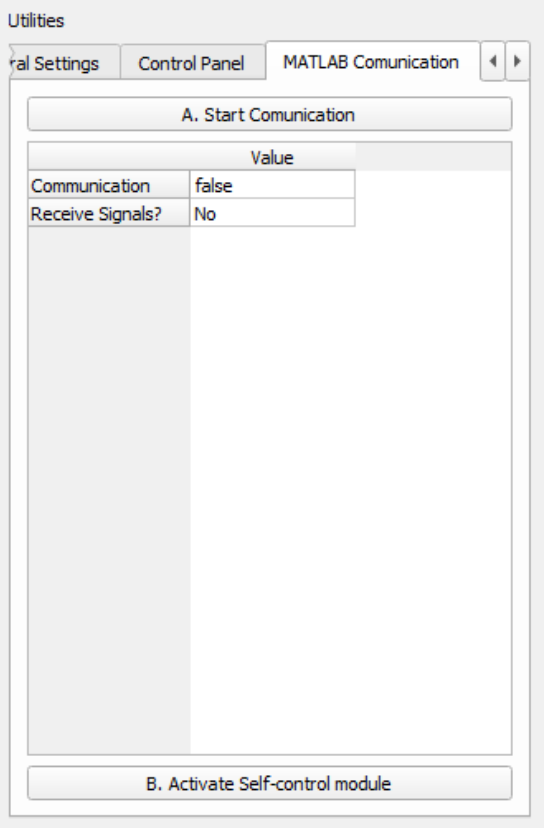


Figure 4.6: "MATLAB Communication"

.

and velocity for each joint, and end-effector.

Environment modelling is a bit more complicated and should take into consideration the goals of this work. This environment can be imagined as the interaction between the human arm and the agent. Using the MDP framework is possible to study this problem as a set of states S that can be reached using another set of action A performed in each state.

The objective of the self-control algorithm is to provide a resistance to the force performed by the human arm. On the robot's side, the states must be linked to the effects produced by the UR3, i.e., each value of force performed will form a different state. However, this approach is not sufficient to describe the system as a whole because the data about the patient force is neglected. Each state can also be imagined as a value of the set of forces performed by the patient, but still, some of the environment information would not be used. To solve this problem, the state approach in this works considers the force in the patient's and the robot's

side. Therefore a state will be described as a force pair P_f, R_f, to ensure that all information were used.

The approach described to modelling the environment disregards the position of both UR3 end-effector and human arm, and this represents a benefit to the system because there is no more dependence on this variable. In fact, if the position was also a control variable, the states would have five dimensions $P_f, R_f, X_{pos}, Y_{pos}, Z_{pos}$, which would represent a considerable computational expense.

4.2.2 Actions, Policy and Reward Signal

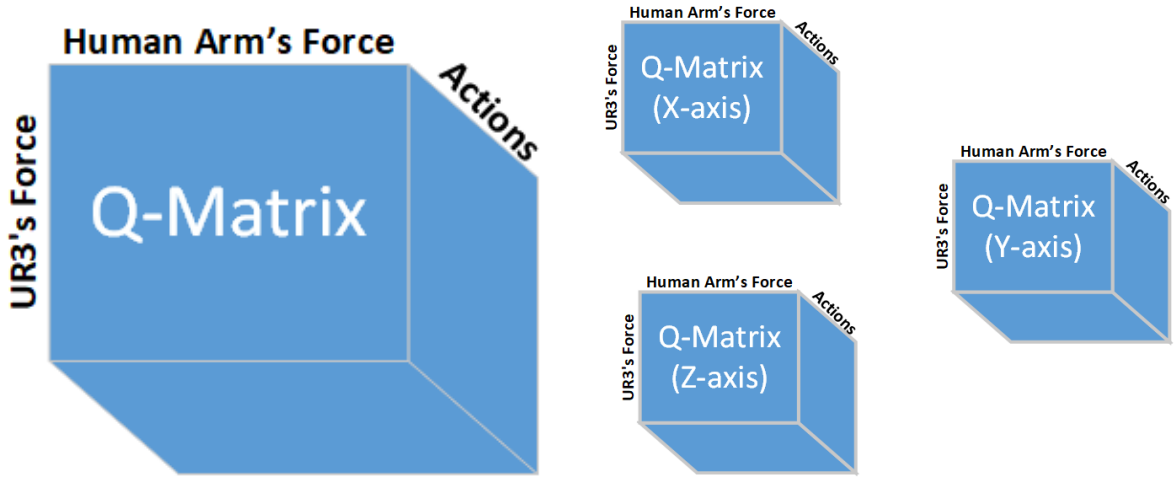
The goal of an RL algorithm is to perform a better decision taking into account the current state of the agent. This decision is chosen through the policy, that consider the reward signal as feedback for the system, and refer to the next action that the agent will perform. So, it is appropriate to study the modelling of action, policy, and reward signal together because these three concepts complement each other.

Before the explanation about these three concepts, it is important to clarify what will be the values of the Q-matrix. As stated in Section 3.1.5, the Q-matrix correlates the states with the actions, in this work this matrix will be three-dimensional P_f, R_f, a because the states are presented as a force pair. For a linear movement only 1 Q-matrix will be used to describe the behaviour of the system, but when the movement performed is in a 3D space will be used three different matrix to present the behaviour to the three axes. Figures 4.7a and 4.7b presents an illustration of these two approaches for the Q-matrix used in this work.

If the method of three matrix was used, another matrix should correlate these and will be described later.

In this work, the agent's goal is to provide a resistance force to the movement. Thus the actions that will be modelled to this problem are represented by the possible decisions that the system can make: increase the resistance force, decrease the same resistance or hold this value — remembering that these forces are always referenced in the UR3's end-effector.

The update of the Q-matrix is based on the states, the actions described, and on the reward



(a) Q-matrix to the linear movement

(b) Q-matrix to the 3D movement

Figure 4.7: Two approaches for Q-matrix used in this work

signals. These reward signals are assigned using the environment feedback, and in this case, these data are the force collected by the sensor. So in the first iteration, the Q-matrix start with all values equal to zero, and thus a random action should be chosen. When the first action is chosen, an evaluating of this choice happens by means of a reward and thus changing the value inside the Q-matrix.

The assign of this reward should be based on the expected behaviour of the system. As the work focuses on the improving of the patient's force, the reward should be assigned when there is a detection of increases in the patient's strength. However, perceive this improvement is not easy, so to get around this situation, a previous test of maximum force of the patient's arm will be performed.

Making the UR3 static and asking the patient to perform the maximum possible force in the XYZ-axes will enable the system to get this value for each axis. Then, this force will be recorded in a variable, shown to the therapist, and used to award the reward to the agent. Another parameter that will be configured by the therapist is the expected force of the patient during the motion. This will be done by indicating the upper and lower limits, in percentage of the force measured in the first test. All these parameters and values will be shown in the web interface. In this work, as an example, the minimum and maximum limits will be taken as 70% and 85%.

The policy used to accomplish the decision is called ε-greedy policy, exemplified in pseudo algorithm 3, and uses the Q-matrix values to choose the next action, in the first iteration this action will be often randomly chosen, but after a certain number of episodes the decision will make based on the best action for the system. It can be predicted that the action will be: increase the resistance when the force of patient is larger than the resistance, decrease in the opposite situation and hold the force when the force sensor reads a value between the range established by the therapist.

With the modelling of the problem, the RL algorithms may be implemented to solve the problem of this work. Both algorithms were used to collect results in the simulation and in the real case.

The algorithms applied to the Rehabilitation problem in this work are the Q-Learning and SARSA Learning, an off-policy and an on-policy algorithm, respectively. The Figure 4.8 illustrates the operation of both. Notice that one of the differences between the first and the second algorithms is one more process in the second. The other difference is in the update Equations (3.4) (SARSA Learning) and (3.5) (Q-Learning). The first takes into consideration the next action to the update.

4.3 Communication Between MATLAB and V-REP

This section will present the communication between the MATLAB (Remote API) and V-REP software, in the implementation of the self-control in the simulation. The development of this communication is valid for the two algorithms implemented. First of all, it is essential to analyse how the simulation will proceed in terms of inserting autonomous control. For this, a flowchart representing the whole control of the virtual functioning is shown in Figure 4.9.

The communication established between V-REP and MATLAB operates in synchronous mode [52]. Basically, in this mode, the simulation in the platform only runs after receiving a request from the API. Thus, a trigger should be created in the MATLAB to inform the V-REP that the simulation may start. The Figure 4.10 shows a general operation for the synchronous mode. Notice that only after MATLAB sent the trigger, the simulation remains running during

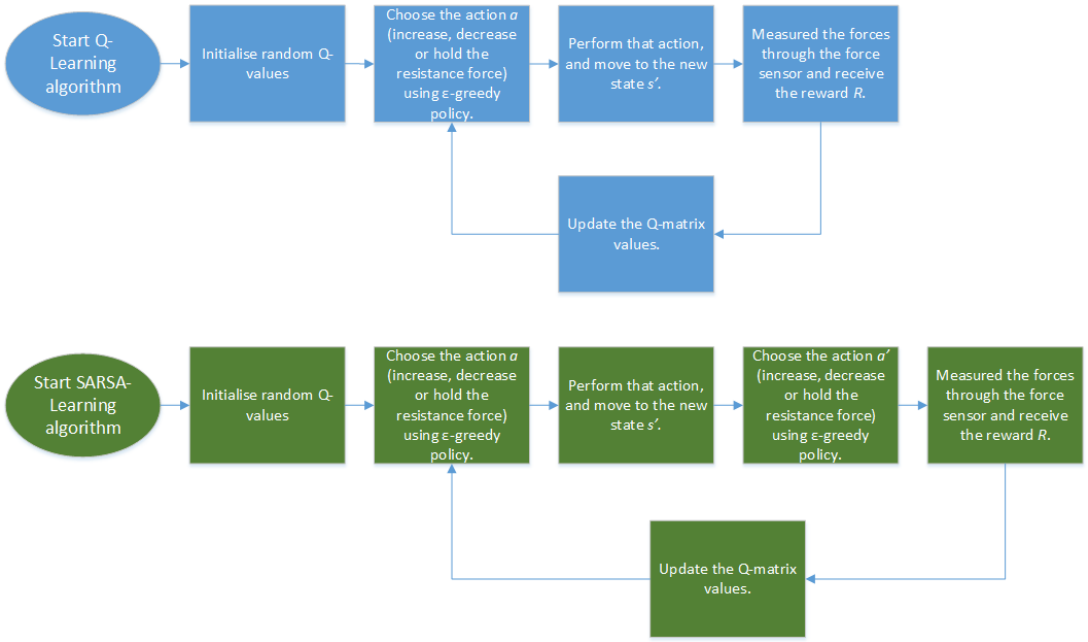


Figure 4.8: Block diagram of both algorithms Q-Learning and SARSA Learning

.

one step (100ms in this case). The adopted step simulation was 100 ms because this is the smallest value the system can perform processing without spending so much time. Other values could be chosen, but if the simulation step was longer, some system characteristics could be lost. On the contrary, if the simulation step was shorter, excessive computation times would make the system implementation impossible.

All codes used to accomplish the simulation in Manual Mode and Autonomous Mode were developed in the LUA and MATLAB language. However, when the approach is on the real robot, these codes should be adapted.

4.4 Using the Collaborative Robot (UR3)

This section addresses the implementation of the self-control algorithms in the collaborative robot. The details of the developed control in Python will be: the communication between the control system and the UR3, the essential problems found in the measure the data, how was the collection of samples, and the issues linked to the resistance force provided by the

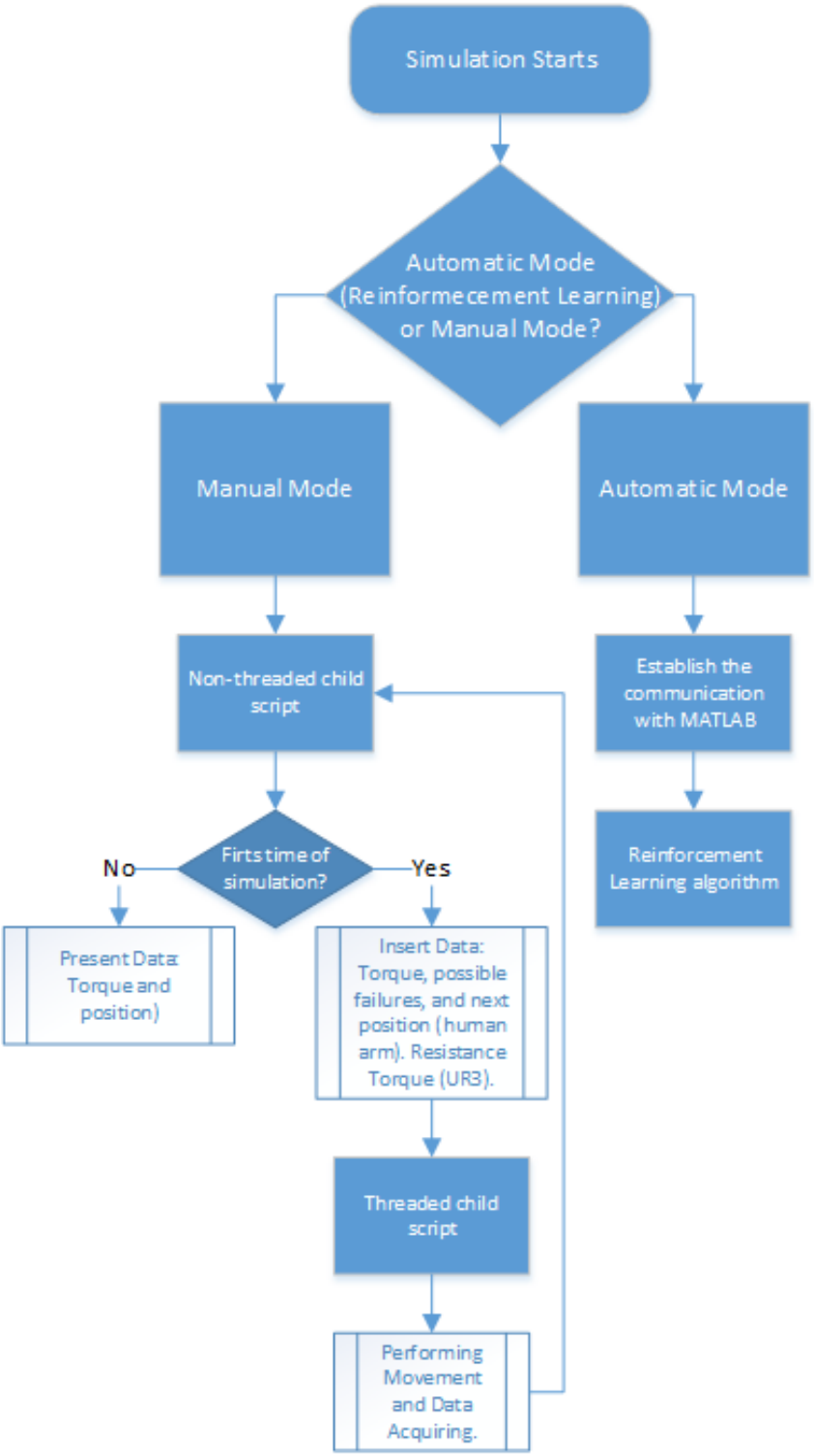


Figure 4.9: Flowchart that represent the control implemented on V-REP

.

