Final Master Thesis

Master's Degree in Automatic Control and Robotics

REINFORCEMENT LEARNING FOR ROBOTIC ASSISTED TASKS

MEMORY

Author :	Aniol Civit Bertran
Director :	Guillem Alenyà Ribas
Codirector :	Cecilio Angulo Bahón
Call :	June, 2020

ETSEIB

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Institut de Robòtica i Informàtica Industrial

Abstract

The increment of dependant people for the realization of Activities of Daily Living is a fact that comes implicit with the increase of population and their lifespan given by the modern technological progress. For this reason, Assistive Robotics is a field that is currently researched and gradually implemented. In this project, two different algorithms to train robots to perform those tasks are being studied and compared, Reinforcement Learning and Imitation Learning. The main objective is to learn the benefits, drawbacks and suitability of both for a given task. To do the experimentation, the Assistive Gym software will be used, to train the nets with Reinforcement Learning and to perform the movements with Dynamic Movement Primitives learned from a demonstration. The environment will consist in a Sawyer robot cleaning the right arm of a human lying down on a bed. The potential of each methodology will be tested in different scenarios, having the user's arm straight or partially flexed. Furthermore, small implementations to improve the execution of the task will be implemented for the Dynamic Movement Primitives to test the performance and the similarity to the demonstration. Both methodologies are capable to perform the desired task successfully, the model trained with Reinforcement Learning is more suitable for an individual user that will use the robot in a long term, due to the training time. The Imitation Learning algorithm is faster to teach and its trajectory and velocities are easily adaptable to the preferences of the user, it is more suitable in places where there is a high rotation of users that require urgent and short attention, such as clinics or hospitals.

Keywords: Assistive Robots, Reinforcement Learning, Imitation Learning, Dynamic Movement Primitives, Compliance Control, Activities of Daily Living, Machine Learning, Deep Learning.

Contents

List of Figures

List of Acronyms

ADL Activities of Daily Living. 9, 11, 45

DMPs Dynamic Movement Primitives. 18, 21, 27

DOF Degrees of Freedom. 28

IL Imitation Learning. 24, 25

PPO Proximal Policy Optimization. 13, 16, 18, 25

RL Reinforcement Learning. 13, 24, 25

ROS Robot Operating System. 18

TRPO Trust Region Policy Optimization. 13

List of Tables

1 Introduction

It is a fact that recent technological advances in the society have provided tools to improve the life quality of people. Due to the increment of the lifespan there surges a human-dependency from people who suffer from different diseases or simply need help for Activities of Daily Living (ADL).

Several technological resources for robotics autonomy have been studied and are already implemented. However, human-robot interaction is still a concern in the assistive robotic scope. For this reason, the utility of assistive robots has been studied over the last years. Assistive robots manipulate the environment to provide assistance and give autonomy to people with disabilities. Their design is user-centered to warranty the maximum safety to the user. The key for social acceptance of assistant robots is their adaptability to the user preferences and their feeling of safety.

The most popular methodology to learn trajectories in assistive tasks in Robotics is Learning by Demonstration, in the form of Imitation Learning. This methodology has proven to achieve better results and easier computation than programming everything from scratch with more classical programming techniques. Another related machine learning technique, Reinforcement Learning, has proven to be a powerful tool for other robotic applications.

This project analyzes these two different methodologies for robots to learn an assistive task: a robot assists a human cleaning its right arm. The first methodology being implemented is Reinforcement Learning. In this case, the robot learns by obtaining rewards and penalties when performing the desired task. For the other methodology under consideration, Imitation Learning, the robot learns the task through an initial demonstration.

Training a robot using Reinforcement Learning to perform an assistive task would take lots of time and the security of the tester would be in risk. The best option to start training is through simulation. To simulate the tasks, the Assistive Gym, a software environment released in October 2019, has been used. It focuses on the assistive robotics field teaching different robots to perform Activities of Daily Living through Reinforcement Learning in a simulated environment. Since it is a new software, it is needed to explore its extensive content in order to assess a full knowledge of its full potential. Not much research has been still developed with the Assistive Gym software, the first paper being published in July 2020. Hence, the first steps are to learn how it works and how to modify it in order to develop different environments and robots.

1.1 Motivation

Knowing the viability to train robots to perform an assistive task through Reinforcement Learning can result in an advance of this field. Since assistive robots is a field that is currently being researched, this project provides a baseline on which methodology is better suited for different scenarios, improving the performance of that assistive task. This leads to provide better and more effective tools to the healthcare system and to individual handicapped users.

And personally, having the possibility to contribute in robotics research, and specifically improving the life quality of dependant people, is very motivating. Plus this research project has been carried out at the Perception and Manipulation's laboratory at the IRI in Barcelona, which is a Research Center with extensive knowledge in several robotic fields, amongst which a lab studying assistive robotics.

After two years of the Master's Degree in Automatic Control and Robotics, exploring and learning basics of the different fields of robotics, have given the chance and the ambition to deepen the knowledge in this specific branch of robotics. This project is also self challenging in the terms of proving the capacity of reaching good results with the knowledge and tools acquired at the Master's Degree.

1.2 Objectives

The main goal of this project is to evaluate both Reinforcement Learning and Imitation Learning algorithms for teaching robots to perform assistive tasks, remarking the benefits and the drawbacks of each and to select the most suitable method for a given scenario involving assistive robots.

Both methodologies will be implemented in the Assistive Gym software, simulating a Sawyer robot to clean the right arm of a human mannequin lying down on a bed. Their performance will be analyzed in two different scenarios, the first one is when the mannequin's arm has a straight configuration, the second one is when the arm is partially flexed. The latter configuration adds complexity in the movements and represents another challenge for the learning algorithms.

Furthermore, to improve the execution of the task in Imitation Learning, an active compliant control will be programmed and its fidelity with the demonstration and improvement of the action will be tested. Another experimentation included is the addition of a handicap for the user such as having a conflicting zone in the arm that must be avoided.

2 Background

2.1 Assistive Robots

A lot of literature has been written exploring robotic assisted tasks and its set of applications, for different robots from wheelchair-mounted arms to mobile manipulators. Assistive robots are the ones that aim to give aid or support to a human user. There are different types of robots capable of performing these tasks [8]:

- **Rehabilitation robots:** Biomedical engineering technologies being easily employed by patients, therapists, and clinicians increasing the ease of exercising for patients. They can rather be active assisted, active constrained, active resistive, passive or adaptive. In active assisted exercises, the robot performs a force to accompany the user's movement to make it easier or to difficult that movement so the user makes more effort on moving it. There are different kinds of rehabilitation robots, each one designed focusing on the rehabilitation of a part of the body, the objective of rehabilitation robots is to help the user recover faster or even better from a physical trauma. Robotized prosthetic body parts are included in this group as well as forearms stabilizers for people who suffer from tremors or Parkinson.
- **Mobility aided robots:** These robots provide support, guidance and even health monitoring for the movement of disabled people or with difficulties. They can be smart wheelchairs, exoskeletons, electronic sticks for blind people and walkers.
- **Companion robots:** Social companion robots that can do the chores, guard the house, teach children and keep company to the elderly or people with health issues. The focus is on the interaction with humans and their environment keeping the distance, they usually have human or animal shape.
- **Manipulator arms:** Designed for physically disabled people or with low autonomy for mobility. The objective of these robots is to give autonomy to those people and helping healthcarers to perform ADLs for needed people. Another strong power is the autonomy of the robots for decision-making, taking into account the possibility of being manually operated, in this case the user must have at least partial control of the upper limbs. They can be wheelchair mounted or mobile humanoid robots. Some examples are explained in Section 3.1.
- **Educational robots:** Discipline to introduce students to robotics and programming while learning other cognitive skills. It can adapt to all ages and difficulty levels.

2.2 Autonomous Assistive Robots for Activities of Daily Living

The tasks that are more commonly requested among adults with dependency associated with Activities of Daily Living are feeding, drinking, dressing, itch scratching, bed bathing and arm manipulation. This section focuses on different methods, algorithms and approaches to give autonomy to the robot to perform the commented actions.

For the feeding activity (see Figure 1) there are studies that determine the factors influencing good bite transfer. It helps for designing the primitives and developing a robotic feeding system that uses them to feed people autonomously [9]. There are other studies where the trajectories are computed by Learning by Demonstration [2]. Moreover, some even improve the computed trajectories by Parameterized Similar Path Search (PSPS), where the robot can improve a trajectory over a known cost function [17]. The detection of the type and cause of anomalies during the action in the real system has also been studied with good results [14]. Many improvements have been made such as the online actualization of the trajectory while the user is moving and even the recognition of the face expression to make decisions whether giving food or not [13].

Drinking is an action with similar characteristics than feeding, where the cup and the fork or spoon should keep an accurate orientation during the trajectory and when performing the action close to the mouth. Brain-machine interfaces for patients who suffer from paralysis are presented in [20].

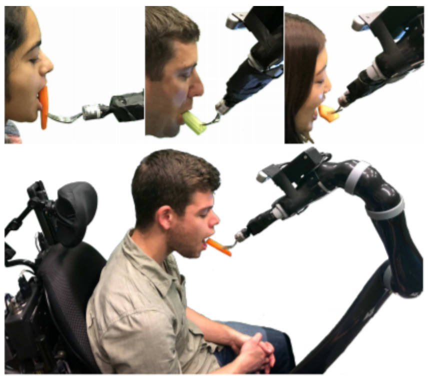
Figure 1: The Jaco robot feeding users using different manipulation strategies. Source: [9]

The dressing action (see Figure 2) has been a significant focus studied in recent years. In this action the handling of non-rigid materials is another difficulty to add to the problem studied independently. Referring to the assisting movements, in [12] the user is involved by re-positioning requests. The robot and the user take turns, a visual module captures the human's motion to determine if it's following the re-positioning requests. This way the robot learns the constraints of the user and can apply it in assistive tasks like dressing.

It is shown in [5] how to estimate in real time the local pose of a human limb using a multidimensional capacitive sensing technique. A neural network model estimates the position of the closest point between the end-effector and a person's limb and the orientation of the limb's central axis. This is proved to be useful with two activities of daily living, bathing and dressing.

In [16] it is proposed a programming by demonstration method to efficiently learn and adapt skills for dressing assistance getting information relative to the user and relative to the robot. This method needs few demonstrations to learn the action and it adapts the trajectory online in case the user moves. In [1] the motion is learned by human demonstration and they join high-level symbolic planning with low-level motion primitives through Hidden Semi-Markov Models that also fits the user's preferences, can readjust the trajectories based on the user movements and can also handle foreseen situations.