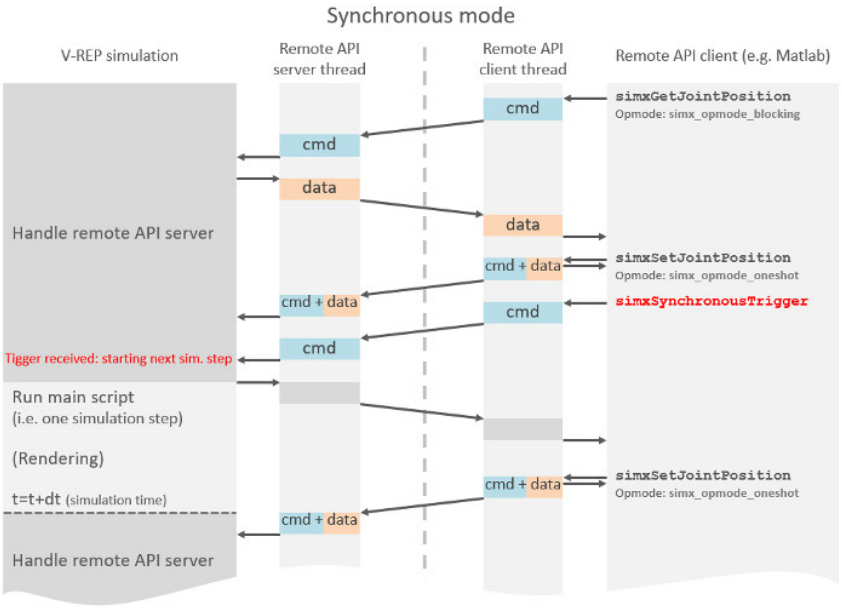


Figure 4.10: Synchronous Mode bettwen V-REP and an external API [52]

.

robot. The Figure 4.11 shows the architecture of the proposed approach using the real robot.

The item “1” in figure 4.11 represents the user who interacts with the robot by moving his arm, resulting in the application of forces on different Cartesian axes (X, Y, Z) acting on the robot tool. Item “2” is the robot itself, which is a collaborative manipulator. This manipulator is equipped with a force-torque sensor FT-300 (item “3”), capable of measuring force in the X, Y and Z directions of the Cartesian plane and orientations. Coupled to the force sensor is a cone-shaped 3D printed tool (item “4”) where the user can hold and perform movements with the robot. This printed piece has high rigidity, able to withstand the force exerted by the user on the robot, this iteration of forces between the robot and the user is represented by item “5”. The item “6” represents the Reinforcement Learning (RL) algorithms (Q-learning and SARSA), which, depending on the movement performed by the human, will adjust the resistance to movement with the rehabilitation manipulator. Finally, item “7” indicates MODBUS TCP communication between the robot and the computer running the python application. The robot sends the force data applied to all axes, which is processed by the program, which returns the resistance force that the robot will apply in the iteration with the user.

It is also worth to remember that the control algorithms developed for the real system

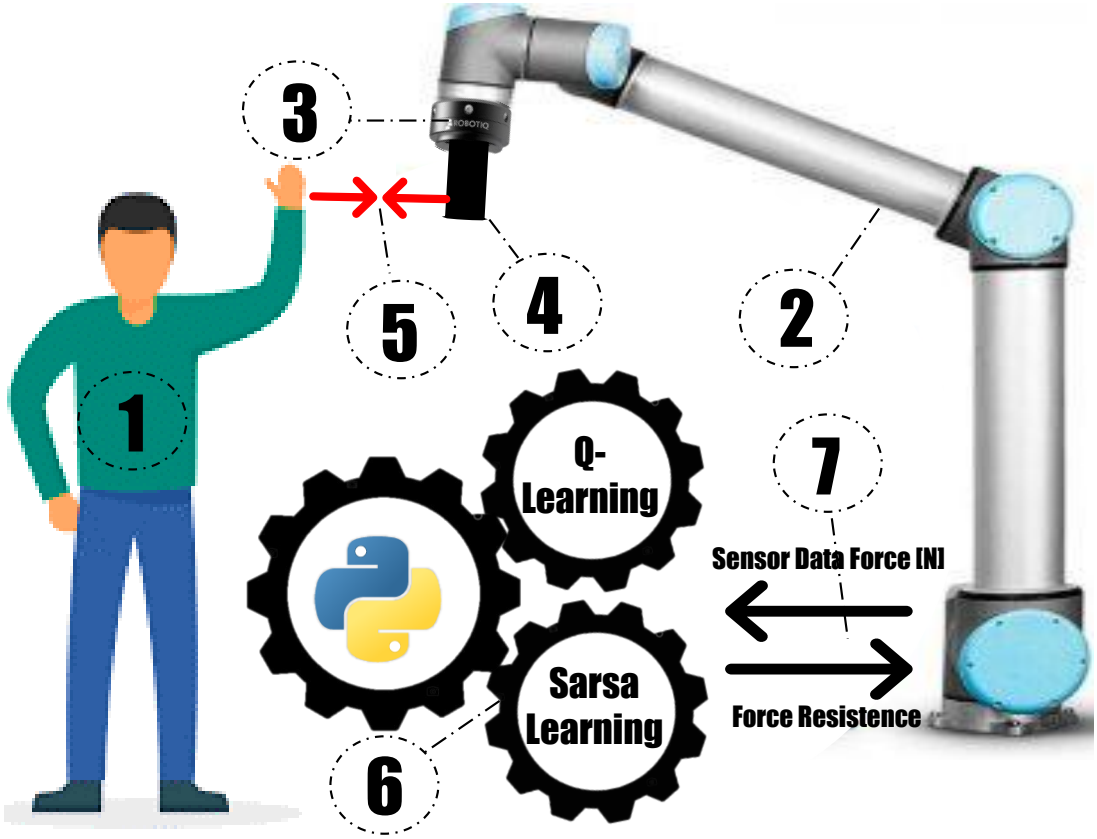


Figure 4.11: System Architecture of proposed approach.

only concern to the autonomous mode. Manual mode will not be used to control the real robot, unlike simulation.

4.4.1 Development of the self-control in Python

The self-control developed in Python using the same algorithms presented in the simulation (Q-learning and SARSA). Both algorithms was implemented separately and their results compared.

The MATLAB was used to implement the algorithms in the simulation part because the V-REP presents a synchronous communication that can be applied with this software. However, when the real UR3 is used is more simple to use the control of the collaborative robot through an own platform the “Polyscope GUI” that uses the touch screen teach pendant with a proprietary programming language as shown in Section 3.3.1. The robot, through its own

programming language, allows the communication with a remote API through a MODBUS protocol.

The use of the MODBUS protocol requires a quick response from the external API because the data acquired by the robot will be sent to this and a return response will go back to the UR3, in real-time. Despite being an excellent tool for development, the use of the MATLAB in real-time requires an excessive computational expense. This is a huge problem in this case, because, besides the real-time processing of the information measured by the robot, the algorithm must still make the calculations with three-dimensional matrices as presented in Section 4.2.

Then, the chosen language to develop the RL algorithms was Python 3.7.4. This is a high-level language programming, easily implemented and widely used to data science and Artificial Intelligence (AI). Besides, there are numerous developed applications available on the Internet using Python to control the UR3 through the MODBUS protocol. Another benefit is the various modules that can be installed to provide an easy programming, in this work the essential modules used are: "*pyModbusTCP.client*", "*numpy*", and "*random*".

The MODBUS protocol is used by the UR3 to perform the communication through a TCP/IP protocol. Therefore, this robot can send or receive messages through an "'intranet" (private network) or "Internet" (public network). The MODBUS of the UR3 has 16-bits ports, some of these ports only can be used to acquire specific data, and are not configurable. The description of these specific MODBUS ports are presented in [53]. Other ports can be used to obtain or send variables created by the user to control the system.

The main element to measure the force performed by the patient is the force sensor attached at the end-effector of the UR3. This sensor is explained in Section 3.3.2. This sensor also communicates through a port (63351) and provides the variable for the three Cartesian axes. The MODBUS ports used in this work will be presented forward in the Table 4.1.

Table 4.1: Used MODBUS ports

MODBUS port	*Description*
128	Send Resistance Force
129	Control Aux
130	Status Port
131	Force Signal
400	Position in X
63351	Sensor Force

4.4.2 Force Measurements and Resistance provided by UR3

As explained in Section 3.3.2, the force sensor will be the main element to acquire the data about the patient's arm. The sensor is capable of measuring the forces and torques in the three Cartesian axes, totalling six different values to describe the movement. However, only the three values linked to the force will be used.

Two experiments were accomplished, the first executing a linear movement in only one axis and the second in the 3D space. For linear motion, one axis will be in free drive and the other two locked. It is intuitive to think of working with only the force measured in the free axis. However, even with the axes locked, there are measurements of force in the three dimensions. This forces should be considered in the algorithm because they carry a piece of movement information. Therefore, to get around this problem, the resultant force is used by the algorithm as the parameter to award rewards and calculate the Q-matrix. The second experiments use the forces in each axis to calculate the corresponding Q-matrix.

When the sensor recorded a force, the algorithm uses this value to decide what will be the new resistance force provided by the UR3. The insertion of this resistance force in the robot is accomplished using the command of force mode (own UR3 command), shown in Figure 4.12. This command is found in the manual of the URScript Programming language [54].

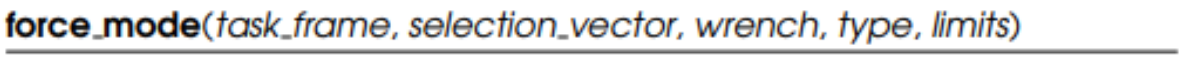


Figure 4.12: Force Mode Command [54]

.

The command presented in Figure 4.12 depends on some parameters, the first called

"*task_frame*" is the pose vector reference of movement, in this work, the vector contains the values of the tool installed at the force sensor to facilitate the human grab. This tool showed in Figure 4.13 was not developed in this work, but was taken from another previous thesis. The pose vector refers to the positions and rotation in Cartesian axes. The second parameter is the "*selection_vector*", is a six-value vector where is possible lock or unlock a linear axis or a rotation axis. The values of these variables are X, Y, Z, Rx, Ry, Rz, if the value is 1, the axis is accessible to the movement. Otherwise, the axis is locked. The other parameter is the "*wrench*" that represents the forces performed in each linear or rotation axis. The parameter "*type*" basically choose the way that how the force will be interpreted. The final parameter "*limits*" will represent the maximum allowed speeds for the unlocked axes and the maximum deviation for the locked axes.

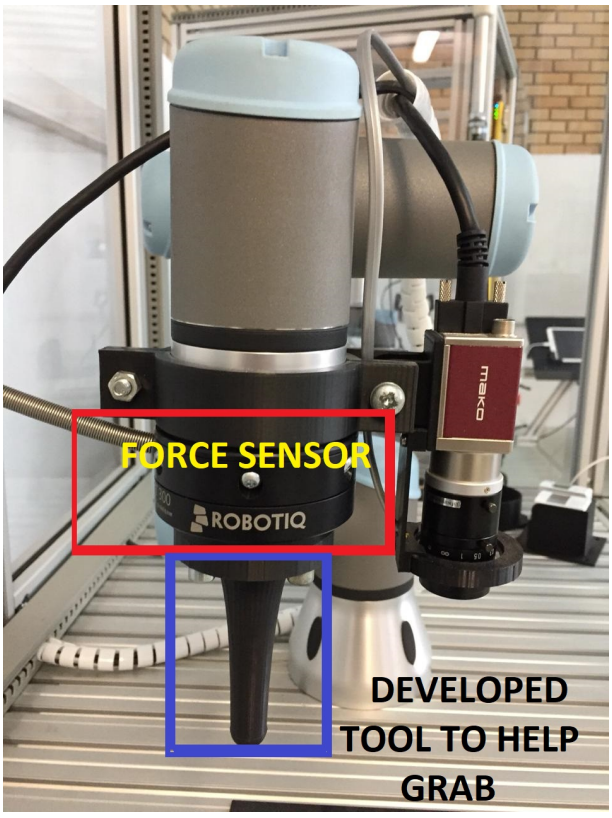


Figure 4.13: Tool to help the human hand grab

.

The parameters configured for the force mode depend on which axis will be in free mode, if just one linear axis is unlocked, so just the force in this axis should be modified. Otherwise,

the set will be in the three forces. The operation of this function is interesting because the values for the forces performed in the axis can be changed dynamically, perfectly fitting the problem.

The decision performed by the system refers to the absolute value of the resistance force. However, depending on the direction of the movement the value recorded by the sensor will be negative. The negative and positive values indicate the direction of the movement. As the resistance was set positive in the first iterations, the UR3 had the desired behaviour only in one direction. To solve this problem a function in the algorithm to recognise the direction of the force measured by the sensor and compensated in this in the resistance was inserted. So, when the force measured is positive, the resistance will be negative, otherwise will be positive.

The force sensor is the most important element to measure the environment feedback. However, the continuous use of this sensor results in the calibration issues. Therefore, if the sensor is calibrated at this time, probably after three or four iteration, it will not be presenting reliable values. To fix this problem, at the beginning of all iterations a command is sent to the UR3 to reset the values of the sensor. It is important that the calibration is performed when there is no applying force to the sensor.

When the command "*force_mode*" is employed the robot exert a force in one or more axes against the human force. However, if the human initiate a execution of some exercise and drop the tool of the UR3 in the middle, probably the robot will initiate some movement without a restriction (initially provided by the human hand) in the same direction as the last force command that the self-control algorithm provide. To solve this problem another function was implemented, when a force around or equal zero is detected, the algorithm starts counting until a preset value, if a force is detected by the sensor before the end of the counting the system keep running normally, otherwise, when the preset value is reached the resistance force is set to zero and the UR3 stops its movement.

4.4.3 Force Signal Sampling

To acquire the force signal from human arm the force sensor is used. However, even with a good calibration, some signals measurement may be wrong due to, for example, dynamic alteration of the direction, hand grip weakness, random measure errors, and noise signal. Therefore, the strategy is to get samples of force signal and average these values. But to use this tactics, it is relevant to think about the sample size.

As the system will operate with a human, the resistance force must respond in real time the force of the patient. So, if the system takes a long time to calculate the decision or collect the samples, the human will realise this phenomenon. So, the sample size must be limited to the human's ability to perceive touch, according to a database presented in [55], the touch reaction time is about 155 ms. So, the samples must be collected and processed in a less time.

Also, the signals have a maximum sampling frequency that is limited by the sensor or MODBUS communication. In this case, the limitation is provided by the force sensor output data which has a speed of 100 Hz. This means that every 10 ms a signal is collected by the sensor and sent to the algorithm. Other value that should be taken into account is the time that the self-control algorithm waste to make the decision.

$$t_{hp} \geq n_{sig} t_{samp} + t_{processing} \tag{4.1}$$

Therefore, the goal is to change the force without the human realising. This limitation is considered in the Equation (4.1).

Where, t_{hq} is the time of human perception or 155 ms, n_{sig} is the number of signals that compose one sample, t_{sig} is the time to collect one signal or the time to sent the signal via MODBUS protocol, and $t_{processing}$ is the time spent to calculate the new decision.

So, to calculate the value of n_{sig}, the value of $t_{processing}$ should be known. However, in the best case, the computational expense may be considered too small and near by zero. Therefore, the Equation (4.1) could be disregard the $t_{processing}$ element. Thus, the number of signals that can compose one sample can be calculate based in the parameters established.

$$n_{sig} \leq 15. \tag{4.2}$$

The maximum number of signals that compose a sample will be 15. And this value will be used to measure the results. All elements of the implementation in the real system will be explained, thus the real robot used in the real environment is shown in the Figure 4.14. The camera that appears in the image is used in other works that use the same robot.

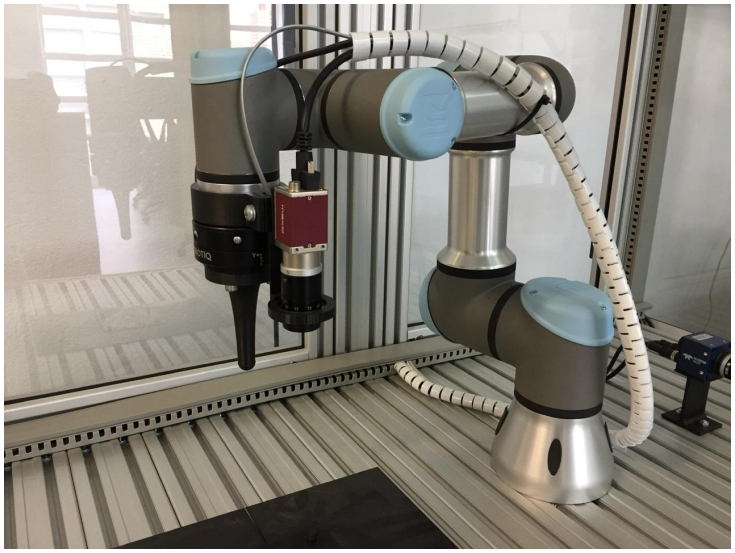


Figure 4.14: UR3 in the real environment

.

Chapter 5

Results and Discussion

This chapter presents the essential results obtained through the developed topics explained in this thesis. At first will be showed the results referring to the simulation of the system with the self-control running or not. After that, the results about the training of the real system will be presented and compared with the simulation outcome. Finally, it is exhibit the simulation of the model of the human arm presented in the Section 3.2, and the results are compared with the one axis experiment with a healthy person.

5.1 Simulation Results

All parts connected to the development of the simulation environment and its functioning will be showed in this section that is divided in the presentation of the developed graphic part, the force measurements acquired in the simulation and the values obtained with the insertion of the RL algorithms.

To test the simulated system through the interface, the values considered in the arm are provided from [51], thus the maximum torque of shoulder is 41 N.m. The failures in the movement are set to occur in 17% of the path and stop in 42% and are presented as a decrease in force that the shoulder is capable to perform. A value of 6 N.m is configured to be the maximum torque that the robot joints can execute in opposition to the movement. Figure 5.1 show the configured values in the HMI tab (presented in Section 4.1.3). These values will be the reference for all results presented in simulation without the self-control algorithm insertion.

After setting the values the simulation can be running to test that determined situation of forces, positions and failures. The simulation environment is presented in the Figures 5.2a

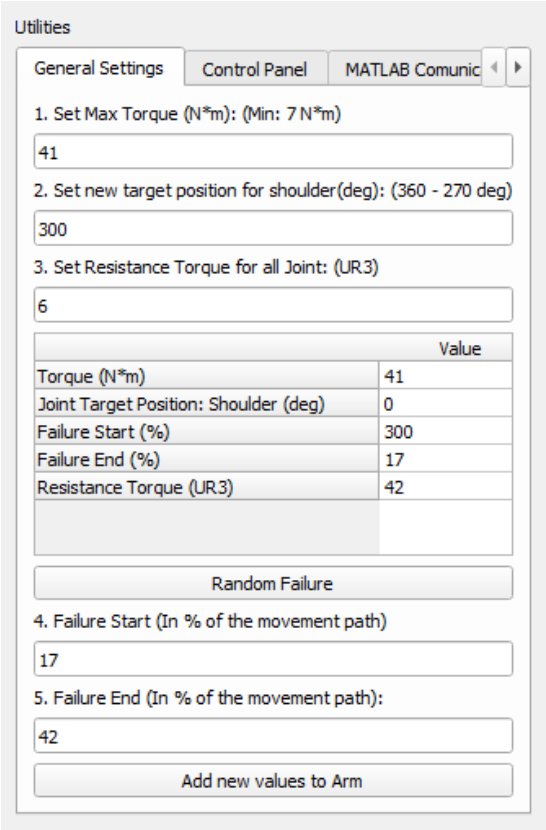
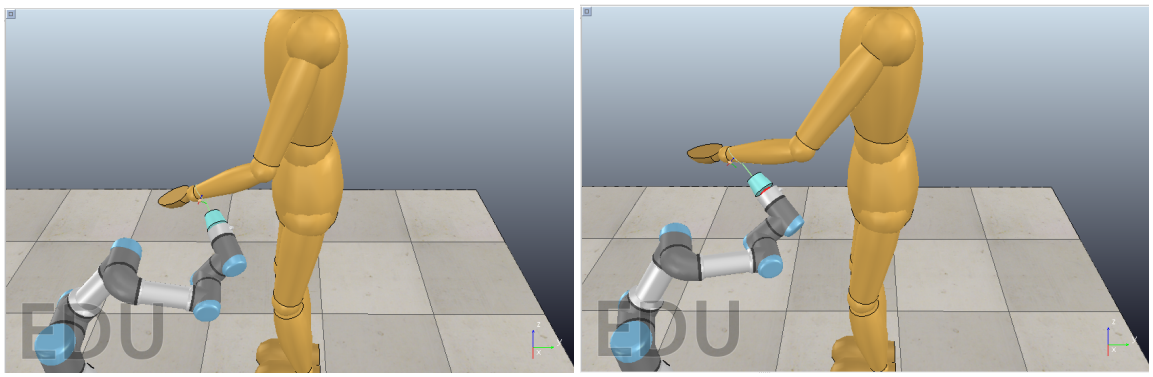


Figure 5.1: "General Settings" Tab with the values configured (simulation)

.

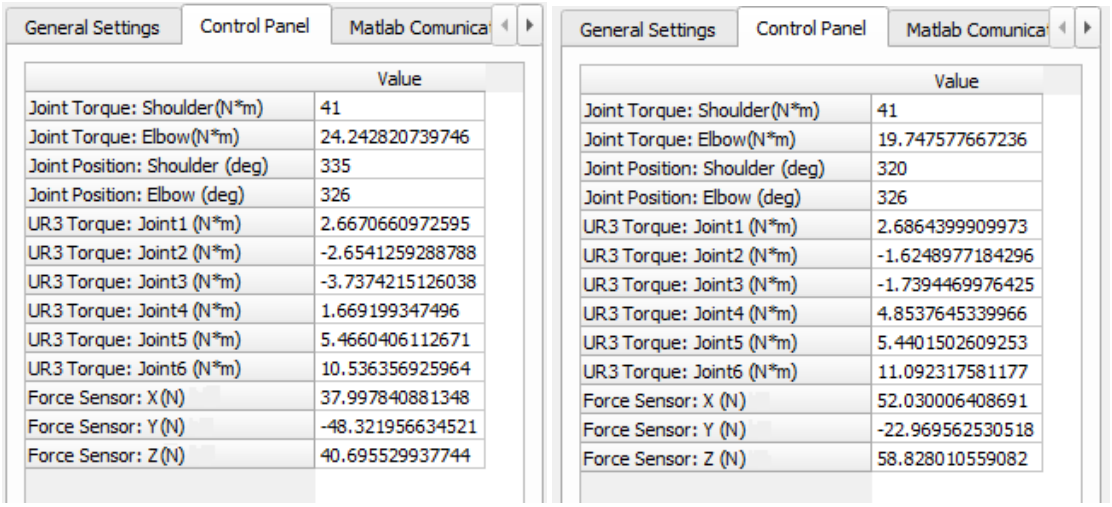
and 5.2b. Notice that the following figures presents the movement in 40% and 67% of the proposed system path, respectively. The proposed system path represents the go forward shoulder movement.



(a) Simulated System - 40% of the path **(b)** Simulated System - 67% of the path

Figure 5.2: Simulated Movement of human arm

While the simulation running, the “Control Panel” tab record the values of the movement in some points of the system. This data can helps the physiotherapist to analyse the simulated situation. As an example the Figures 5.3a and 5.3b shows this values in the same conditions of Figures 5.2a and 5.2b.



(a) Data - 40% of the path **(b)** Data - 67% of the path

Figure 5.3: Data related to movement

As expected, the Shoulder torque remaining in 41 N.m. However, in the elbow reference, this variable changes according to the arm’s position. It is interesting analyse the UR3 torque individually, the simulation shows that hardly it exceed the 12 N.m, even with the arm applying a larger value of torque. Finally, the most significant value is that measured by the force sensor. As the arm move around the three axes, it can be seen a considerable value throughout the motion in each axis, and the values are fluctuate according to the arm’s position. So, a value that could represent the force exercised by the entire arm is the resultant force calculated with the force in each axis measured by the sensor. This force value is showed in the Figure 5.4. Notice that there is a great variation in the curve, this represent the failure between 17% and 42% of the movement trajectory.

The curve presented in Figure 5.4 represents only a specific situation. However, when the objective is to provide forces according to a variable human arm force some control should be implemented. In this case the control is performed by an RL algorithm, this means that, after some training, the network is able to calculate and deliver this value linked to the human

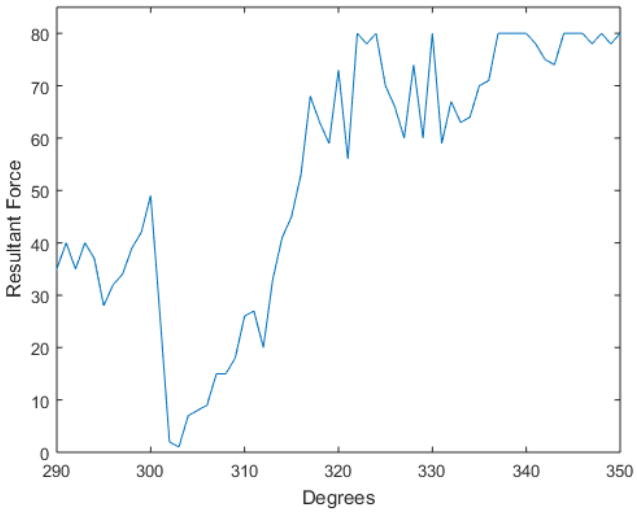


Figure 5.4: Resultant Force (Force Sensor)

.

force.

As soon as the self-control module is active, the human arm will execute the movement of raise up, using the shoulder as the main joint. The movement of the human arm pulls the robot attached in the wrist. However, the raising up of arm suffers a resistance force provided by the link with the robotic system, and this value is acquired and sent to MATLAB.

With the data acquired to the contact with the patient, the RL algorithm updates its values and find a better resistance force that the UR3 should perform, then this information return to V-REP through the synchronous communication and the UR3's maximum torque is changed. This process occurs in loop until the simulation stop or the maximum number of iterations in the RL algorithm is reached. This value of resistance is given as a percentage of maximum force that human arm can execute.

In the first episode, the behaviour of the measured force is not constant and can vary a lot, however after a few number of episodes this force tends to be constant. In Figure 5.5 the acquired force is shown after a few number of episodes.