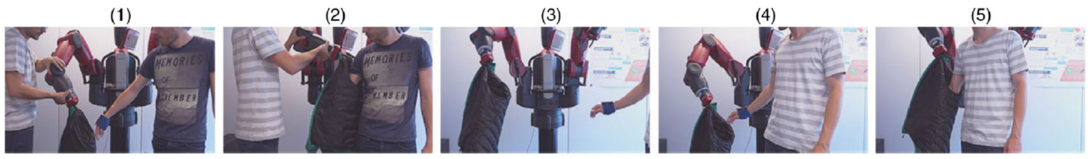

Figure 2: Demonstration and reproduction of a dressing task on Baxter robot. Source: [16].

In Figure 2(1,2) the demonstration process is occurring. Next, in Figure 2(3–5) the robot reproduces the process.

The first apparition of reinforcement learning in assistive robotics is in [22], which action consists on T-shirt clothing assistance, and rewards are penalties of the distance of the neck points of the T-shirt and the neck of the mannequin, and the distance of the sleeves of the T-shirt and the arm of the user.

The research in [4] explores the forces applied to a person's body by the garments in order to improve the effectiveness of the assistance of the robots.

In [25] they propose a system that learns a haptic classifier for the outcome of the task given few real-world trials with one person. The system optimizes the parameters of a physics simulator using real world data, then Hidden Markov Models are trained on the data collected. Next, then they can classify and predict the outcome of the assistive task based on measures of the end-effector. In [6] they propose a deep recurrent model that predicts the forces a garment will apply to a person's body, and show that with this knowledge the robot can provide a better assistance.

Different control strategies for dressing are also studied. Clegg et al. presented a co-optimization approach for a KUKA IIWA [3] robot and person to learn an assistive dressing task in simulation.

In [10] researchers have developed the Wouse, which is a device for detecting wincing from skin movement near the eye, consisting of optical mouse components mounted near a user's temple via safety googles. This device is used for semi-autonomous people, who have more autonomy than users with paralysis. Those people are more exposed to the less restricted robot in case of an anomaly in the trajectory of the robot and may have not enough capacity to push a run-stop button.

2.3 Proximal Policy Optimization

Proximal Policy Optimization (PPO) [21] is the Reinforcement Learning (RL) algorithm to be used for training the robot in the desired environment. It is a relatively new policy gradient method for reinforcement learning, which alternates between sampling data through interaction with the environment and optimizing an objective function using a stochastic gradient ascent. There are many different objective functions that PPO can use, which will be explained below.

Policy gradient methods compute an estimator of the policy gradient and plug it into a stochastic gradient ascent algorithm. The most common has the form,

$$\hat{g} = \hat{\mathbb{E}}_t\Big[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t\Big]$$

where π_θ is a stochastic policy, s_t are the states of the system, a_t are the actions and $\hat{A}_t$ is an estimator of the advantage function at time step t. The expectation indicates the empirical average over a finite batch of samples. The estimator $\hat{g}$ is obtained from the differentiation of the objective function,

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t\Big[\log \pi_\theta(a_t|s_t)\hat{A}_t\Big]$$

The optimization for this loss function often leads to destructively large policy updates.

In Trust Region Policy Optimization (TRPO) an objective function is maximized using a penalty instead of a constraint in order to simplify the optimization problem, resulting the optimization,

$$\max_\theta \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t - \beta KL[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)]\right]$$

This adds the difficulty of choosing a proper value of β that performs well in different problems, or even the same problem where the characteristics change due to the learning, and even some studies prove that it isn't sufficient. θ_{old} is the vector of policy parameters before the update.

Clipped Surrogate Objective denotes the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$. TRPO maximizes a "surrogate" objective

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]$$

CPI refers to Conservative Policy Iteration. Without a constraint, maximization of L^{CPI} would lead to an excessively large policy update. In order to penalize changes to the policy that move $r_t(\theta)$ away from 1 the objective function is modified,

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

where ϵ is a hyperparameter. This new objective function includes the L^{CPI} and avoids r_t for moving outside the interval $[1-\epsilon, 1+\epsilon]$. The minimum of the clipped and the unclipped objective is taken, so the final objective is a lower bound, this way the change in probability ratio is ignored when it would make the objective improve, and it is included when it makes the objective worse.

An alternative or addition to the Clipped Surrogate Objective is the Adaptive KL Penalty Coefficient. It uses a penalty on KL divergence and adapts that coefficient to achieve some value of the KL divergence d_{targ} each policy update. The procedure is to optimize the objective function,

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t - \beta KL[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)] \right]$$

and compute,

$$d = \hat{\mathbb{E}}_t \left[KL \left[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t) \right] \right]$$

so,

$$\begin{aligned} d < d_{targ}/1.5 \quad &\longrightarrow \quad \beta \leftarrow \beta/2 \\ d > d_{targ} \cdot 1.5 \quad &\longrightarrow \quad \beta \leftarrow \beta \times 2 \end{aligned}$$

This updated β is used for the next policy update. This parameter can have punctual significant different values but it quickly adjusts. The initial value, with the 1.5 and 2 values are chosen heuristically, the algorithm quickly adjusts the β value and the coefficients don't influence much the algorithm.

The surrogate losses from the previous baselines can be computed and differentiated with a minor change to a typical policy gradient implementation. Most techniques for computing variance-reduced advantage function estimators make use of a learned state-value function $V(s)$. If using a neural network that shares parameters between the policy and the value function, the loss function selected must combine the policy surrogate and a value function error term. This objective can be augmented by adding an entropy bonus to ensure sufficient exploration. Combining all this results in:

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t \left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t) \right]$$

where c_1, c_2 are coefficients, S denotes an entropy bonus and L_t^{VF} is a squared-error loss $(V_\theta(s_t) - V_t^{targ})^2$.

One style of policy gradient implementation that is well-suited with recurrent neural networks, runs the policy for T timesteps, and uses the collected samples for an update. This requires an advantage estimator,

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T)$$

where t specifies the time index in [0,T]. Generalizing this estimator, it can be reduced,

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$

An example of a Proximal Policy Optimization algorithm that uses fixed-length trajectory segments is shown in Algorithm 1. Each N actors collect T time steps of data. Then the surrogate loss on these $N \cdot T$ time steps of data is constructed and optimized with Adam optimizer, for K epochs.

The comparison between different surrogate objectives under different hyperparameters has been developed through different scenarios. And also with different methods from the literature obtaining a better overall performance.

The policy is a fully-connected MLP with two hidden layers of 64 units, and hyperbolic tangent nonlinearities, outputting the mean of a Gaussian distribution. The best performance is for the clipping algorithm with a ϵ value of 0.2.

Algorithm 1: PPO, Actor-Critic Style

```
Result: $\theta$
initialization;
for $iteration = 1,2 \dots$ do
    for $actor = 1,2,\dots,N$ do
        run policy $\pi_{\theta_{old}}$ in environment for T ;
        Compute advantage estimates $\hat{A}_1, ..., \hat{A}_T$ ;
    end
end
Optimize surrogate $L$ wrt $\theta$, with $K$ epochs and minibatch size $M \leq NT$ ;
$\theta_{old} \leftarrow \theta$ ;
```

3 Resources

This project has been developed at the *Institut de Robòtica i Informàtica Industrial* (IRI).

In this section it is shown the used tools to carry out the project, which basically consist on the Assistive Gym software and the Robot Operating System.

3.1 The Assistive Gym

Assistive Gym [7] is an open source physics-based simulation framework, developed in Python, for physical human-robot interaction and robotic assistance. The difference between this simulation environment and others is the focus on the interaction between robots and humans, which has been studied in depth by the designers.

Its environments are built in the open source PyBullet physics engine, and the strategy used to teach the robot is through Reinforcement Learning algorithms. At each time step, the robot records observations from the state of the system, executes an action according to a control policy and then receives a reward. Those observations are properties that can be obtained from a real-world assistive robotics scenario, such as positions and orientations of parts of the robot or the human, the interaction forces between them and characteristic features of the action performed, e.g. the food velocity at the feeding action or the quantity of water poured into the user's mouth.

The deep reinforcement learning technique called Proximal Policy Optimization (PPO) is used, as explained in Section 2.3. It is a policy gradient algorithm, using a fully-connected neural network with two hidden layers of 64 nodes. To perform the PPO it is used a library based on PyTorch that lets the developer train and evaluate different models even with multi agent policies using co-optimization.

It is integrated into the OpenAI Gym framework, that is a toolkit for developing and comparing Reinforcement Learning tasks, which allows to introduce physics-based environments where robots can assist people with six different tasks in Assistive Gym's case. These tasks are the ones that are more commonly requested among adults with dependency associated with Activities of Daily Living (see Figure 3 for a representation of each task):

- **Itch Scratching** A small scratching tool is hold by the robot's end-effector, it has to reach a target placed randomly by the software and perform scratching motions near that target.
- **Bed Bathing** The user lies on a bed in a resting position and the robot has to use a washcloth tool to clean off the right arm. To check the proportion of the arm that is cleaned, it is filled with particles uniformly distributed along the arm.
- **Dressing** A robot holds a hospital gown and must pull its sleeve up a person's left arm.
- **Drinking** The robot holds a cup filled with small particles simulating the liquid, the objective is to place those particles inside the mouth without spilling them.
- **Arm Manipulation** The objective of this action is to place the right arm of a user on the bed and close to its torso in case that the arm is hanging off the bed.
- **Feeding** The action consists on feeding a user that sits on a chair with a spoon placed at the end-effector of the robot full of particles simulating food.

For every different action and environment the starting position and orientation of the robot are calculated by optimization in order to achieve a better performance. Rewards are implemented by

[4]Image source: https://github.com/Healthcare-Robotics/assistive-gym

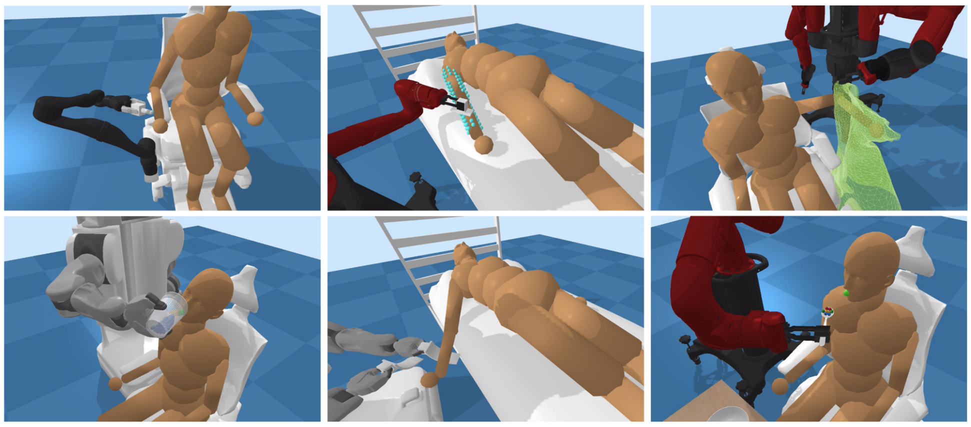
Figure 3: Four collaborative robots in Assistive Gym performing six different tasks, as listed in the text.[4]

succeeding in the mission of the action, always taking into account the human preferences and the interaction forces between the robot and the human. It is crucial to consider the human preferences since it is a keypoint for the acceptance of the assistive robots.