As mentioned before, the RL insertion of the autonomous mode works with the update of a maximum torque in the collaborative robot in order to correct the resistance force. This update is shown in Figure 5.6, and as expected in the first episode, the torque increase until a

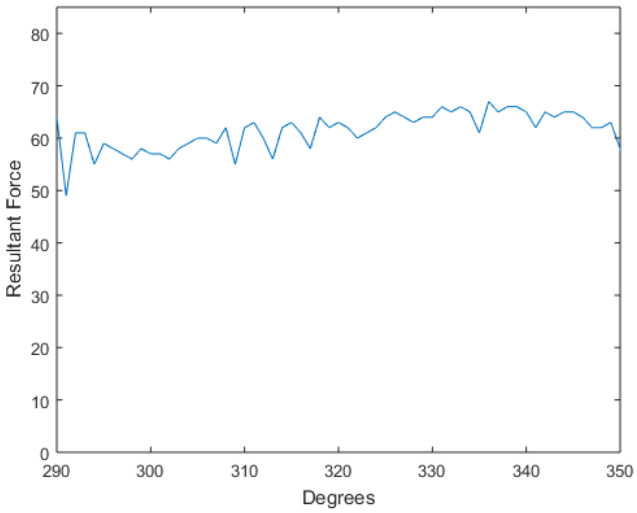


Figure 5.5: Resultant Force with the insertion of self control module

value and stands around it, remaining in this condition in the other episodes. The maximum torque applied in the test was set to 5 N.m because if the value were larger, the arm could not move.

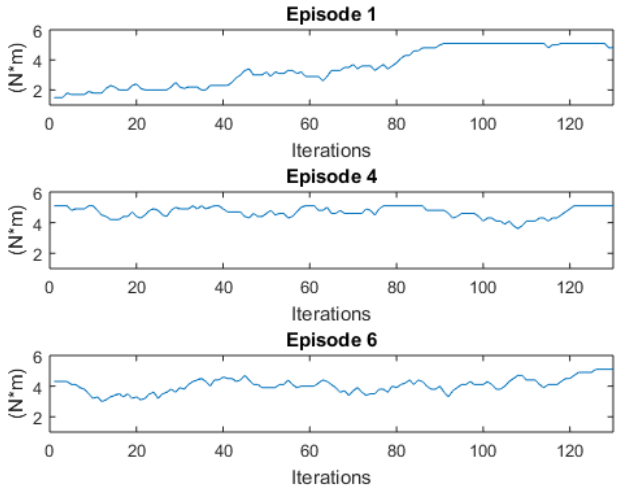


Figure 5.6: Maximum Torque Update of UR3 in some episodes

Lastly, it is relevant to show the learning of the self-control module. That behaviour of the system can be presented through the average reward obtained over the episodes. This information represents the evolution of the system after a number of episodes and can be

seen in Figure 5.7, Notice that in figure the y-axis represents the average reward to a set of 10 episodes and the x-axis exposes this set, i.e., the first value in the x-axis represents 10 episodes and so on.

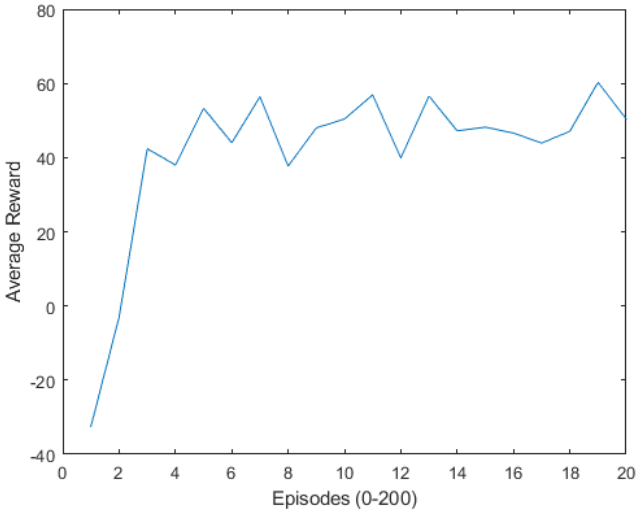


Figure 5.7: Average reward of the RL algorithm

5.2 Experimental Results

The results are measured using the RL algorithm as the self-control model applied in the UR3 to training with a healthy person. The patient has 23 years old, 1,92 meters, and 0.81 meters of arm's length. The set up of the experiment is: the individual stands in front of the robot, grabs the UR3's end-effector as shown by Figure 5.8, and moves the robot along a path stipulated by the experiment therapist.

At first the patient was requested to move the end-effector of the robot only along the x-axis. The UR3 was configured also to move in the same axis. Remember that before the movement begins, the force sensor attached to the robot is re-calibrated to not interfere in the measurements. When the patient starts the motion, the sensor starts the measuring and these data are used to feed the RL technique. To a complete study, two similar algorithms called Q-learning and SARSA were used. These concepts were explained in the Section 3.1. The results obtained with the SARSA algorithm will be showed. However, the Q-learning results

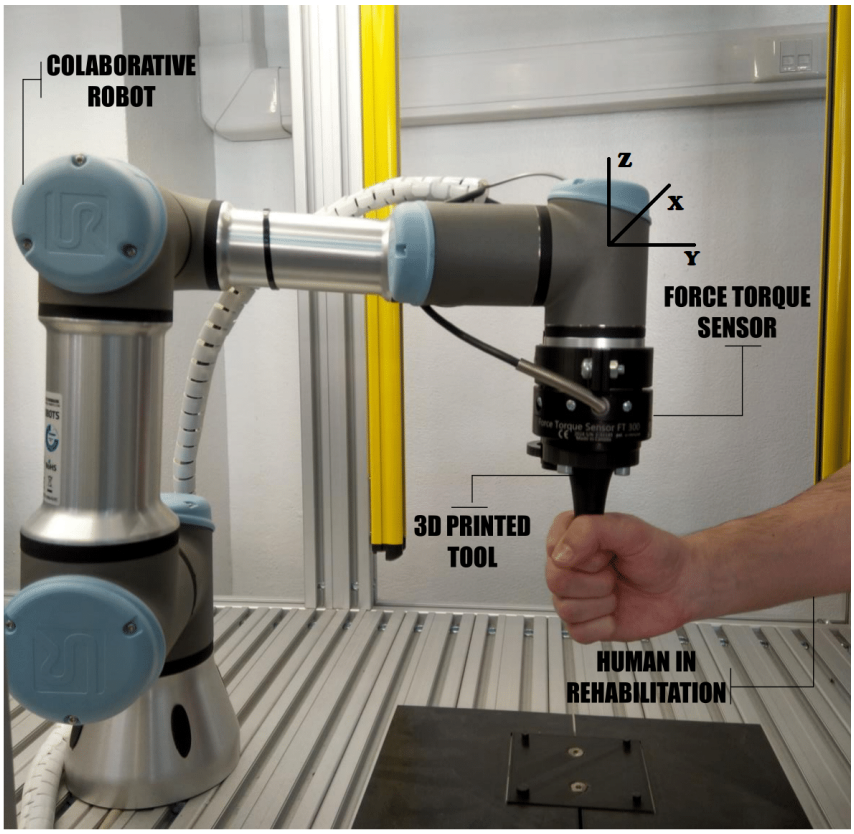


Figure 5.8: Force training with the patient

will be placed on Appendix A. The objective of this first experiment is training the algorithm to provide a resistance force according to the patient's needs, moving along one axis.

After completed the first stage of the experiment, the patient was requested to move the robot in any direction. This makes the decision of the system much more complex, because the robot should deliver the force in three axes and none of them can interfere in the others. Notice that on both stages of the experiment the patient tried to execute a movement with a constant velocity.

The "force_mode" command explained in Section 4.4.2 is used to control the robot with the information passed by the SARSA algorithm, i.e., the SARSA algorithm receive the information about the movement, calculates the resistance and through the "force_mode" command pass this information to the UR3. The command as stated in Figure 4.12 has some parameters that should be configured. The first is the "*task_frame*" which represents in which position will be referenced the application of the resistance by the robot. The "*selection_vector*" controls

Table 5.1: Command force_mode applied on the two experiments

Task Frame	*Selection Vector*	*Wrench*	*Limits*
tool_pose()	[1, 0, 0, 0, 0, 0]	$[ResX, 0, 0, 0, 0, 0]$	[0.15, 0.15, 0.15, 0.1, 0.1, 0.1]
tool_pose()	[1, 1, 1, 0, 0, 0]	$[ResX, ResY, ResZ, 0, 0, 0]$	[0.15, 0.15, 0.15, 0.1, 0.1, 0.1]

which axis will be enabled for movement. The value of resistance force in inserted on the robot through the "*wrench*" parameter. "*type*" refers to how the robot interprets the force frame. The variable "*limits*" represents the limit velocity for each Cartesian or rotation axis enabled. These parameters except "*type*" are shown in Table 5.1.

In Table 5.1 the "tool_pose()" represents the end-effector vector of position in Cartesian and rotation axes. The six position vector (X, Y, Z, RX, RY and RZ) in column *Section Vector)* represents which axes will be enable in the motion. If the value inside the vector is one the axis is active, if the value is zero the axis cannot execute any movement. The variables represented by $ResX$, $ResY$ and $ResZ$ are the forces calculated by the SARSA algorithm and that will be executed by the robot. The "*limits*" column shown the limit values for the velocity in each axis in m/s. The "*Type*" is not metioned in Table 5.1 because have the value 2 for all experiments and represents a simple executing force along the selected axis.

The experimental results presented on this section will be divided in two parts, the one axis and the three axes experiment, respectively.

5.2.1 The one axis experiment

This part refers to the first iteration of the intelligent system with the patient arm. The system records the values of the movement provided by sensors and robots. The first variable shown in Figure 5.9 is the velocity of the UR3's end-effector. It is expected that the velocity stands around value and it seems to happen during all training for the two algorithms. The value of this velocity is 150 mm/s and the variation between the negative and positive value occurs due to the changing of direction of movement. It is also noticed some overtaking in the velocity's curve. This may occur due to measurement errors or the motion changing.

It is expected that the velocity stands around a value and it seems to happens during all training for the two algorithms. The value of this velocity is 150 mm/s and the variation

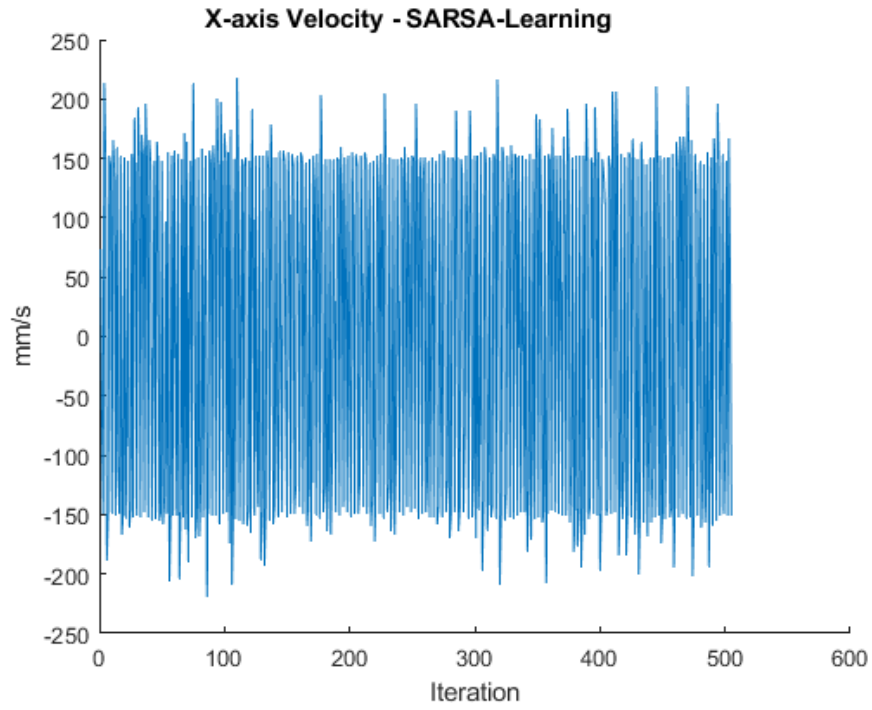


Figure 5.9: Recorded velocity on the movement - SARSA algorithm

between the negative and positive value occur due to the changing of direction of movement. It is also noticed some overtaking in the velocity's curve, this may occur due measurement errors related to the movement's direction changes.

Another variable recorded during the system training is the position of the UR3's end-effector. As the behaviour is independently of the trajectory, this curve is not always be well defined. However, the position is also important when it is performed the comparison with the human arm model, then the Figure 5.10 presents this variable for the SARSA algorithm.

Even disregarding the trajectory some pattern in the movement may be noticed. This happens because the limitation of the robotic arm. This curve can be also compared with the values of resistance force provided by the UR3. To analyse how this value changes and when this happens, the Figure 5.11 shows the comparison between the curves. Notice that it is presented a set of episodes to facilitate the reading of the data.

The position curve almost every time represents the force direction of the human arm. This means that the resistance applied by the UR3 shall be the same behaviour, but inverted.