Assistive Gym also provides tools for the researchers to develop their own environments, including robots and tasks with different sizes and joint limits, and even different control policy learning algorithms which can be easily compared.

The robots that are implemented are four commercial robots, as depicted in Figure 4: Jaco, Sawyer, Baxter, and PR2 robots. However, as commented before, other robots can be easily introduced, such as TiaGo that is the one that the laboratory where the project has been developed owns. To summarize, there are 48 different combinations of robots, tasks and possible co-optimization.

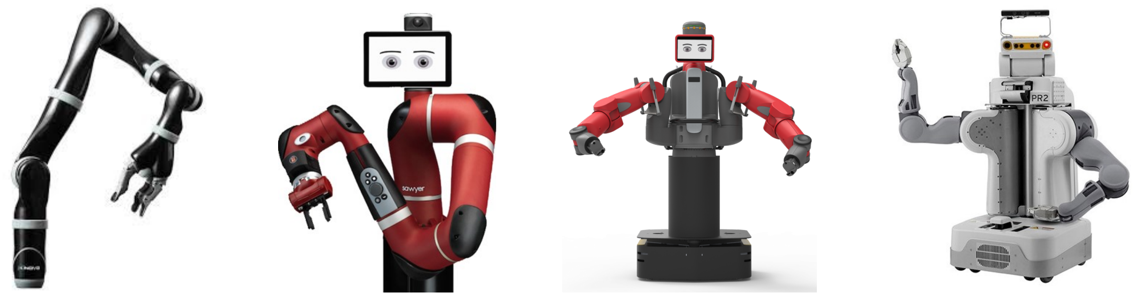
(a) Jaco Robot (b) Sawyer Robot (c) Baxter Robot (d) PR2 Robot

Figure 4: Robots included in Assistive Gym.[5]

Simulated human joints have been modelled with the same limitations as the real human body, because the objective is to make the robots learn to provide safe assistance without creating discomfort. Furthermore, software takes into account the possibility of some form or limited motor functionality of the user, and robots learn to provide better assistance for those cases by modelling them as a co-optimization problem in which both human and robot are trained.

Default male and female human models are provided, with dimensions and joint limits matching published 50th percentile values [23], as it is shown in Figure 5. It is remarkable that the collisions between different parts of the body are enabled, and it has 40 controllable joints. The limitations modelled are: head and arm tremor, joint weakness and limited range of motion.

In order to perform the training for the control policies, the designers of assistive gym use Proximal

[5]Images sources: (a) http://servo.com.my/products/jaco/, (b) https://www.active8robots.com/services/certification-services/, (c) https://www.freepng.es/png-xziiti/download.html, and (d) https://robots.ieee.org/robots/pr2/.

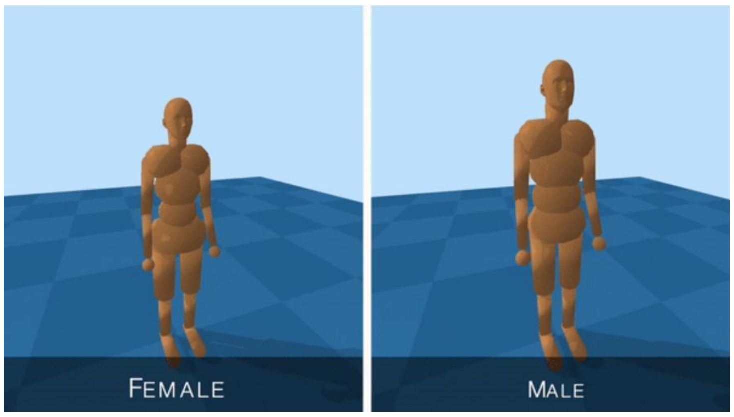

Figure 5: Default female, at left, and male, at right, models in Assistive Gym.

Policy Optimization (PPO) through a PyTorch library they also coded. In order to use this library it is mandatory to have installed the OpenAI Baselines library. For each task the robots are trained a total of 10,000,000 time steps, or 50,000 simulation rollouts (trials) for 36 agents. Each rollout consists on 200 time steps that result in 20 seconds of simulation. The policy executes a new action at each time step. A 10 epoch update of the policy is performed after each actor completes a single rollout. It has been proved [23] that PPO is able to learn reasonable control policies for all four robots.

Finally, the designers compare the trained policies for each of the 48 possible combinations, 100 trials each environment to evaluate the performance of each robot for an specific task.

Their study proves that Assistive Gym is a powerful open source framework for the development of autonomous robots that can provide physical assistance.

3.2 The Robot Operating System

The Robot Operating System (ROS) is a flexible framework for writing robot software that provides a collection of tools and libraries. It aims to facilitate developing complex and robust robot applications across a wide variety of robotic platforms. ROS provides the same standard services than an operating system has, such as hardware abstraction, implementation of shared functionalities, device control at low level, data packages management and message transmission between processes. ROS is not an OS, it's a framework using the concept of an OS. Its *modus operandi* is based on communication between modules called nodes by message exchange of the following types:

- Topics: Named buses over which nodes exchange messages. They have anonymous publish and subscribe semantics. Nodes that are interested in the data that the topic is publishing can subscribe to it. There can be multiple publishers and subscribers to a topic. The communication is supposed to be unidirectional, hence the subscriber doesn't send information to the publisher, this way it isn't possible to know if the data has been received correctly. The information sent is strongly typed by the message's type used to publish, nodes can only receive messages with a matching type.

- Services: A method for request/reply, a ROS node, called server, offers a service under a string name, and a client calls the service by sending a request and waits for the reply.

- Action: They have the same communication working mode as services, server/client with concrete specifications regarding to the information of the performance of an action such as Goal, Feedback and Result messages.

ROS has an explicit package [22] that provides a general implementation of Dynamic Movement Primitives (DMPs). It provides a stable reformulation of DMPs as later described in Section 4.2.

Multi-dimensional DMPs can be learned from an example trajectory. It can also generate full and partial plans for an arbitrary starting and ending points as well as defining the parameters of the DMPs.

Using this package a trajectory can be defined by the points and the velocities at every point. Other features like force or acceleration are unknown. The methodology to implement N-dimensional DMPs is to build N separate DMPs and link them together with a single phase system. The function approximation is computed by local linear interpolation, but they provide code for a global function approximator using Fourier basis, along with another local approximation scheme using radial basis functions in cases of having a big amount of data.

The ROS node provides three services:

- LearnDMPFromDemo: Given a trajectory as demonstration and the DMP parameters, returns a learned multi-dimensional DMP. The input DMP parameteres are the proportional gains K for each of the dimensions of the DMP, the damping gains D for each of the dimensions of the DMP. They recommend to set this parameter as $D = 2\sqrt{K}$. And the last parameter is the number of basis functions to use, this parameter increases as it does the complexity of the movement to perform regarding not to overfit the model.

- SetActiveDMP: Sets the DMP that will be used for planning as active.

- GetDMPPlan: Creates a full or partial plan from a starting point to the goal point using the active DMP. The input parameters for this service are:

  * x_0: Starting point or current state of the robot.
  * $\dot{x}_0$: The first derivative of state from which to begin planning.
  * t_0: Time in seconds from which the plan begins.
  * Goal: The goal of the trajectory where the DMP should converge.
  * Goal threshold: A threshold for each dimension of the DMP's goal that the plan achieve come before stopping planning, unless it plans for segment length first.
  * Segment length: The length of the plan segment in seconds, this can be used to do piecewise, incremental planning and replanning. If it is set to -1 the planning will continue until the desired convergence.
  * τ: This is the time constant for the DMP. In case that the same value for LearnDMPFromDemo is chosen, then the returned trajectory and the demonstration will be performed at the same velocity.
  * dt: The time resolution of the plan in seconds.
  * Integrate iterations: The number of times to numerically integrate when changing acceleration to velocity to position. They recommend to set this value to 1 unless dt is large (i.e. greater than 1), in which case it should be larger.

4 Design and Implementation

In the following section the theoretical background and the implementation of the work will be described. As it is commented in the objectives, one of the key points is to perform correctly an action from the software Assistive Gym through Imitation Learning, which can be adjusted through compliant control.

4.1 Bed Bathing Environment

The chosen environment that better suits the requirements of the project is Bed Bathing. This is given by the factor that the human-robot interaction in this action implies a practically continuous contact of the tool with the human, this allows to perform a compliant control algorithm in the trajectory given by the DMPs.

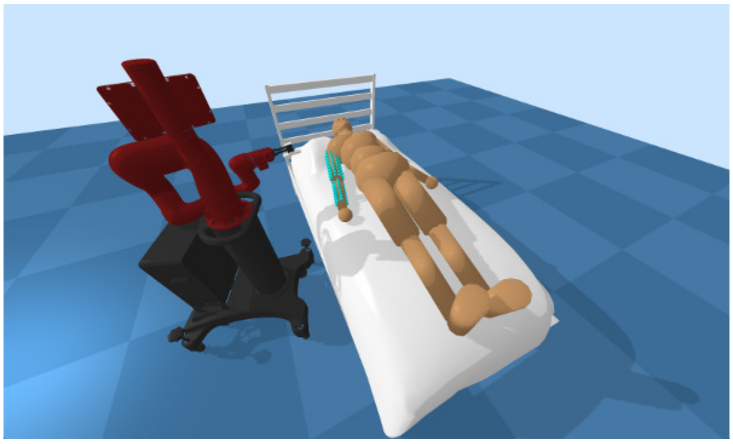

Figure 6: Sawyer robot about to perform the Bed Bathing task in Assistive Gym.

Bed Bathing consists in a person that lies on a bed in a random resting position while a robot must use a washcloth tool to clean off the person's right arm, as depicted in Figure 6. The robot is rewarded for moving its end effector closer to the person's body and for wiping the bottom of the washcloth tool along the surface of the person's arm. There are uniformly distributed markers around the person's right arm. The mission of the robot is to wipe off these markers. The training of the model for Reinforcement Learning is given by the reward:

$$\begin{aligned} r &= w_d r_d + w_a r_a + w_w r_{ncp} \\ hp &= w_v r_v + w_{fnt} r_{fnt} + w_{hf} r_{hf} \end{aligned}$$

where w_d and r_d are the weight and the reward for the distance between the tool and the human. The reward is calculated by the minimum distance. w_a and r_a are the weight and the reward for the action that the robot will perform, the reward is a penalty calculated by the negative of the sum of the absolute value of the output of the neural network with every element clipped between 1 and 0. The output of the neural network is the robot joints variation. w_w refers to the weight for wiping, and r_{ncp} is the reward of the new contact points, which mean the number of marks along the arm that have had contact with the tool.

For the second part, hp, is the set of rewards given by the accomplishment of the human preferences. w_v and r_v are the weight and the reward for the end-effector's velocity. The reward is computed by the negative value of the end-effector's velocity. w_{fnt} and r_{fnt} refer to the forces that the robot apply to the human, apart from the tool's force. The reward is a penalty of that force value. And the last terms w_{hf} and r_{hf} are another penalty for high forces. If the force applied to the human is superior than 10N then the reward is a penalty of that force.

Once the model is trained with Reinforcement Learning and the reward policy explained above, the procedure of the robot to perform the action starts with the initial position, at each time step, the robot records observations of the environment that could be obtained in the real-world scenario. It gets the distance vector between the robot's torso and the end-effector, the orientation of the tool, its joint positions, the distance vector between the shoulder, the elbow and the wrist positions from the robot's torso and the forces that the tool and the robot apply to the human. This information is the input of the trained policy model and it returns the output that is the change the joints must do. Assistive Gym checks if the movement exceeds the limits of the joints.

The chosen robot for this action is the Sawyer, because as it shows Table 1 it is the one that performed the action the best after the same training for all the robots and has bigger workspace than Jaco.

Task	PR2	Jaco	Baxter	Sawyer
Bed Bathing	86.7	104.4	88.4	**109.0**

Table 1: Mean reward for 100 trials.

4.2 Dynamic Movement Primitives

Complex movements have been thought to be a composition of primitive action 'blocks' in sequence. Dynamic Movement Primitives (DMPs) [15] are a nonlinear dynamical system formalization of those primitive blocks.

DMPs can generate discrete and rhythmic movements. For discrete movements the base system is a point attractor meanwhile for rhythmic movements it is a limit cycle. From an engineering point of view, the movement can be interpreted as a linear spring system perturbed by an external forcing term,

$$\begin{aligned} \tau \dot{v} &= K(g-x) - Dv + (g-x_0)f \\ \tau \dot{x} &= v \end{aligned}$$

where x and v are position and velocity of the system, x_0 and g are the start and goal positions. τ is a temporal scaling factor. K acts like a spring constant, and D is the damping term, which is chosen such that the system is critically damped.

This forcing term f is the nonlinear function that helps getting the desired behaviour for the system. DMPs uses an additional nonlinear system to define the forcing term over time, giving the problem an easily solvable problem with a well defined structure. The forcing function f is defined as,

$$f(s) = \frac{\sum_{i=1}^{N} w_i \psi_i(s) s}{\sum_{i=1}^{N} \psi_i(s)}$$

where

$$\psi_i = \exp(-h_i(x-c_i)^2)$$

w_i is a weight for a given basis function ψ_i that defines a Gaussian centered at c_i, where h_i is the variance. This means that the forcing function is a set of Gaussians where they activate as the canonical system converges to its target. The function f depends on a phase varibale s, which changes from 1 to 0 during a movement and it is obtained through:

$$\tau \dot{s} = -\alpha s,$$

where α is a pre-defined constant. This equation refers to a canonical system, what means that the system converges to the desired goal g since $f(s)$ vanishes at the end of a movement. The weights can be learned to generate any desired smooth trajectory as it can be observed in Figure 7.

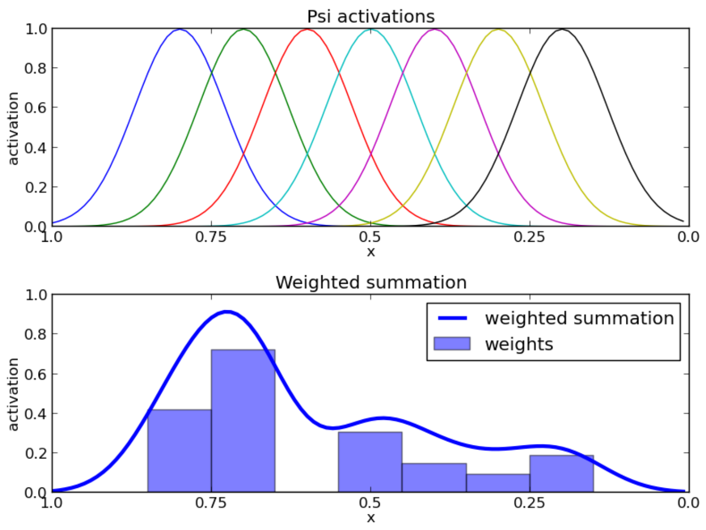

Figure 7: Activations of Gaussians ψ in the top image and the weights w_i and the one-dimensional forcing function f (blue line) in the bottom image in the case of s going from 1 to 0. [6]

The equations are spatial and temporal invariant, it isn't necessary to adjust the weights in order to change the goal or the velocity of the trajectory. Those movements generated are robust against perturbation due to the attraction dynamics of the system.

To learn a movement from demonstration it is needed the trajectory $x(t)$, then its derivatives can be calculated for each time step. Then the canonical system is integrated, $s(t)$ is computed for an adjust of the temporal scaling τ. $f_{target}(s)$ is computed as,

$$f_{target}(s) = \frac{-K(g-x) + Dv + \tau\dot{v}}{g - x_0}$$

The weights w_i are computed by minimizing the error $J = \sum_s (f_{target}(s) - f(s))^2$. This is a regression problem that can be solved efficiently.

A movement is generated by reusing the weights w_i and specifying a desired start and goal, setting $s = 1$ and integrating the canonical system.

To adapt the movement to a desired goal, a modification of the DMP formulation must be done. The current formulation has three drawbacks. If the goal and the initial positions are the same, the non-linear factor won't have effect on the system, so the system will remain at x_0. If the goal and the initial positions are very close, a small change in g can lead to huge accelerations which can damage the robot. At last, when assigning a new goal, the terms $(g_{original} - x_0)$ and $g_{new} - x_0$ are compared in sign, so depending on the new goal the trajectory can be mirrored.

To avoid this problems the DMPs have been reformulated, but keeping the good properties. The result is the following,

$$\begin{aligned} \tau\dot{v} &= K(g-x) - Dv - K(g-x_0) + Kf(s) \\ \tau\dot{x} &= v \end{aligned}$$

The same canonical system is used, the difference is that the non-linear function is no longer multiplied by $(g - x_0)$. The term $K(g - x_0)$ avoids jumps at the beginning of a movement. The target function $f_{target}(s)$ is computed,

$$f_{target}(s) = \frac{\tau\dot{v} + Dv}{K} - (g-x) + (g-x_0)s$$

[6]Figure source: `https://studywolf.wordpress.com/2013/11/16/dynamic-movement-primitives-part-1-the-basics/`

This new formulation is used in the performed simulations.

To implement the DMPs, a script in Python using the ROS package [18] has been programmed and slightly modified to work with the data from the simulations of the selected environment and robot in Assistive Gym. The script takes as inputs the demonstration's trajectory, with its seven degrees of freedom, the starting and goal points, and the implicit parameters of the DMPs, such as the proportional gain K, the damping gain D, τ and the number of basis functions. The output is a trajectory starting and ending at the desired and declared goals and with a similar shape to the demonstration model.

4.3 Compliance Control

Active compliance control is an increasingly employed technique used in the robotic field such as haptics, safe grasping [19], service robots, virtual reality, telemanipulation, human augmentation and the focus of this project which is assistant robotics [11]. The compliance control uses force feedbacks. There are two different categories of compliance control, the force control and the impedance control.

Force control is a control technique where both desired interaction force and robot position are controlled. A desired force trajectory is commanded and the actual force is measured in real time for the feedback control. This kind of control requires having force sensors.

Impedance control is based on position control which requires position commands and measurements in order to close the feedback loop. Force measurements are also needed to compute the target impedance characteristic. It uses different relationships between acting forces and manipulator position to adjust the mechanical impedance of the end-effector to the external forces. It can be stiffness (position proportional), damping (velocity proportional) or general (position, velocity and acceleration proportional).

For this project the interesting application of the compliance control is to control the interaction force between the human and the robot in order to respect the human preferences in such a manner that the robot doesn't exceed a maximum tolerated force, hence the needed control is the force control. During the trajectory given by the DMPs, when the interaction force between the human and the robot is bigger than the maximum, a modification of the trajectory is applied so the tool applies less force, on the human's arm, taking into account that in the trajectory it only touches the arm. The trajectory is modified in such a manner that it gets the point of contact where the maximum force is applied, between all the contact points, and the end-effector is moved away 0.5cm in the normal direction of the point of the contact in order to reduce the force applied.

To be consequent with the action, another implementation of an active compliance control is computed. At each time step the robot checks the total force that it is applying to the human, if the resulting force is zero means that there is no contact between the tool and the user's arm, so the arm is programmed to force small contact forces to better perform the task.

4.4 Data Acquisition

To perform the movement of cleaning the right arm of the user, the necessary data has to be obtained from a demonstration of the trained net using Sawyer, which is the robot that has the best performance between the robots included in Assistive Gym as it was shown in Table 1. It has been collected from a performance of the robot in a simulation in Assistive Gym software of a descending movement, from the shoulder to the wrist, and of an ascending movement, from the wrist to the shoulder. Each time step of the simulation the system collects the world Cartesian coordinates and the orientation of the tool in quaternions, which means that the data collected consists on 7 parameters.

The choice of selecting the position and orientation of the tool of the robot instead of the joint space is given by the fact that in joint spaces, to perform the compliance control that is easier to implement with the position of the tool, it would be needed one more extra computational step because it needs the forward kinematics conversion from joints to coordinates. So working in Cartesian space makes the task computationally faster.

It has to be taken into account that the net is trained as explained in Section 4.1, with the human preferences commented there. The performance of the net starts at an initial optimized configuration of the robot, in Sawyer's case in a configuration having the end-effector over the human. The simulation begins from that position, going practically straight to the user's arm. The collected data starts at the point where the robot is about to make contact at the upper arm and ends after making a descending and ascending movement. The inertia that the robot has to go from the initial configuration to the upper-arm makes the data have a small curve approaching the torso of the human after making contact.

This project compares the two learning methods, RL and IL, in the Bed Bathing environment in two different scenarios. The first one is the default and simple process, where the user's right arm is straight placed next to the torso. The second one is focused in another natural placement of the arm when the user lies on the bed, where the arm isn't straight but has a light inclination of the elbow as it is shown in Figure 8.