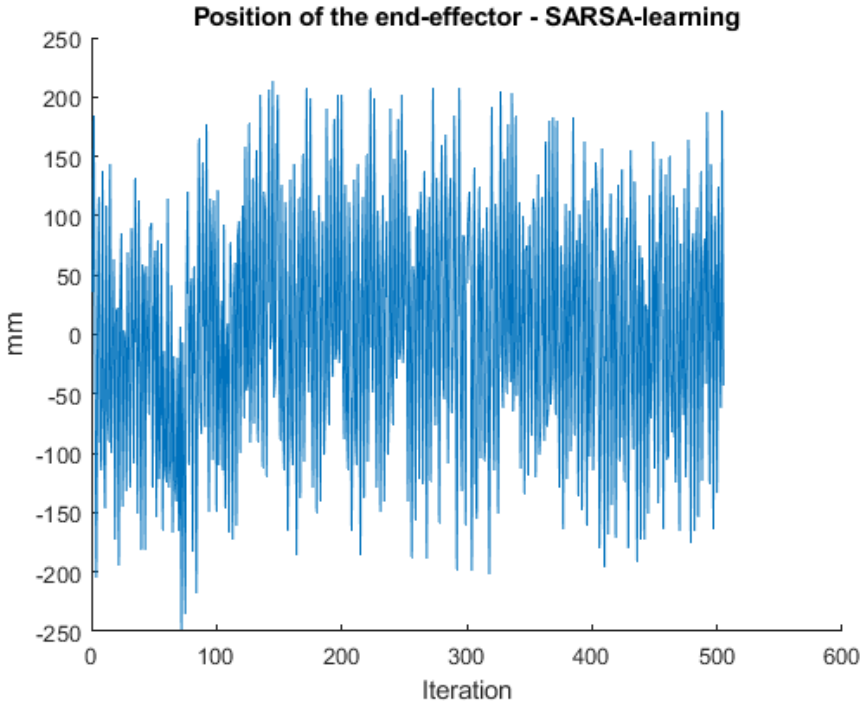


Figure 5.10: Recorded position on the movement - SARSA algorithm

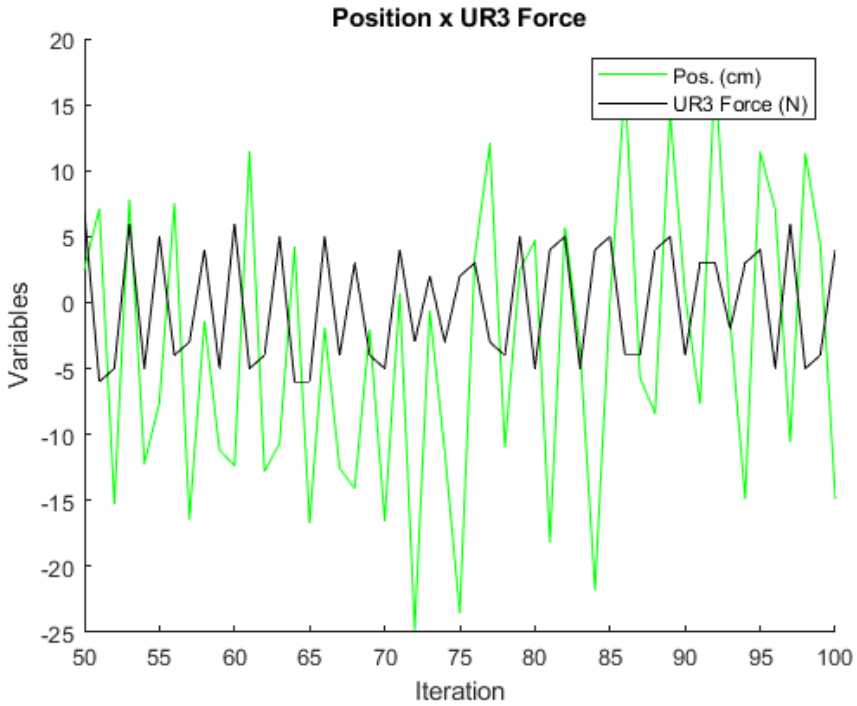


Figure 5.11: Comparison between position and resistance force - SARSA algorithm

The UR3's force showed in Figure 5.11 is calculated by the self-control using as an input the measured resultant force by the sensor. It is also considered as a goal of the system, that the same variable is around the value of 25 N, inside a proposed range (20 N - 40 N). The system cannot guarantee that the forces are always inside the proposed range, however, most times has to seek this target. The Figure 5.12 shows this data to the entire training. Notice that the data represents the force measured in the three axes (XYZ). Even that motion is only in one axis, there are reading forces in all axes and these variables should be considered.

Figure 5.12: The measured resultant force by the sensor - SARSA algorithm

Figure 5.12 presents many values that exceed the 40 N. Some of these values can be considering that the patient can execute this force for a short period (time). However, this value not always represents reality. Even with the technique used to mitigate the reading errors, some of them were detected, principally when the patient changes the direction of the movement. Therefore, the analysis must be done considering the average value of the force, which seems to be around 28 N, and this is acceptable behavior. This overtakes are also a result of the force applied in the other axes recorded by the sensor. Thus, if the forces on the

other directions were disregarded, the force should perform better.

There are forces applied in the other axes that are recorded by the sensor and this explain some of the overtakes noticed in the curve. Thus, if the force recorded on the other directions was diregard, the curve will be presented a better behaviour as shown in Figure 5.13.

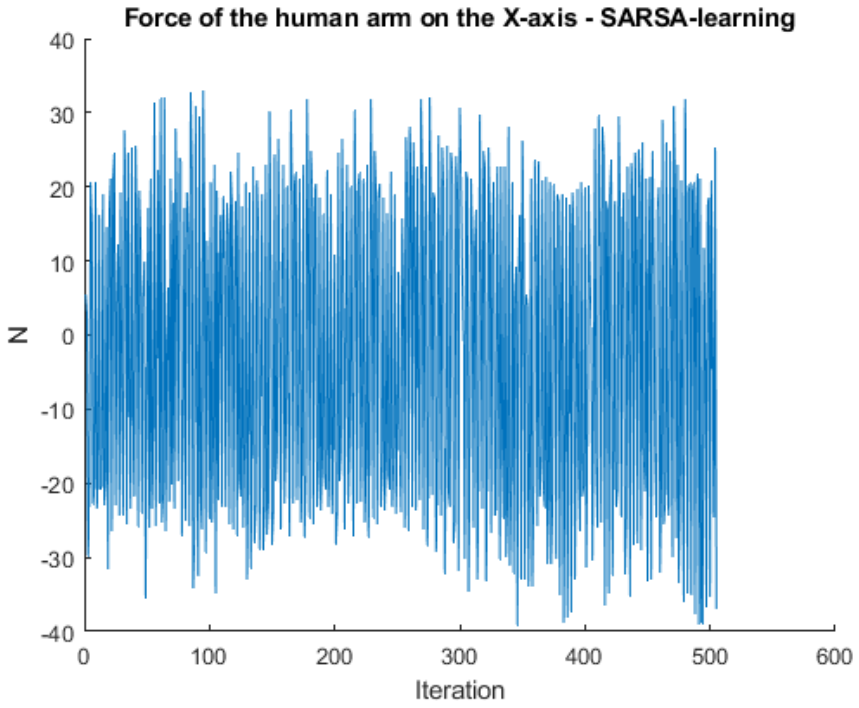


Figure 5.13: The measured force by the sensor on the x-axis - SARSA algorithm

Figure 5.13 presents the force for the measurements about the x-axis. The value of the force depends on the direction of the movement. So, the reading of this variable should only be considered the value module. However, this curve also shows to the reader that the systems choose the next resistance force independently of the direction, promoting an almost constant variation in the x-axis force.

Inside the set of proposed actions there is no correct action that the system can make, however, there are correct actions in some situations. It is expected that the self-control chooses these correct actions to get the rewards and at the end of training the negative be smaller than the positive. This objective is reached and the Figure 5.14 shows a comparison between the number of positive (+1) and negative (-1) rewards assigned. The number of

positive rewards, where the system recorded a human arm force inside the proposed range, is at least two times larger than the negative rewards.

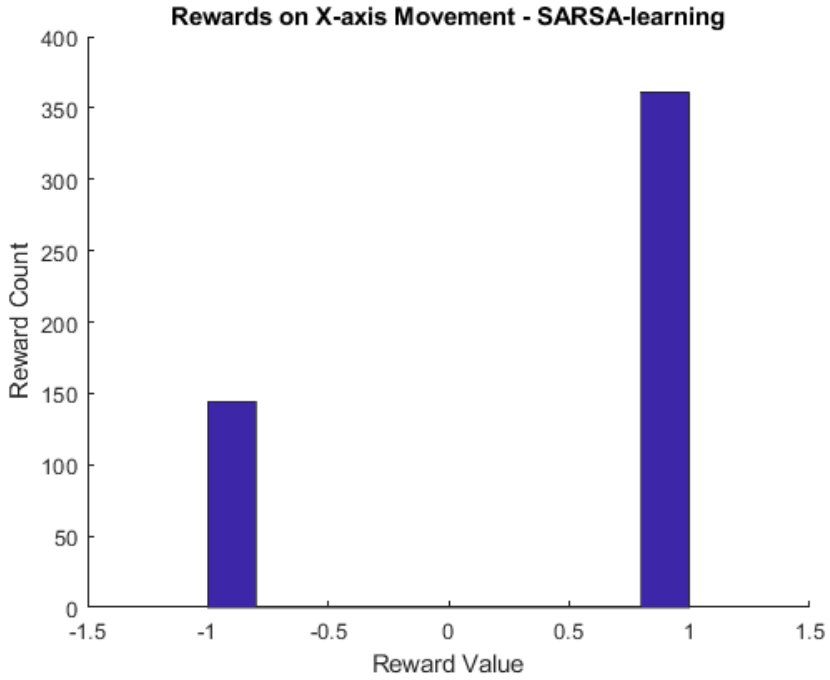


Figure 5.14: Rewards assigned by the SARSA algorithm

The goal pursued was successfully achieved, precisely because the force of the human arm remained around the force established during almost all training. However, it is not clear what is the better resistance force that the robot should deliver to the patient in each episode. The data recorded on the selected force shows exactly this situation, as stressed in Figure 5.15. Nevertheless is possible to analyze that the robot didn't learn the better force, but the learning was the best moments to increase or decrease the force.

It was noticed that empirically an UR3 force of 6 N is capable to ensure that the arm will execute a force close to expected, but this value depends on the position of the patient relative to the robot. Thus, sometimes a smaller or higher resistance force is required to accomplish the goals. As the system is dynamic and complex, 500 interactions were not enough to make the system to fit optimally, but it is possible to verify that this same number of episodes was enough for the robot to have the expected behavior.

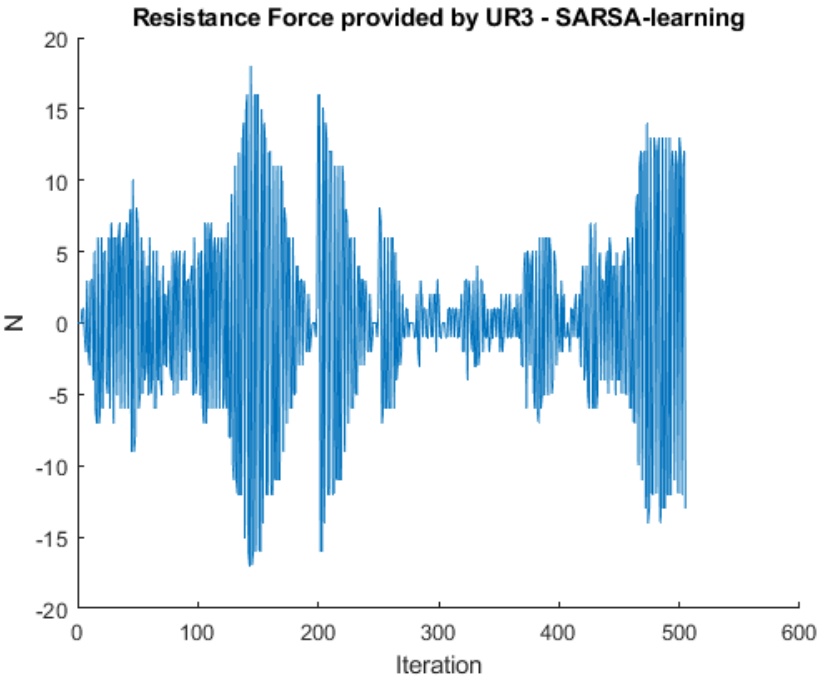


Figure 5.15: The UR3's applied force - SARSA algorithm

5.2.2 The three axes experiment

In this experiment stage, the patient was free to move the UR3's end-effector to any possible place, respecting the limitation of the robot. Before the training starts, the sensor was again re-calibrated due to the same reasons for the first part. The tests were performed using only the SARSA algorithm because showed a fast convergence compared to Q-learning in the previous experiment.

As the algorithm works with a more complex system and should provide the resistance force in the three axes, some modifications had to be made. There are two ways to implement the RL algorithm in this experiment stage. The first is considering three different choices, one resistance force for each axis, i.e., each axis has their own Q-matrix and these matrices are updated separately, this method was also represented in Section 4.2.2. The other way is considering the increment of the resistance force separately, but the reward is awarded to the entire system, i.e., only one Q-matrix is responsible to capture the information that represents the entire system, but the choices are made separately considering the values that represent the force in the determined axis. This second method to accomplish this experiment was

developed after the failure of the first.

Using this first method the goal was leading the patient to execute a force between 20 and 40 N, but the system behaviour was different. As the system learns separately which resistance force should be performed considering each axis, the reward will be awarded taking into account only the force executed in that axis. However, if the patient move the robot along only one or two axis, the sensor will record a force near by zero in that other axis. So, when the force is smaller than the expected, the RL algorithm will put a greater resistance force in that direction. Therefore, after some time, a provided resistance could be so larger that the patient will not be capable to execute any motion, or the robotic arm would push the arm in the opposite direction. The Figures 5.16 and 5.17 present the force in one axis (X-axis) and the resultant force measured by the sensor, respectively.