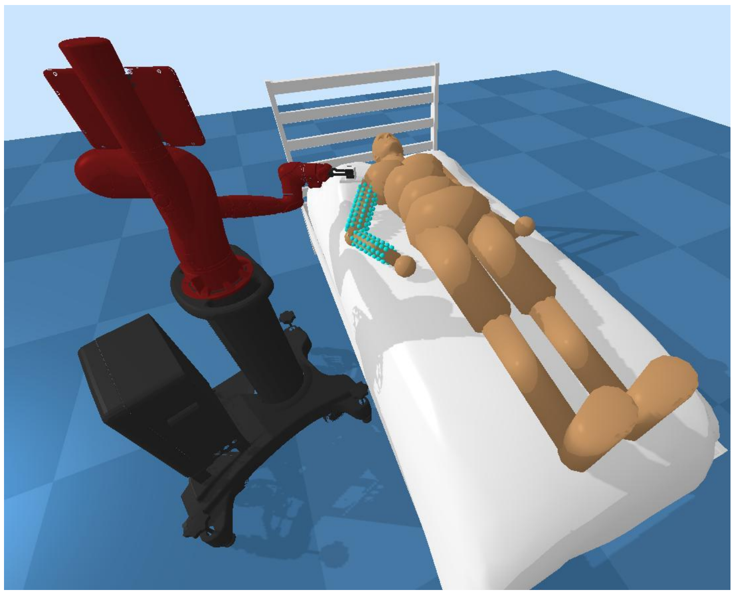

Figure 8: Bed Bathing environment with Sawyer robot and the user with a different arm configuration than the straight one.

5 Simulation and Results

This section aims to explain the methodology to compare both methods, RL and IL, and shows the results obtained from both. For this, different tests will be performed. Firstly, nets will be trained with Reinforcement Learning for the two configurations of the arm and its behavior will be examined. Secondly, the DMPs will be implemented for the straight arm configuration and their results will be compared with the trained net. Thirdly, an active compliance control will be applied to the Imitation Learning algorithm for both configurations of the arm and the results will be compared with the trained nets. Lastly, an improvement for the Imitation Learning algorithm will be programmed to avoid a conflicting zone of the arm.

5.1 Trained Nets

To compare the two methods it is necessary to start with the training of the nets, checking if the Reinforcement Learning algorithm performs the action correctly and in a real manner. PPO is used to train the control policy for the task, the methodology is explained at the section Section 2.3. In this section the nets are computed and supervised.

5.1.1 Straight Arm Scenario

As it is explained in Section 3.1, the trained policy for the straight arm in the Bed Bathing environment is the default one. This training consists on 36 concurrent simulation actors for a total of 10.000.000 time steps, or 50.000 simulation trials. Each trial consists of 200 time steps, 20 seconds of simulation at 10 time steps per second. A 10 epoch update is performed after each actor completes a single trial, every 7.200 time steps.

For this action the robot has learned reasonable control policies with a good performance in the task. The motion that it has learned is to clean the superior part of the task, the robot keeps cleaning the same part of the arm for the duration of the simulation, performing descending and ascending movements.

The policy has been evaluated over 100 simulation trials of the task, and the average reward and success are calculated. The success is the percentage of markers removed during the trial. For the Bed Bathing action with the Sawyer, the average reward is of 109.0 and the average success is a 24%. These results are for the PPO control policy when only training the robot. When the PPO is co-optimized for the robot and the human the results improve significantly, giving an average reward of 166.2 and a success of 81%. In that performance, the user has the arm resting until the robot has cleaned the superior part, then it moves approaching the robot and rotating the forearm about 180 degrees and the elbow, then the robot performs the cleaning of the new accessible zone. The performance of the robot for the co-optimization training can be appreciated in Appendix A.

The following Table 2 shows the interesting features of the performance of the simulation.

Feature	Value
Mean force (N)	2.26
Max force (N)	4.10
Mean velocity (cm/s)	19.46
Reward	126.47

Table 2: Properties of the trained net for the Straight Arm

ETSEIB

5.1.2 Flexed Arm Scenario

At first the policy has been trained the same time steps than in the Straight Arm Scenario, using 12 concurrent simulation actors for a total of 30.000.000 time steps, or 150.000 rollouts. The total training time has been of about 60 hours. The results of the performance show the robot is capable to successfully perform the task, the motion starts at the elbow, cleans the forearm until practically the wrist and then it moves back to the elbow and cleans that zone, it doesn't reach the superior part of the upper arm and the inferior part of the forearm leaving markers to clean, so the overall reward is lower than in the straight arm. At some points the robot does troublesome movements by trembling, and this could lead to insecurity or dissatisfaction from the user. The application of the forces is done punctually along the arm, and we observe that the tool isn't constantly applying low forces to the arm. The coordinates, velocities, and forces of the simulation can be appreciated at the Appendix B, the perturbations can be seen by the variation of the Z coordinate and the sudden changes of velocities.

The following Table 3 shows the interesting features of the performance of the simulation.

Feature	Value
Mean force (N)	2.09
Max force (N)	16.29
Mean velocity (cm/s)	12.86
Reward	87.07

Table 3: Properties of the trained net for the Flexed Arm

5.1.3 Discussion

In the environment with the Straight Arm the robot is able to perform a natural movement to clean the arm with the conditions defined at the algorithm. In addition, this movement is soft and continuously touching the arm.

For the Flexed Arm scenario the model has been able to learn cleaning the arm successfully, however, it shows some odd movements at some points of the trajectory. Tables 2 and 3 show that, with the same training time, the more complex scenario, with the arm partially flexed, executes the action slower and with a maximum force that exceeds the human preferences limit of 10N. To improve the performance more training of the model should be done.

Using co-optimization implies that the robot interacts and selects the action that it will perform depending on the user's movement, and the user does the same, this leads to more realistic human motions and improves the assistance from the robot. The co-optimized net has a peak of about 20N and others up to 10N, the velocities are of the order of 15 cm/s with a peak up to 38 cm/s approximately and in different circumstances they overpass the 20 cm/s.

5.2 First Implementation of the DMPs

Once the net model for the straight arm is trained enough to perform correctly the action, the data for the DMPs is extracted from a demonstration of it. In addition, the capacity of the DMPs to reproduce the learned movement is tested for different values of the temporal scaling factor τ as well as the conservation of the shape of the demonstration trajectory.

The parameters of the computed DMPs are, 7 dimensions, for the Cartesian coordinates of the end-effector's position and orientation, the constant $K = 100$, $D = 2\sqrt{K}$ and the number of basis is 4 due to the simplicity of the cleaning movement, if the movement had been more complex then a bigger value of number of basis would be needed.

The trained DMPs for the simulations use the Cartesian's coordinates and orientation in quaternions from the trajectory of the net, and create a movement from a desired initial point to an specific goal. The single action of cleaning the arm consists on two DMPs since the movements are the descending and the ascending movements, which are slightly different.

5.2.1 Setup

This experiment has been made with help of the Bed Bathing environment of Assistive Gym, and a script in Python that computes the DMPs, starts the environment and interacts with the robot to perform the desired trajectory. The selected robot to perform the tests is the Sawyer, given the fact that it is the one that has the best reward among the other robots implemented for the trained net as it was shown in 1.

In order to improve the performance of the robot for the action, four movements are computed, two ascending and two descending. This means that four DMPs are calculated before starting to move. This methodology also proves the capacity of adaptation of the DMP for different starting and goal points.

The initial and goal points are the upper points close to the shoulder and the wrist positions, as it can be appreciated in Figure 9. These positions are interchanged respectively depending on the movement that has to be performed, descending or ascending. For the second iteration, the initial and goal points are the same than the first ones but shortly displaced around the arm respect to the original points. These DMPs return a trajectory of the same dimensions that were introduced, giving the Cartesian coordinate points and the orientation of the tool in quaternion. The different and initial points have the particularity of having a different normal direction from the arm, the robot will have to adapt the motion in two different paradigms. The points from the upper-arm have been selected because the ones that are over them create a collision between the shoulder and the tool, this Figure shows the oversized dimensions of the shoulder.

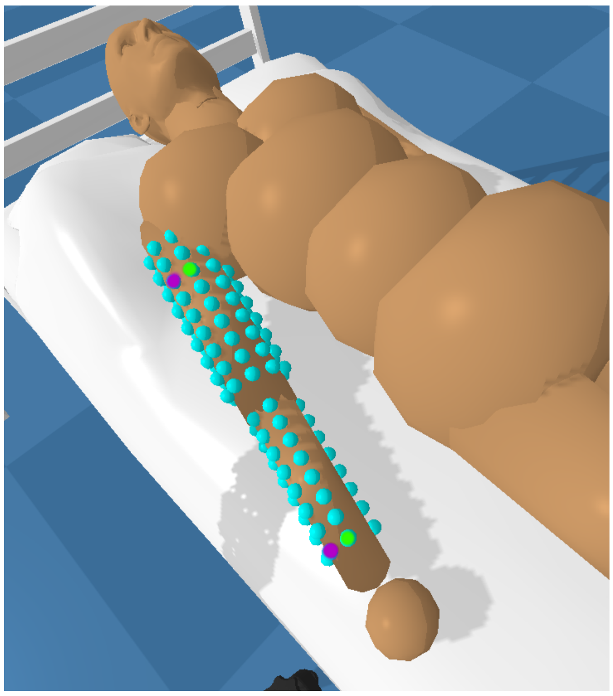

Figure 9: Initial and goal points for both DMPs, in green for the first motion and in purple for the second motion.

At this point the robot approaches the right arm of the user over the top of the shoulder to respect the starting position and orientation of the first descending DMP.

To move along the trajectory, in every time-step of the simulation, the coordinate and orientation points of the trajectory from the DMP are used to calculate the inverse kinematics of the robot. The

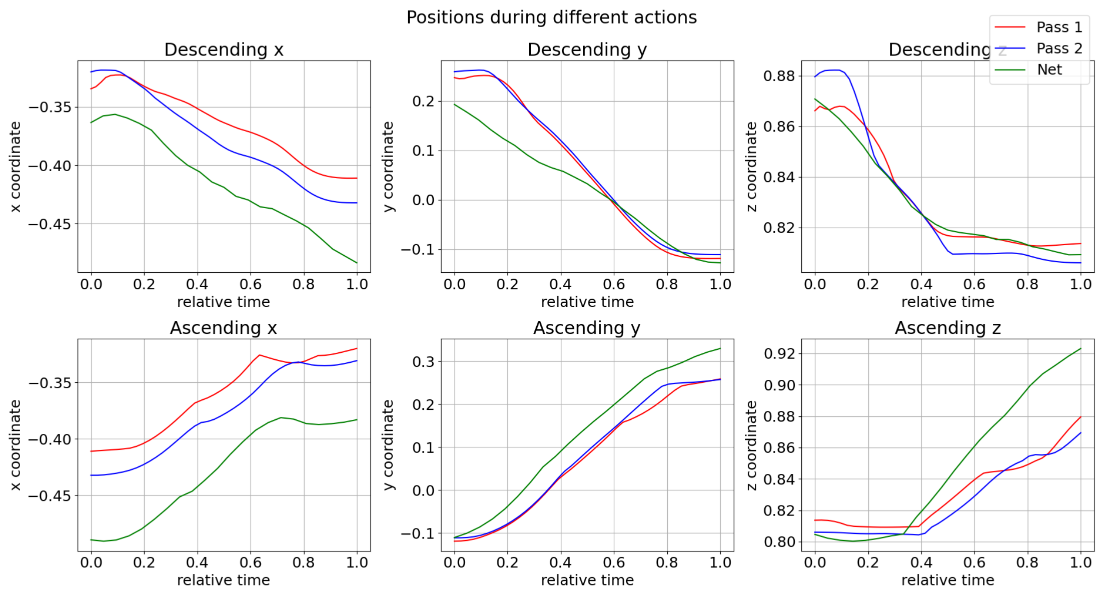

Figure 10: x, y and z coordinates trajectory of the DMPs and the Net for each movement.

inverse kinematics return the joint positions of the robot to have the end-effector at the desired position and orientation. To avoid different joint configurations, due to the 7 DOF of the robot arm, the inverse kinematics is calculated with a joint damping factor that makes the displacement of the joints continuous. The PyBullet's library offers a function that providing the desired joint configuration of the manipulable joints of the robot, through a default PD controller from pybullet it tries to reach that configuration.

During the motion of the computed DMPs the compliance control is checking the forces applied to the human and, in case of exceeding the maximum force tolerated by the human preferences (in the experiment is set to 10 Newtons), it calculates the normal of the contact point between the tool and the arm, and modifies the trajectory by moving it one centimeter away in that normal direction.

The goal of the experimentation is to verify if the performance of the DMPs calculated are similar to the net trajectory, that in this case is the demonstration to train them, to test the results, the trajectory among the forces, velocities and rewards calculated the same way than in 4.1. The smoothness of the trajectory will be checked as well.

5.2.2 Performance

The comparison of the Net and the DMPs has been carried out by comparing them along all the trajectory. This experimentation has been done with a value of the time scaling factor $\tau = 2$, which means that the trajectory is done at half the velocity than the demonstration. This comparison experimentation will be performed with different values of the time scaling factor τ and the results, which are similar, will be found at the Appendix C.1.

Figure 10 shows the evolution of the x, y and z coordinates for the two DMPs and the net cleaning actions in the descending and ascending movements. The red and blue lines are for the DMPs and the green is for the trained net with Reinforcement Learning, this graphs show the resemblance in the shape of the coordinate's evolution. It can be appreciated that the behavior of the trajectories is very similar even if they are displaced a little. The reason for this displacement is that the DMPs have the starting and ending point, and all the trajectory, defined to clean with the middle point of the tool, while the net is cleaning with an extreme o the tool, even with this differences the behavior is the same. The difference in the ascending subfigure in the z coordinate is given by the fact that the net ends the trajectory over the shoulder, and the ending points of the DMPs are different from that point. It is necessary to remark that the importance of this Figure is the similarity of the shapes and not the value of the coordinates to

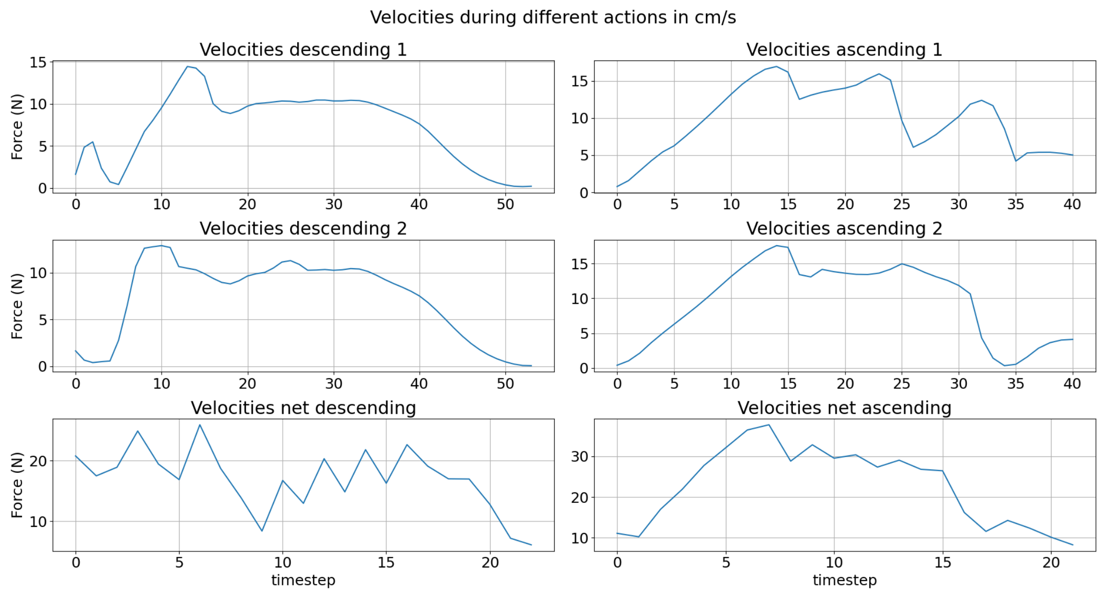

Figure 11: Forces of the DMPs and the Net along the trajectories.

be the same.

In Figure 11 it is shown the total velocities for both DMPs and the trained net. The important fact from this figure is to compare the shapes from the DMPs velocities with the shape of the trained net. Once seen that the trajectories are very alike then the velocity observed is the total velocity of the end-effector of the robot. Then, comparing the shapes of the velocities at simple view it can be seen that both shapes for the descending and the ascending action are similar, specially for the ascending action. In the descending action the net appears to be sharped at some points, taking a closer look to the descending trajectories of the coordinates it is appreciable how little displacements of the x and specially the z coordinate appear. To understand better the reason of this behavior it will be necessary to take a look at the Figure 12. Another factor to take into account is the number of time-steps of the simulation, as the DMPs have closely the double time-steps than the trained Net. The velocities from the DMPs appear to not have significant punctual changes and they are also very regular with an exception in the first ascending motion where the velocity decays considerably in a few time-steps, this fact will be better understood looking at the Figure 12.

The Figure 12 shows the evolution of the forces along the trajectory for the DMPs and the net. This figure shows the particularity that the net has learned to apply punctual forces to clean. With this consideration it can be understood like the net has learned to clean, removing the markers, just by pressing them. This occurs because of the implementation of Assistive Gym for the Bed Bathing task, it checks if the tool placed at the end-effector of the robot makes contact with the arm, if there is contact and at that place is a marker, the marker will be removed and the reward for removing it obtained. As the reward calculus penalizes the excessive forces, superior to 10 N, the trained machine will avoid applying excessive contact with the human's arm, this hypothesis would confirm the velocities behavior from the above figure. From the DMPs force figures some information can be extracted, in the first descending motion the tool practically doesn't make contact with the human's right arm, in the first ascending motion appears a high pick of force that checking with the velocities Figure and the simulation's performance a collision with the human's torso occurs, this is given by the influence of the demonstration, as it was explained in the Section 4.4. The second ascending motion shows more contact with the tool and the user's arm, the forces are very low, the peak change at the time-step 26 is given by the elbow. The human model is rigid and doesn't take into account its deformability, the forearm and the upper-arm of the human are cylinders of different radius at it can be seen at Figure 5. This factor

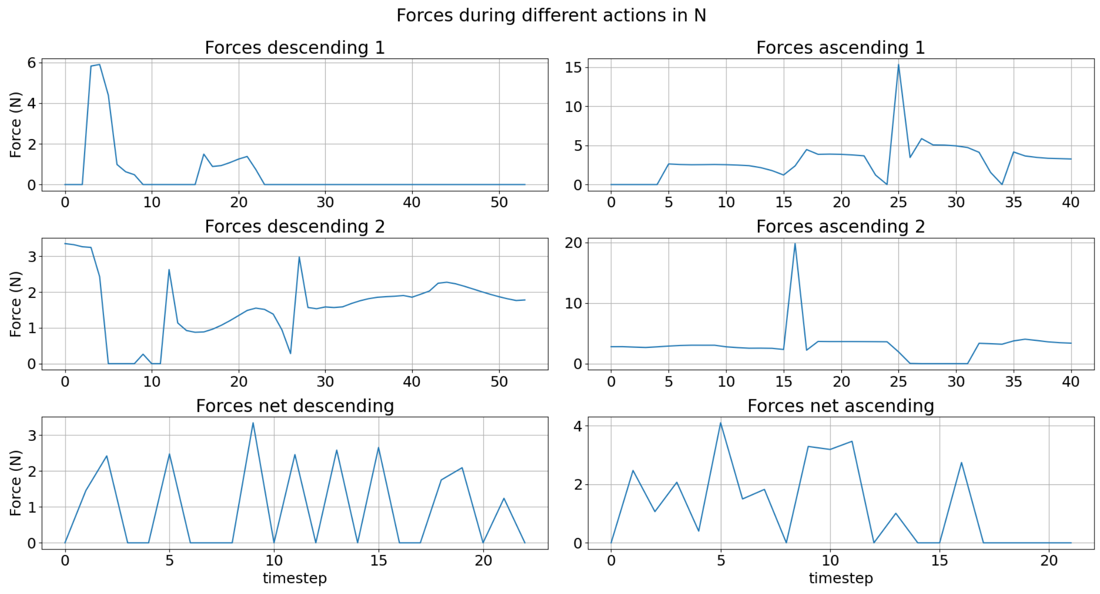

Figure 12: Forces of the DMPs and the Net along the trajectories.

makes the tool have a small change of forces at that point. The same occurs at the second ascending motion but with a higher peak of force, in the descending the tool accompanies the trajectory while at the ascending the tool has more probabilities to collapse with the user's arm.

The following Table 4 compares relevant features, such as mean forces and velocities, maximum forces and the reward, for every motion, for the DMPs and the Net. To be rigorous with the human feelings, the mean force has been calculated by the mean of the points of contact.