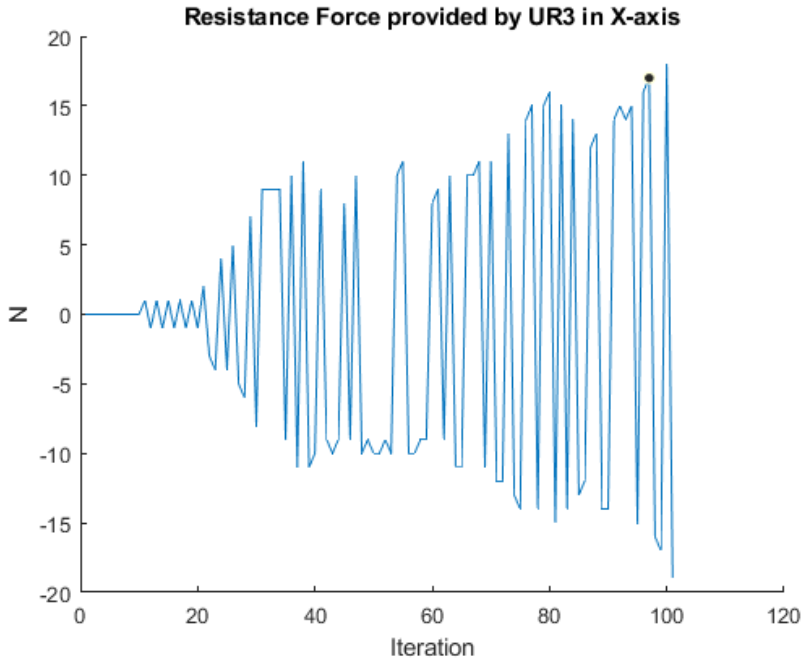


Figure 5.16: Measured force in X-axis

Notice that in the Figure 5.16 the fore starts in zero and growing indefinitely, the same happens for the other axis, this means that the resultant in the ninetieth iteration is larger that the force applied by the human arm, and at that moment started to be driven by the robot. When this happens, the force immediately decrease, and this moment can be seen in

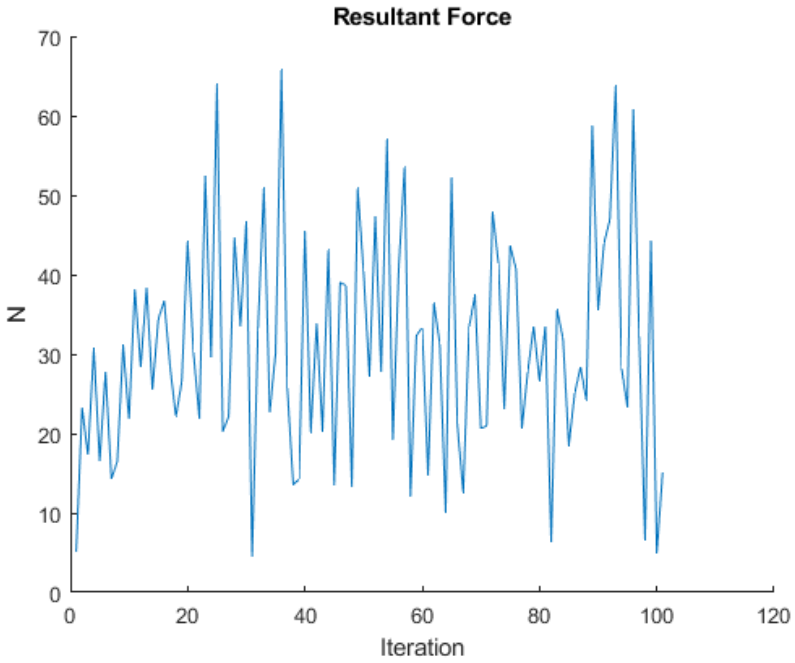


Figure 5.17: Resultant force measured by the sensor

the Figure 5.17.

After consider the results of this training a crucial error was detected. The rewards was awarded considering the expected resistance force for the three axes, that is, the x-axis would try to keep the arm performing a force of 40 N, in the worst case, as well as the Y and Z axes. Thus, the algorithm only will award the system when a force between 34.64 N and 69.28 N was measured, considering the axes forces equals.

The other method was proposed to correct the errors of the first, instead award each axis, the algorithm assign the reward using the resultant force. Thus, the system will know which human's force is acceptable and will provide the resistance force based on this. The choice to increase or decrease the resistance is made using the updated values of the Q-matrix. This technique shows a better behaviour, because the force in each axis will not grow up fast and the resistance felt by the patient was described as sufficient.

The resultant force measured by the sensor shows a better behaviour compared with the previous method, this data is presented in Figure 5.18. This force is too large in the first iteration, but in the sequence it seems to normalise near by an acceptable value.

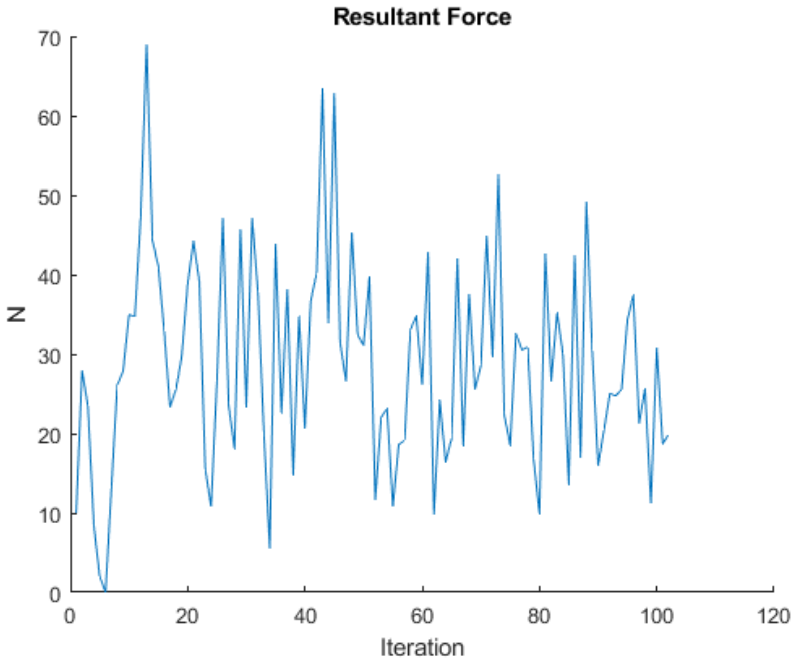


Figure 5.18: Resultant force measured by the sensor

The same happens for the robot force. As the Figure 5.19 shows, the force seems to remain near by a value of 4N. The same happens for the other two axes and is presented in the Appendix B. Is also important remember that the learning is linked to the actions of increasing, deceasing or maintaining the force.

Even using this technique, the learning was not perfectly suitable to this problem. The algorithm grants the rewards based on the resultant force and this ensures that the system always will search for a value of force inside of the range specified in the algorithm. However, as only one Q-matrix controls the three axes the value of an axis interfere in another, thus the system will not learn the better force for any movement in each axis, but will learn some range of values that can be used in the three axis to seek the desired resultant force.

Perhaps the solution for this problem is to create an algorithm that is capable to perform a decision based on the robot force, the measured resultant force, the action and the three axis. This implies in a Q-matrix with six dimensions. Another solution is to implement three tri-dimensional Q-matrix, each one to evaluate each axis, and another that evaluate the entire movement.

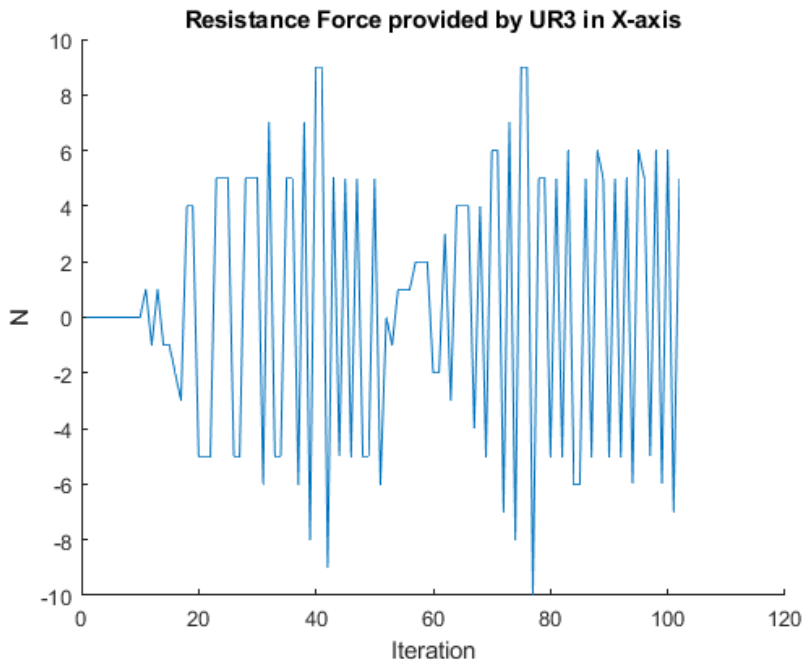


Figure 5.19: robot force in X-axis

5.3 Model of Human Arm

This section is a part of the results that shows a simulation of the human model presented in Section 3.2 and has the goal to provide some force values to compare with the experimental results presented in the Section 5.2.

Using Equation 3.8 as a reference, is possible to discover which human arm's force is necessary to move it along the path established. To that Equation 5.1 is added the opposite force provided by the robot, thus the equation can be rewritten to find the force performed by the arm.

$$F_{arm} = k(x_d - x) + F_{rob} \tag{5.1}$$

The model will be used based on the experiment with one axis because is simpler and the obtained results using the SARSA algorithm for this experiment were better. The k will be a single value that only represents the stiffness in x-axis and will be 200 N/m. The experiments performed by Ajoudani [42] found similar k values. The k value is dynamic and changes

according to the position of the arm, the muscle strength, the direction of the movement, and the force execute by the limb in the situation. It is valid to remember that the factors M and D were neglected because the Figure 5.9 shows a movement velocity almost constant.

Then, if the Equation 5.1 is applied with the data obtained by the system, the result are the F_{arm} value for that situation, and this is presented in Figure 5.20. When this Figure is compared with the Figure 5.13 many similarities appear, proving the concept used. The possible errors could be assigned to the dynamic of the movement.

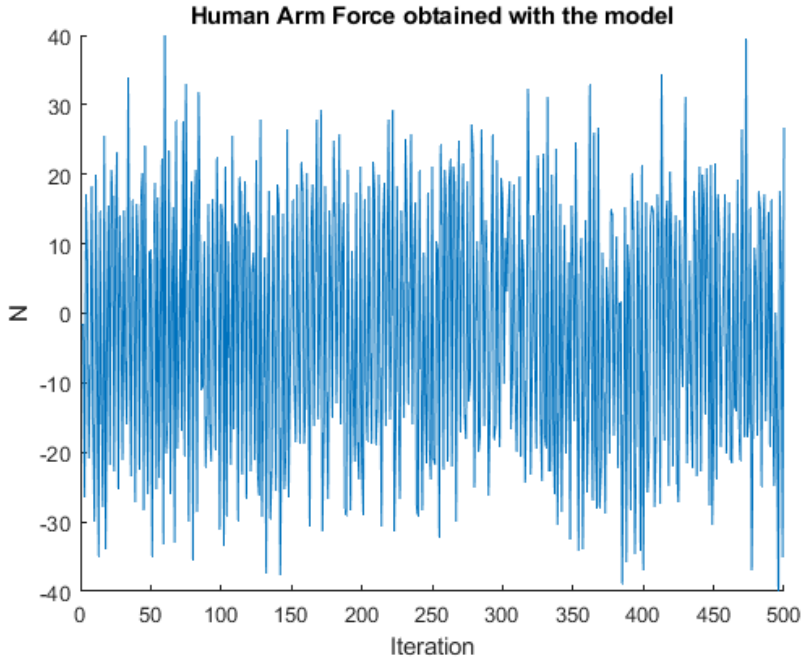


Figure 5.20: Human Arm force based on Equation 5.1

The use of the Equation 5.1 may also aim to find the k value for this patient. In this case, the used human arm force is that measured by the sensor. For the experiment using the SARSA algorithm in the x-axis the average k value is 190.63 N/m.

The use of the human arm model applied to this experiment can be beneficial for the further experimentation, because is possible to define this value for the patient during the training and uses it to feed the RL algorithm to find a better solution for this problem.

It is also important to comment that no experiments were found where this k value is calculated for people with upper limb dysfunction. Thus, it is not possible to compare the

experiment performed here with another authors.

Chapter 6

Conclusions

In the course of this thesis a system based on the UR3 robot from Universal Robots© to help the upper limb physiotherapy of patients with some dysfunction providing a force training for the limb was developed. The focuses of this work were the self-control algorithms used, based on the RL algorithms: SARSA and Q-learning. These algorithms have the purpose to present an alternative way to control this type of system when the expected behaviour is dynamic and specified to each patient. So, the aim is making this system more robust and simple in the therapist and patient perspective, and also decrease the time of system configuration observed in other solutions.

At first, to help the development and test of these algorithms was proposed a simulation in the V-REP platform, the environment had the UR3 and the human arm model to simulate the contact between these elements. The algorithms, in this part, were implemented in the MATLAB platform and were prepared to deal with the force problem. Synchronous communication between these platforms was configured giving to the algorithm control over the simulation data. The results of this part of the experiment were satisfactory because the system has an expected behaviour in terms of resistance force provided to the human movement.

Sequentially, the tested algorithms were implemented in the real robot to test the system with a human subject. The algorithms were converted to Python because this language presented some advantages over the MATLAB, principally in terms of communication speed with the UR3. The tests with the human subject were prepared in two acts, the first was enabling only one axis and implementing the two algorithms studied, and the second was enabling the three axes and using only the SARSA algorithm due its fast convergence.

The results obtained with subject also followed the expected behaviour, although some

errors due to dynamics movements and measurements. Therefore, it can be stated that the robot was able to perform a resistance according to the patient force and direction. However, the experiment with the three axes do not presented satisfactory results. Although the system is capable to perform the resistance according to the patient force and direction, the learning was not the better value of force for each situation, but which action is correct (increase, decrease or hold) on the axes that return the sought human arm value.

The development of this work has already provided the production of an article of scientific interest. In conclusion, the work proposed a contribution for the rehabilitation field, using a robotic system and a self-control algorithm to provide a simpler and robust tool to the therapists and patients. This proposed solution can also be applied in several fields of industrial robotics due to its flexibility.

6.1 Future Works

Based on the presented work, it is possible to identify several points that can support future studies. The studies to be carried out in the future can be searched through the following topics:

- This work uses revolute joints to emulate the shoulder in the simulation part, despite the satisfactory results, it will be an asset the use of a human arm model to emulate the joints and get better results that describe the movement of the human arm;

- The position was disregarded in this work due the initial objective to create a system without the need of a path planning, but this data is critical to analyse the results and how the variation of this data interfere in the value of resistance that should be provided by the robot to the patient. Therefore, it is proposed a mapping of the possible positions that the robot can reach and use this data to assign the rewards to the system. Another way to deal with the position problem is to implement some neural network to predict the next position and use this value to feed the RL algorithm. If the system is capable to

predict the next position, the three axes problem becomes simpler because the system can predict how forces will be distributed along the axes based on the next position;

- Is also proposed, to make a robust solution, to calculate the k values in each iteration and use to feed the RL algorithm. This value can be an optimal parameter to determine the condition of the human arm and its improvement during the physiotherapy. It is interesting, for the research purposes, to establish a database about the k values measured from the contact with patients.

- Lastly, it is also recommended, the use of this system on patients with upper limb dysfunction to determine the efficiency of this solution;

References

[1] F. Gimigliano and S. Negrini, "The world health organization "rehabilitation 2030–a call for action"", *European Journal of Physical and Rehabilitation Medicine*, vol. 53, no. 2, pp. 155–168, 2017, ISSN: 19739095. DOI: 10.23736/S1973-9087.17.04746-3.