Comparing features for different DMP velocities with the net					
Feature	Action	$\tau = 1.5$	$\tau = 2$	$\tau = 3$	Net
Mean force (N)	Descending 1	2.68	2.00	1.86	2.25
	Ascending 1	3.06	3.65	3.32	2.26
	Descending 2	1.95	1.78	1.78	-
	Ascending 2	3.09	3.49	2.52	-
Max force (N)	Descending 1	16.10	5.89	6.12	3.34
	Ascending 1	4.28	15.35	12.93	4.10
	Descending 2	3.72	3.35	4.78	-
	Ascending 2	3.86	19.79	4.24	-
Mean velocity (cm/s)	Descending 1	9.91	7.15	5.23	16.95
	Ascending 1	13.26	9.87	6.42	21.97
	Descending 2	10.34	7.27	5.22	-
	Ascending 2	13.30	9.48	6.10	-
Reward	Descending 1	13.23	13.16	17.92	68.38
	Ascending 1	93.06	107.75	102.38	58.09
	Descending 2	8.40	3.37	3.37	-
	Ascending 2	18.27	7.20	8.10	-
Cumulative Reward	Descending 1	13.23	13.16	17.92	68.38
	Ascending 1	106.29	120.91	120.30	126.47
	Descending 2	114.69	124.28	123.67	-
	Ascending 2	132.96	131.48	131.83	-

Table 4: Relevant features for different DMP velocities and the net.

Since the trained model with Reinforcement Learning performs the same movement repetitively, the comparison with the DMPs is done in just one descending and one ascending movements. From this table the following observations can be done, at first the mean force for the different taus doesn't show significant differences, it can be said that when $\tau = 3$ the mean force is lightly lower. Regarding the maximum forces in the motions, the first ascending presents the biggest number of peaks, which says that that motion needs a closer look to find the reasons of the peaks, for all the motions the maximum force is just punctual, the figures of the forces with different τ can be found at the Appendix C.1. The mean velocity from the DMPs are practically proportional to the demonstration (Net) by the time scaling factor τ. The reward feature can show that the majority of the arm is cleaned during the first motions, and the second motions end taking some additional markers, it can be appreciated that the DMPs clean a lower number of markers at the descending than in the ascending motions, specially at the first ones, watching the Figure 12 in the descending motion there's a lack of contact along most of the trajectory. The cumulative rewards show how the first motion cleans the most of the markers but the overall performance is lower than the net, with the second motion and the markers removed at that motion the resulting reward is over the net, but for a short value.

5.2.3 Discussion

From this experimentation some conclusions can be extracted. The first of them is that the DMPs can reproduce a trajectory respecting the shape of the movement, forces and velocities from a demonstrated motion to clean the right arm of a user, even if this demonstration comes from a trained net to perform this action. Also the DMPs present a more continuous and softer variations of velocities than the net does, this perturbations that are up to 5cm/s can produce some unsafety feelings for the user and should be avoided. In addition, applying contact to the arm punctually isn't enough to perform a good clear action in real conditions.

Besides the respect of the trajectory, velocity and force shapes, the performance of the DMPs show some peaks of forces that are not excessively big but are over the human preferences.

Furthermore, the forces figure show that there are many points of the trajectory where there's no contact between the tool and the user's right arm, this is translated into a bad of optimization of the robot movements, in order to optimize a better cleaning motion, an active compliance algorithm is needed to force a soft contact all the possible time, this way the cleaning will be better.

5.3 Forcing Soft Contact

After being proved that the DMPs follow the same trajectory shape than the demonstration, an extra implementation for the improvement of performance of the DMPs is done. This implementation consists on forcing contact along all the trajectory to ensure the good cleaning of the arm, to do so, during the simulation, the closest points from the tool to the shoulder, the upper-arm and the forearm of the user's right arm are calculated. The algorithm selects the minimum distance of the tool to those three parts of the arm and gets the vector of the minimum point distances between the tool and the closest part of the arm. Then the coordinates of the trajectory of the tool computed with the DMPs are modified an offset of 1cm closer to the user's arm.

5.3.1 Setup

The whole performance has been made with help of the Bed Bathing environment of Assistive Gym, and a script in Python that computes the DMPs, starts the environment and interacts with the robot to perform the desired trajectory, as it has been done at the previous section, the selected robot to perform the tests is the Sawyer given the fact that it is the one that has the best efficiency among the other robots implemented for the trained net.

Again, four movements will be computed to improve the performance of the action and to test if the DMPs are adaptable in different points of the arm, two will be a descending motion and two ascending, starting with the descending motion.

The initial and goal points are the same than in the previous section as well as the approaching to the arm, the inverse kinematics configuration and implementation, the controller for the trajectory and the compliance implementation to avoid excessive large forces.

This experimentation will be done with two different configurations of the right arm of the user, the first one is with the straight arm, the second one is with the arm partially flexed in a natural position. For the second configuration of the arm, the DMPs will be separated in two, in the descending motion the end-effector will move from the shoulder to the elbow, from the elbow to the wrist and vice versa for the ascending motion.

The goal of this section is to test if the DMPs with the compliance control can follow the desired trajectory and have a similar shape than the demonstration and the smoothness of the movements will be checked. Also the comparison between the effectiveness and efficiency of DMPs and the trained nets will be done.

5.3.2 Straight Arm Scenario

After observing that the DMPs can follow the trajectory of the demonstration respecting the shape, forces and velocities, this implementation consists on applying a little offset to that DMPs calculated trajectory, the performance of this implementation is shown in Figure 13.

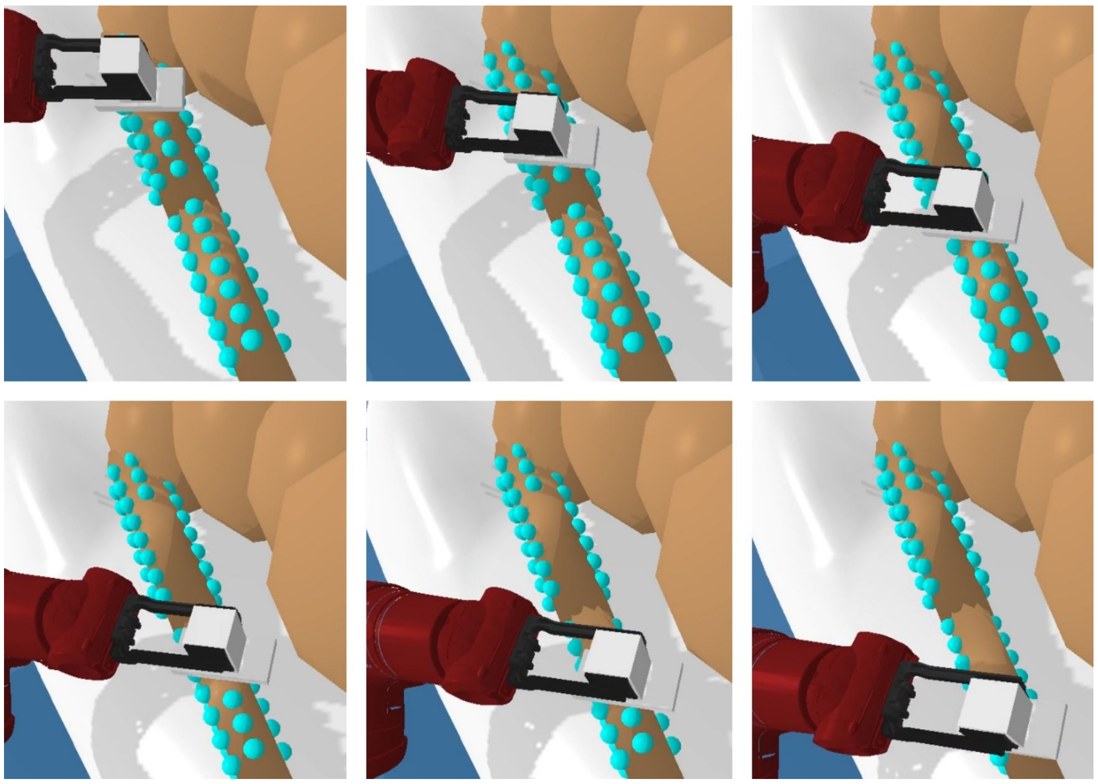
Figure 13: Performance of the first descending DMP in the Straight Arm Scenario, from left to right.

It is important to check if the trajectory still has a similar behavior to the demonstration, to do it, the DMPs with the time scaling factor $\tau = 2$ has been selected. The time scaling factor, as it has been seen at the previous section and in the Appendix C.1, doesn't affect the coordinates of the trajectory, but it does to the velocities. Again, the red and blue lines are for the DMPs and the green one is for the net trained with Reinforcement Learning. The Figure 14 shows how the DMPs still follow a similar behavior to the net, comparing it to the Figure 10 the most significant changes are in the Z coordinate, where the trajectory is lower, hence the robot is moving to force soft contact with the arm. The trajectories for different time scaling factor can be found at the Appendix C.2

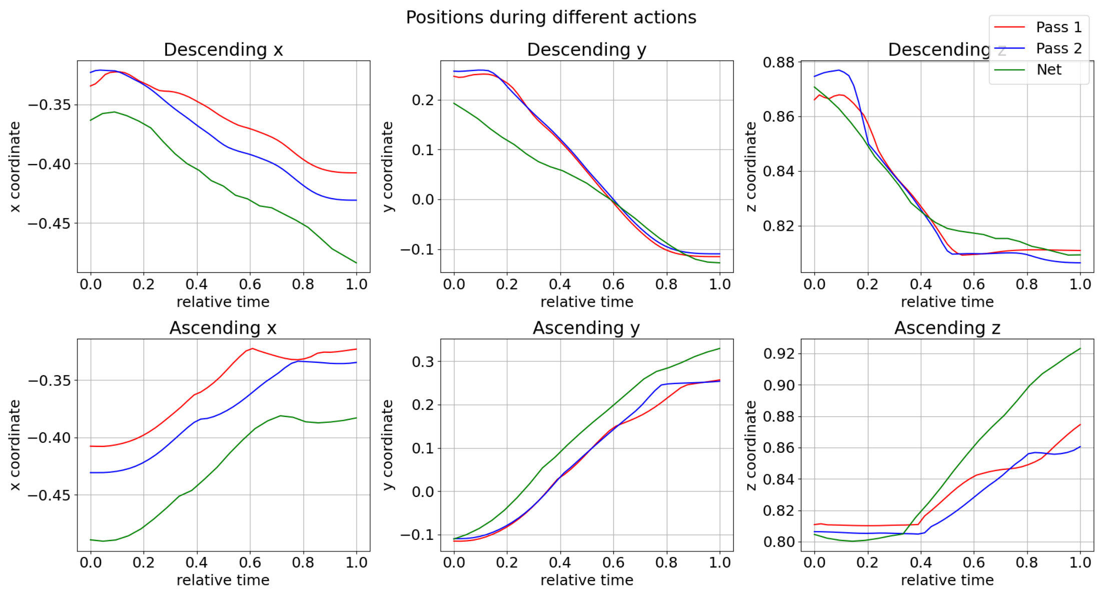

Figure 14: x, y and z coordinates trajectory of the DMPs and the Net for each movement with the forcing contact implementation.