[2] W. R. 2030, *A call for action plan: The need to scale up rehabilitation*, 2017.

[3] W. H. Organization *et al.*, *International classification of functioning, disability and health: ICF*. Geneva: World Health Organization, 2001.

[4] W. H. Organisation, *World report on disability*, 2011.

[5] W. H. Organization *et al.*, *WHO global disability action plan 2014-2021: Better health for all people with disability*. World Health Organization, 2015.

[6] P. L. Ruiz, "Uso da robótica na reabilitação: Aplicação para a fisioterapia", *UNILUS Ensino e Pesquisa*, vol. 14, no. 37, pp. 188–191, 2018.

[7] J. Mackay and G. A. Mensah, *The Atlas of Heart Disease and Stroke*, 1st. Geneva, Switzerland: World Health Organisation, 2004, ISBN: 9241562765.

[8] C. A. d. P. Piassaroli, G. d. Almeida, J. Luvizotto, and A. Suzan, "Modelos de reabilitação fisioterápica em pacientes adultos com sequelas de avc isquêmico", *Rev Neurocienc*, vol. 20, no. 1, pp. 128–37, 2012.

[9] A. McConnell, X. Kong, and P. A. Vargas, "A novel robotic assistive device for stroke-rehabilitation", in *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, IEEE, 2014, pp. 917–923.

[10] D. Koo, P. H. Chang, M. K. Sohn, and J.-h. Shin, "Shoulder mechanism design of an exoskeleton robot for stroke patient rehabilitation", in *2011 IEEE International Conference on Rehabilitation Robotics*, IEEE, 2011, pp. 1–6.

[11] N. Tejima, "Rehabilitation robotics: A review", *Advanced Robotics*, vol. 14, no. 7, pp. 551–564, 2001.

[12] F. Badesa, R Morales, J. Sabater-Navarro, N Garcia-Aracil, J. Azorin, and C Perez, "Experiencias en el desarrollo de un sistema robótico para rehabilitación de miembro superior para pacientes con daño cerebral sobrevenido", *Revista Universitaria en Telecomunicaciones, Informática y Control*, vol. 1, no. 1, 2012.

[13] F. C. M. Castaño, F. Y. P. Peñaranda, F. M. Y. P. Bernal, and F. J. C. Ruíz, "Aplicación de la terapia robó tica para el tratamiento de la mano espástica del adulto con hemiplejía. artículo de revisión", *Rev Mex Med Fis Rehab*, vol. 27, no. 3-4, pp. 80–85, 2015.

[14] A. Tóth, G. Arz, G. Fazekas, D. Bratanov, and N. Zlatov, "25 post stroke shoulder-elbow physiotherapy with industrial robots", in *Advances in Rehabilitation Robotics*, Springer, 2004, pp. 391–411.

[15] E. Taub, N. E. Miller, T. A. Novack, E. W. Cook, W. Fleming, C. Nepomuceno, J. Connell, J. Crago, *et al.*, "Technique to improve chronic motor deficit after stroke", *Archives of physical medicine and rehabilitation*, vol. 74, no. 4, pp. 347–354, 1993.

[16] A. Toth, G. Fazekas, G. Arz, M. Jurak, and M. Horvath, "Passive robotic movement therapy of the spastic hemiparetic arm with reharob: Report of the first clinical test and the follow-up system improvement", in *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.*, IEEE, 2005, pp. 127–130.

[17] of Technology and B. U. Economics, *Project - research: Exercises*, 2000 (accessed August 20, 2019). [Online]. Available: http://reharob.manuf.bme.hu/research/exercises/.

[18] Hocoma, *Arm hand*, 2019 (accessed August 20, 2019). [Online]. Available: https://www.hocoma.com/solutions/arm-hand/.

[19] D. Gijbels, I. Lamers, L. Kerkhofs, G. Alders, E. Knippenberg, and P. Feys, "The armeo spring as training tool to improve upper limb functionality in multiple sclerosis: A pilot study", *Journal of neuroengineering and rehabilitation*, vol. 8, no. 1, p. 5, 2011.

[20] C Colomer, A Baldovi, S Torromé, M. Navarro, B Moliner, J Ferri, and E Noé, "Efficacy of armeo® spring during the chronic phase of stroke. study in mild to moderate cases of hemiparesis", *Neurologia (English Edition)*, vol. 28, no. 5, pp. 261–267, 2013.

[21] S. M. El-Shamy, "Efficacy of armeo® robotic therapy versus conventional therapy on upper limb function in children with hemiplegic cerebral palsy", *American journal of physical medicine & rehabilitation*, vol. 97, no. 3, pp. 164–169, 2018.

[22] E. Oña, R Cano-de la Cuerda, P Sánchez-Herrera, C Balaguer, and A Jardón, "A review of robotics in neurorehabilitation: Towards an automated process for upper limb", *Journal of healthcare engineering*, vol. 2018, 2018.

[23] J. E. Colgate, J Edward, M. A. Peshkin, and W. Wannasuphoprasit, "Cobots: Robots for collaboration with human operators", 1996.

[24] U. Robots, *Universal robot ur3*, 2019 (accessed August 21, 2019). [Online]. Available: https://www.universal-robots.com/br/produtos/ur3/.

[25] KUKA, *Kuka cobots*, 2019 (accessed August 21, 2019). [Online]. Available: https://www.kuka.com/pt-pt/future-production/industrie-4-0/industrie-4-0-cobots-in-industry.

[26] ABB, *Abb's collaborative robot -yumi - industrial robots from abb robotics*, 2019 (accessed August 21, 2019). [Online]. Available: https://new.abb.com/products/robotics/industrial-robots/irb-14000-yumi.

[27] E. Papaleo, L. Zollo, L. Spedaliere, and E. Guglielmelli, "Patient-tailored adaptive robotic system for upper-limb rehabilitation", in *2013 IEEE International Conference on Robotics and Automation*, IEEE, 2013, pp. 3860–3865.

[28] J. Nielsen, *Rehabilitation using industrial robots - universal robotrainer*, 2012 (accessed August 21, 2019). [Online]. Available: http://www.en.patientathome.dk/projects/rehabilitation-using-industrial-robots-universal-robotrainer.aspx.

[29] U. o. S. Denmark, *Universal robo trainer*, 2019 (accessed August 23, 2019). [Online]. Available: https://www.sdu.dk/en/om_sdu/institutter_centre/healthinformaticsandtechnology/significantachievements/robotrainer.

[30] B. C. Weigelin, M. Mathiesen, C. Nielsen, K. Fischer, and J. Nielsen, "Trust in medical human-robot interactions based on kinesthetic guidance", in *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, IEEE, 2018, pp. 901–908.

[31] Z. Pineda-Rico, S. de Lucio, J. Alfonso, F. J. Martinez Lopez, and P. Cruz, "Design of an exoskeleton for upper limb robot-assisted rehabilitation based on co-simulation", *Journal of Vibroengineering*, vol. 18, no. 5, pp. 3269–3278, 2016.

[32] C. Cortés, A. Ardanza, F Molina-Rueda, A Cuesta-Gómez, L. Unzueta, G. Epelde, O. E. Ruiz, A. De Mauro, and J. Florez, "Upper limb posture estimation in robotic and virtual reality-based rehabilitation", *BioMed research international*, vol. 2014, 2014.

[33] Z. Du, Y. Sun, Y. Su, and W. Dong, "A ros/gazebo based method in developing virtual training scene for upper limb rehabilitation", in *2014 IEEE International Conference on Progress in Informatics and Computing*, IEEE, 2014, pp. 307–311.

[34] H. S. Lo and S. Q. Xie, "Exoskeleton robots for upper-limb rehabilitation: State of the art and future prospects", *Medical engineering & physics*, vol. 34, no. 3, pp. 261–268, 2012.

[35] H. I. Krebs, J. J. Palazzolo, L. Dipietro, M. Ferraro, J. Krol, K. Rannekleiv, B. T. Volpe, and N. Hogan, "Rehabilitation robotics: Performance-based progressive robot-assisted therapy", *Autonomous robots*, vol. 15, no. 1, pp. 7–20, 2003.

[36] L. Kahn, W. Z. Rymer, and D. Reinkensmeyer, "Adaptive assistance for guided force training in chronic stroke", in *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, IEEE, vol. 1, 2004, pp. 2722–2725.

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

[38] M. Wiering and M. Van Otterlo, "Reinforcement learning", *Adaptation, learning, and optimization*, vol. 12, p. 3, 2012.

[39] C. Szepesvári, "Algorithms for reinforcement learning", *Synthesis lectures on artificial intelligence and machine learning*, vol. 4, no. 1, pp. 1–103, 2010.

[40] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control", *IEEE circuits and systems magazine*, vol. 9, no. 3, pp. 32–50, 2009.

[41] M. Ashraf, *Reinforcement learning demystified: Markov decision processes (part 1)*, 2018 (accessed August 27, 2019). [Online]. Available: https://towardsdatascience.com/reinforcement-learning-demystified-markov-decision-processes-part-1-bf00dda41690.

[42] A. Ajoudani, *Transferring human impedance regulation skills to robots*. Springer, 2016.

[43] M. J. Fu and M. C. Cavusoglu, "Human-arm-and-hand-dynamic model with variability analyses for a stylus-based haptic interface", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 6, pp. 1633–1644, 2012.

[44] Y. Yang, L. Wang, J. Tong, and L. Zhang, "Arm rehabilitation robot impedance control and experimentation", in *2006 IEEE International Conference on Robotics and Biomimetics*, IEEE, 2006, pp. 914–918.

[45] U. Robots, *Ur3 technical specification*, English, version Item no. 110103, Universal Robots, 1 p.

[46] ——, *Max. joint torques - 17260*, 2018 (accessed September 06, 2019). [Online]. Available: https://www.universal-robots.com/how-tos-and-faqs/faq/ur-faq/max-joint-torques-17260/.

[47] ——, *Ur3e technical details*, 2019 (accessed December 20, 2019). [Online]. Available: https://www.universal-robots.com/media/1802780/ur3e-32528_ur_technical_details_.pdf.

[48] ROBOTIQ, *Ft 300 force torque sensor*, English, ROBOTIQ, 2 pp.

[49] C. Robotics, *V-REP*, http://www.coppeliarobotics.com, Acessed: 2019-05-08, 2019.

[50] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework", in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 1321–1326.

[51] D. C. Ribeiro, M. G. Estivalet, and J. F. Loss, "Modelo para estimativa da força e torque muscular durante a abdução do ombro", *revista portuguesa de ciências do desporto*, vol. 8, no. 3, pp. 321–329, 2008.

[52] *Remote api modus operandi*, 2018 (accessed September 05, 2019). [Online]. Available: http://www.coppeliarobotics.com/helpFiles/en/remoteApiModusOperandi.htm#synchronous.

[53] *Remote api modus operandi*, 2018 (accessed September 05, 2019). [Online]. Available: http://www.coppeliarobotics.com/helpFiles/en/remoteApiModusOperandi.htm#synchronous.

[54] U. Robots, *The urscript programming language*, English, version Version 3.2, Universal Robots, 59 pp.

[55] *Reaction times to sound, light and touch*, 2018 (accessed September 05, 2019). [Online]. Available: https://bionumbers.hms.harvard.edu/bionumber.aspx?id=110800.

Appendix A

The Q-learning results for the one axis experiment

This Appendix has the goal to show the results collected with the experiment performed in the x-axis using the Q-learning algorithm. Another information and analyses about these data is presented in Section 5.2.1.

The Figure A.1 shows the velocity of the UR3 end-effector. Similar to the SARSA this value seems to be constant. The positive and negative variation represents the direction of the movement.

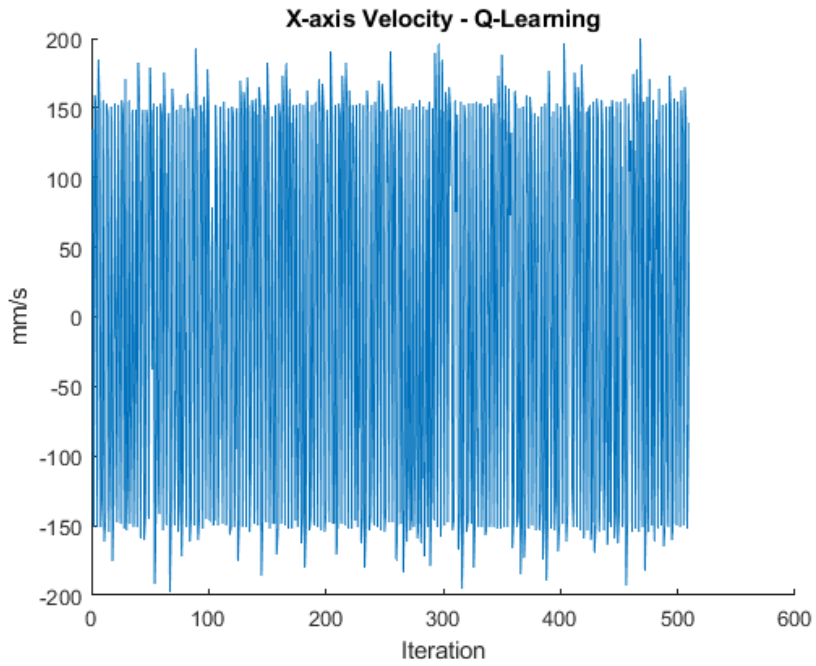


Figure A.1: Recorded velocity on the movement - Q-learning algorithm

The position of the end-effector also seems to be the same behaviour when is using the Q-learning or SARSA. This variable for the Q-learning is shown in Figure 5.10.

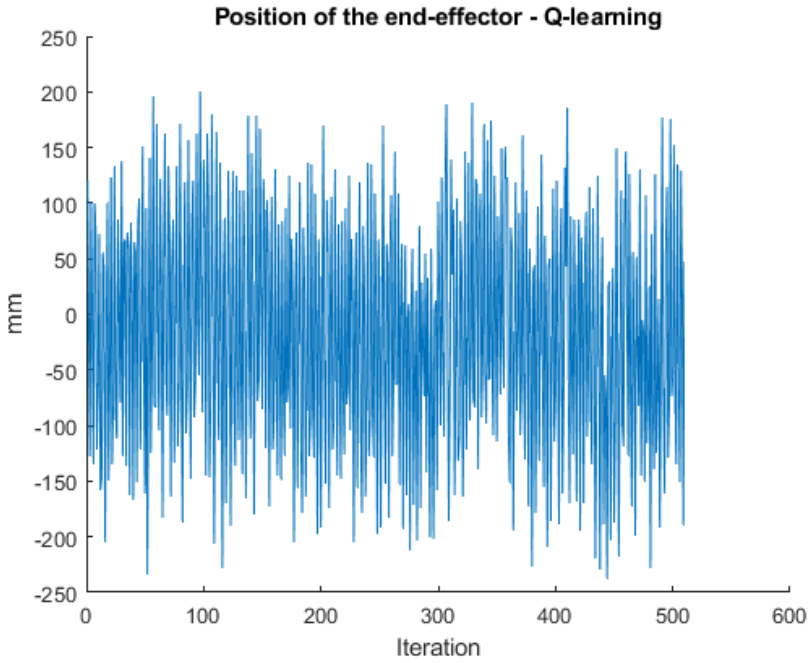


Figure A.2: Recorded position on the movement - Q-learning algorithm

It is important to present the comparison between the position and the force curves, to verify in which moment the force is applied in each iteration. The Figure A.3 presents these curves.

The values measured by the force sensor using the Q-learning algorithm are presented in Figure A.4 that represents the resultant force and Figure A.5 that shows the force only in the x-axis. Remember that there are force in Y and Z axes and this is the motivation to present the resultant force.

The rewards obtained by the Q-learning are presented in Figure A.6 and shows a great difference between the positive and negative rewards. The positive rewards in this case were four times greater than the negative.

The robot force applied in the movement is shown in Figure A.7 and this value also was measured using the Q-learning.

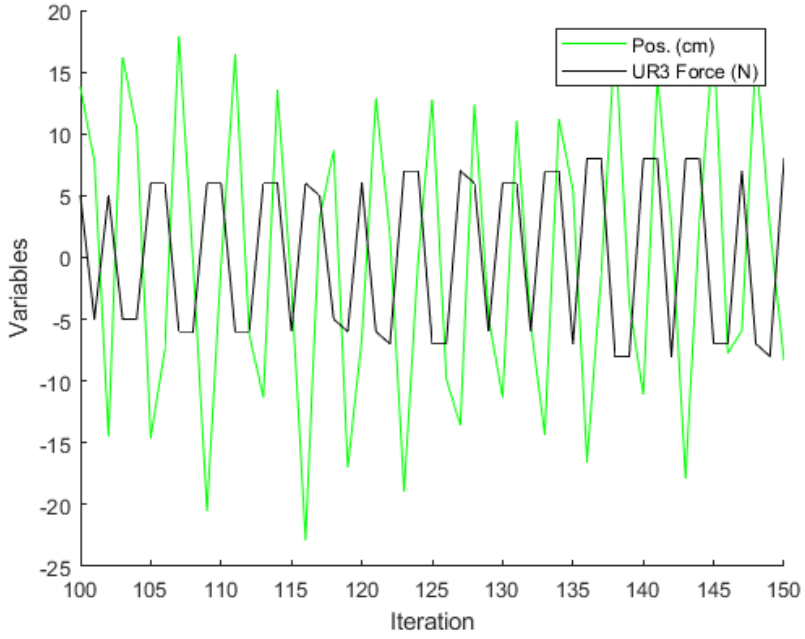


Figure A.3: Comparison between position and resistance force - Q-learning algorithm

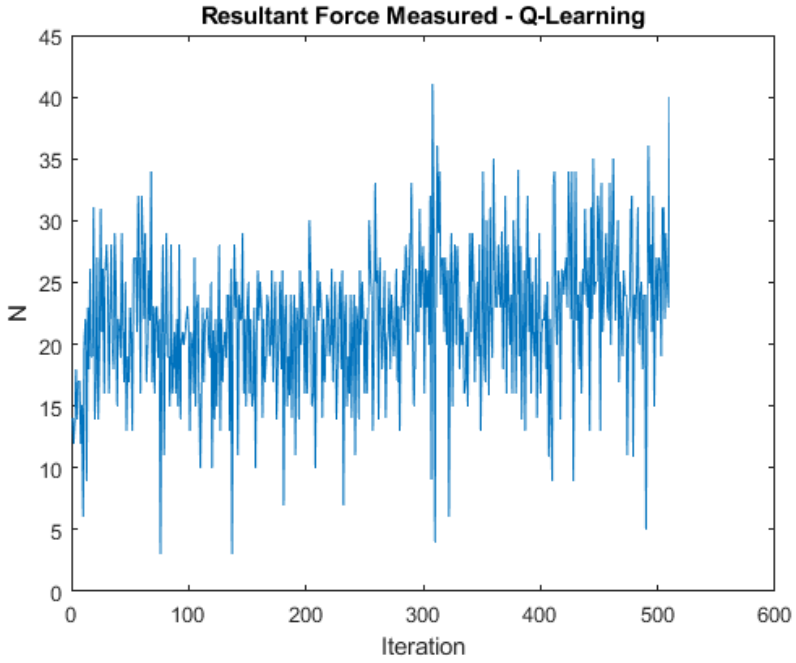


Figure A.4: The measured resultant force by the sensor - Q-learning algorithm

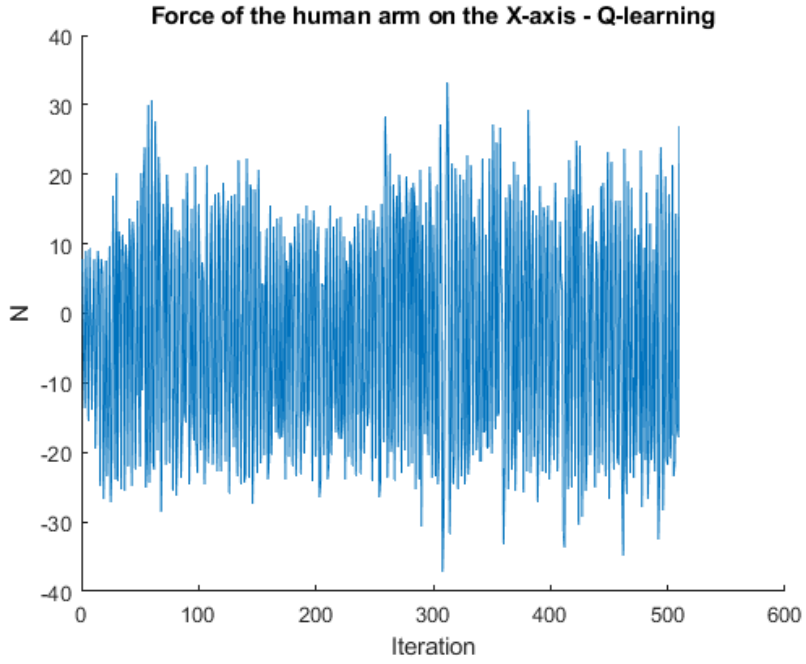


Figure A.5: The measured force by the sensor on the x-axis - Q-learning algorithm

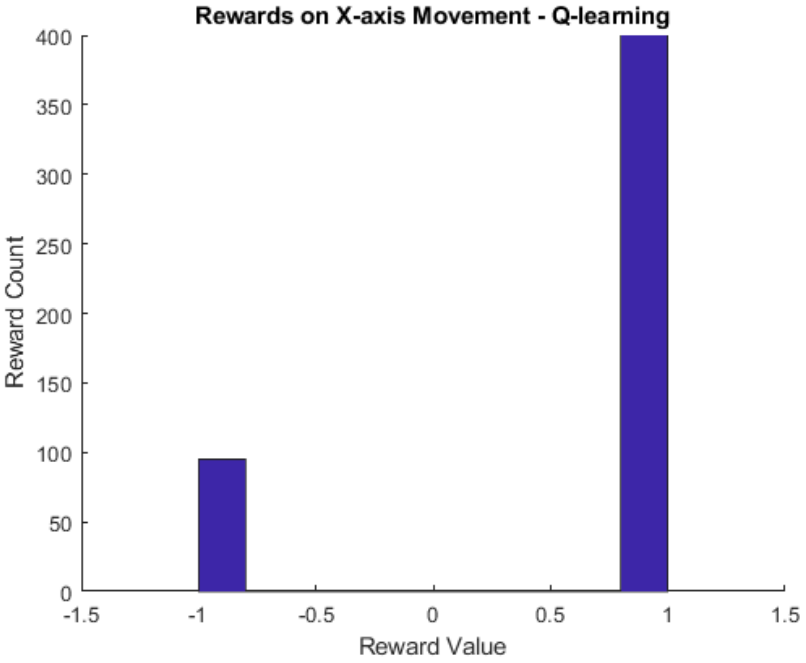


Figure A.6: Rewards assigned by the Q-learning algorithm

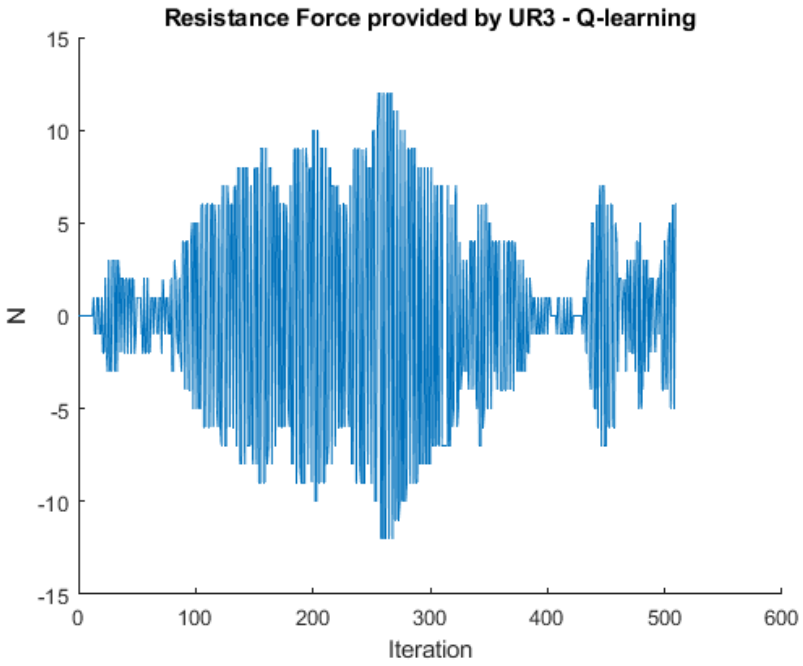


Figure A.7: The UR3's applied force - Q-learning algorithm

Appendix B

Other Results from the three axes experiment

This Appendix presents the forces applied by the robot in the Y and Z axes. The values are measured during the training of the three axes experiment. The Figure B.1 shows the UR3 force for the Y-axis, and the Figure B.2 for the z-axis.

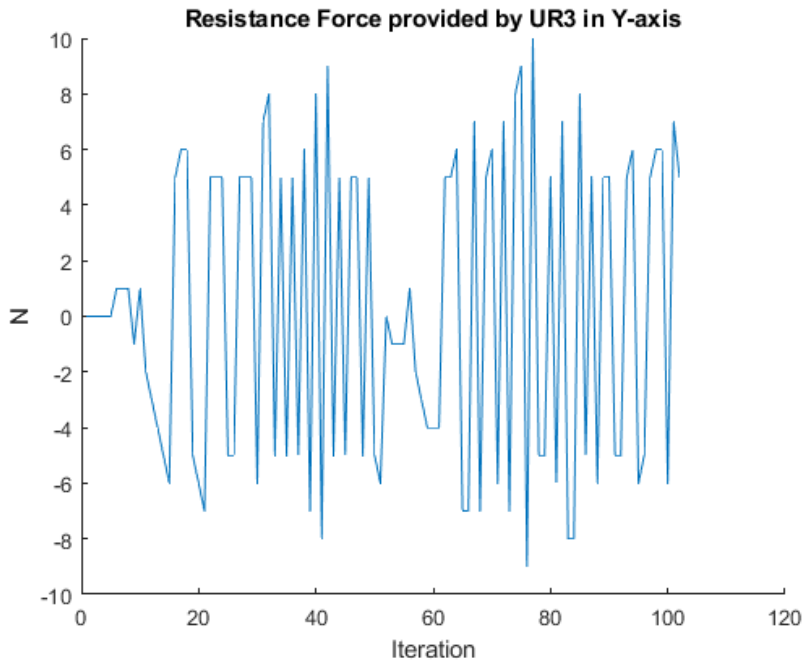


Figure B.1: robot force in Y-axis

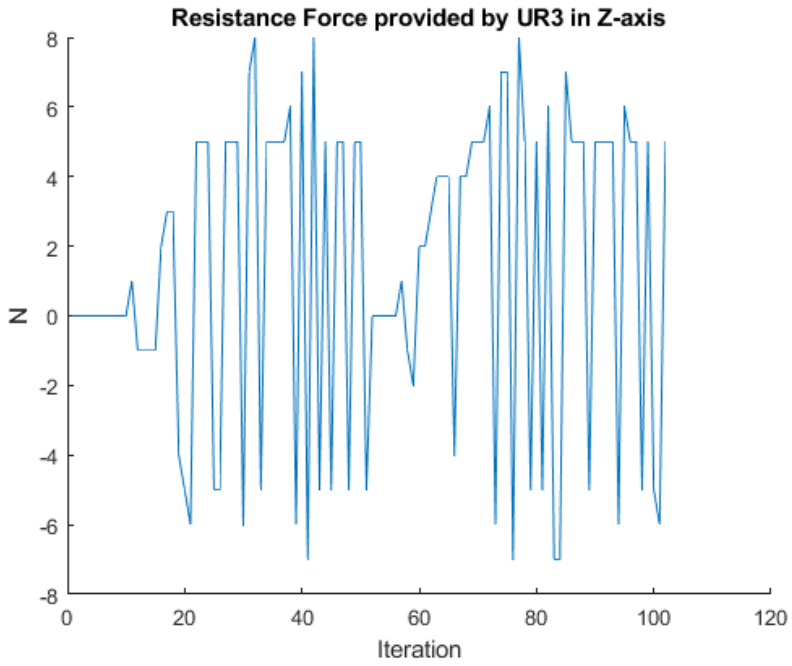


Figure B.2: robot force in Z-axis