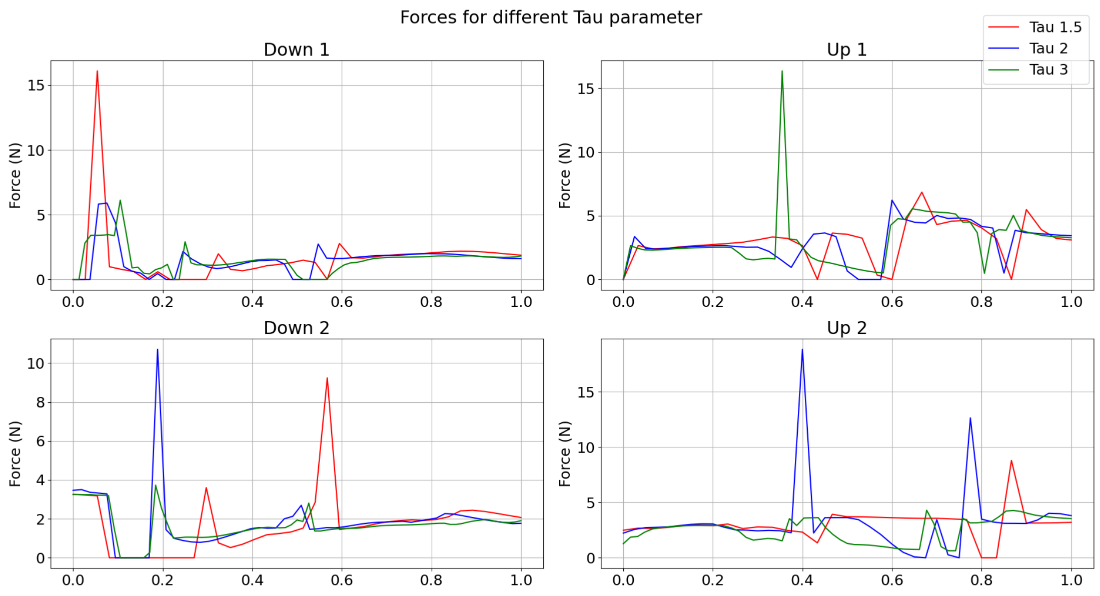

Figure 15: Forces along the trajectories for values of $\tau = \{1.5, 2, 3\}$.

In Figure 15 the forces along the motions are shown, for the values of the time scaling factor $\tau = \{1.5, 2, 3\}$. The differences from the forces shown at the previous section is that the contact created with the active compliance control moves the end-effector of the robot an offset and it creates a low force between the tool and the user's arm. This is the reason to the almost contact force along all the trajectory. In this figure the net force doesn't appear because there is no need to compare the behavior of the shapes, it will be different because the robot always forces the contact. The important features of this Figure are the forces that are different from the implementation without this compliance control, comparing this figure with the Figure 12 it can be appreciated that new forces exist and they aren't excessively high.

Another important feature for this Figure are the peaks or differences of forces. The red lines correspond to the value $\tau = 1.5$, the blue lines to $\tau = 2$ and the green line to $\tau = 3$. Peaks of forces still exist due to the collisions with the elbow, the torso and the shoulder of the human. The first peaks in the descending movements are given by the circumstance that the initial point of that trajectory doesn't make contact with the arm, otherwise the inverse kinematics cannot be calculated because when computing them the collisions are checked for avoiding software problems with the simulation. This handicap makes the algorithm notice that there isn't force applied at that moment and modifies the trajectory with an offset that provokes that impact. Specially with $\tau = 1.5$, as the trajectory for this value is computed in less time-steps, the distance between points is larger, so the robot increases its velocity to reach the first points of the trajectory, when it reaches it, the collision is at a higher velocity where consequentially the collision force will be bigger.

The Figure 16 shows the total velocity of the end-effector for the values of the time scaling factor $\tau = \{1.5, 2, 3\}$ and the net. The red line corresponds to $\tau = 1.5$, the blue line to $\tau = 2$, the green line to $\tau = 3$ and the black line corresponds to the net. The important thing to notice about this figure is the similarity of the shapes of velocity for all the different values of τ and how they follow a similar behavior than the net does. A characteristic feature to observe here is that the net doesn't stop when it reaches the defined goal of the DMPs while the DMPs do it, the net doesn't have goal points to achieve and it is rewarded the maximum markers eliminated in the minimum time steps possible. Another characteristic feature is the sudden decrease of velocity for the first ascending motion at the middle of the trajectory, looking at the forces figure from above it can be seen that at the same time the force increases significantly to about 5N, bigger than the average, this can confirm a soft collision with the human's torso that it is confirmed by checking the simulation.

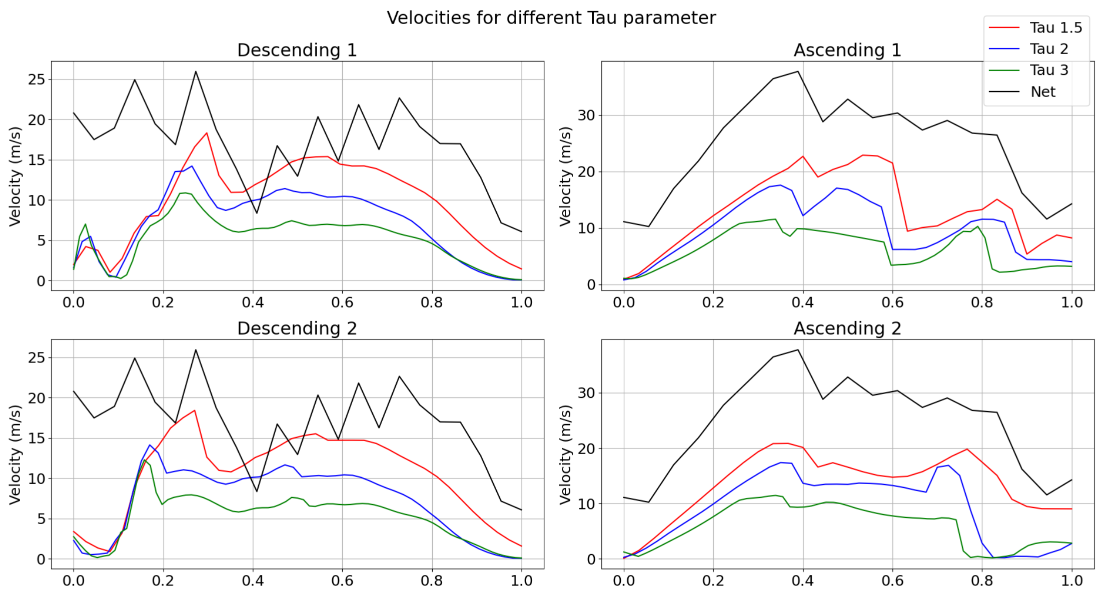

Figure 16: Velocities along the trajectories for values of $\tau = \{1.5, 2, 3\}$ and the net.

The following Table 5 compares characteristic features, such as mean forces and velocities, maximum forces and the reward, for every motion, for the DMPs and the trained net. To be rigorous with the human sensation, the mean force has been calculated by the mean of the points of contact. From this table the following information can be extracted, regarding the mean force, it decreases slightly, for almost all the motions, for the bigger τ value, however, the difference is very low. The maximum forces are corrupted by the collisions with the environment, the significant information from this section are the net maximum force values, which are significantly lower than the DMPs, in exception of the second motion for $\tau = 3$. The mean of the velocities are practically the same than in the performance without this active compliance control, the time scaling factor makes the velocity approximately inverse proportional to the demonstration's by the following equation $DMPvel = DEMOvel/\tau$. The rewards show a significant difference from the previous section without the compliance control, the rewards are more split between the first ascending and descending movements, while at the other most of the reward is at the first ascending motion, this is given by the forcing of the contact, at the first trajectory, where in most of the trajectory the tool didn't contact with the arm. The overall performance, in terms of cumulative reward is practically the same than in the previous section.

Comparing different velocities for DMPs with compliance					
Feature	Action	$\tau = 1.5$	$\tau = 2$	$\tau = 3$	Net
Mean force (N)	Descending 1	2.06	1.81	1.69	2.25
	Ascending 1	3.42	3.22	3.09	2.26
	Descending 2	2.18	2.01	1.76	-
	Ascending 2	3.25	3.39	2.49	-
Max force (N)	Descending 1	16.10	5.89	6.12	3.34
	Ascending 1	6.85	6.22	16.37	4.10
	Descending 2	9.24	10.71	3.73	-
	Ascending 2	8.79	18.83	4.29	-
Mean velocity (cm/s)	Descending 1	9.87	7.17	5.26	16.95
	Ascending 1	13.11	9.69	6.26	21.97
	Descending 2	10.43	7.25	5.17	-
	Ascending 2	13.49	9.11	6.00	-
Reward	Descending 1	53.26	58.28	58.13	68.38
	Ascending 1	52.98	52.85	71.97	58.09
	Descending 2	3.40	2.84	3.32	-
	Ascending 2	23.15	17.22	-1.89	-
Cumulative Reward	Descending 1	53.26	58.28	58.13	68.38
	Ascending 1	106.24	111.13	130.1	126.47
	Descending 2	109.64	113.97	133.42	-
	Ascending 2	132.79	131.19	131.53	-

Table 5: Relevant features for different DMP velocities and the net, with compliance.

5.3.3 Flexed Arm Scenario

The same algorithm applied in the straight arm section is applied with a different configuration of the arm. As it has been explained, the DMPs for this configuration are trained with the same data than in the straight arm but the data is splitted in two parts, the upper arm and the forearm. The trajectory of the first descending movement can be appreciated at Figure 17, and the plot of the trajectories for different time scaling factor values can be found at the Appendix C.3.

In Figure 18 the forces that the robot applies to the human are shown, for different values of the time scaling factors, the red lines correspond to the value $\tau = 1.5$, the blue lines to $\tau = 2$ and the green line to $\tau = 3$. The implementation of the DMPs for this configuration of the arm has already been done with the active compliant control to seek constant contact with low forces. In general all the trajectories accomplish the objective unless two singularities, the fist one is at the first descending movement, where there has been made a punctual force contact about 20N, the reason of this force is that a robot joint touches the lateral side of the arm, the other singularity is the collision of the tool with the oversized and rigid shoulder from the mannequin.

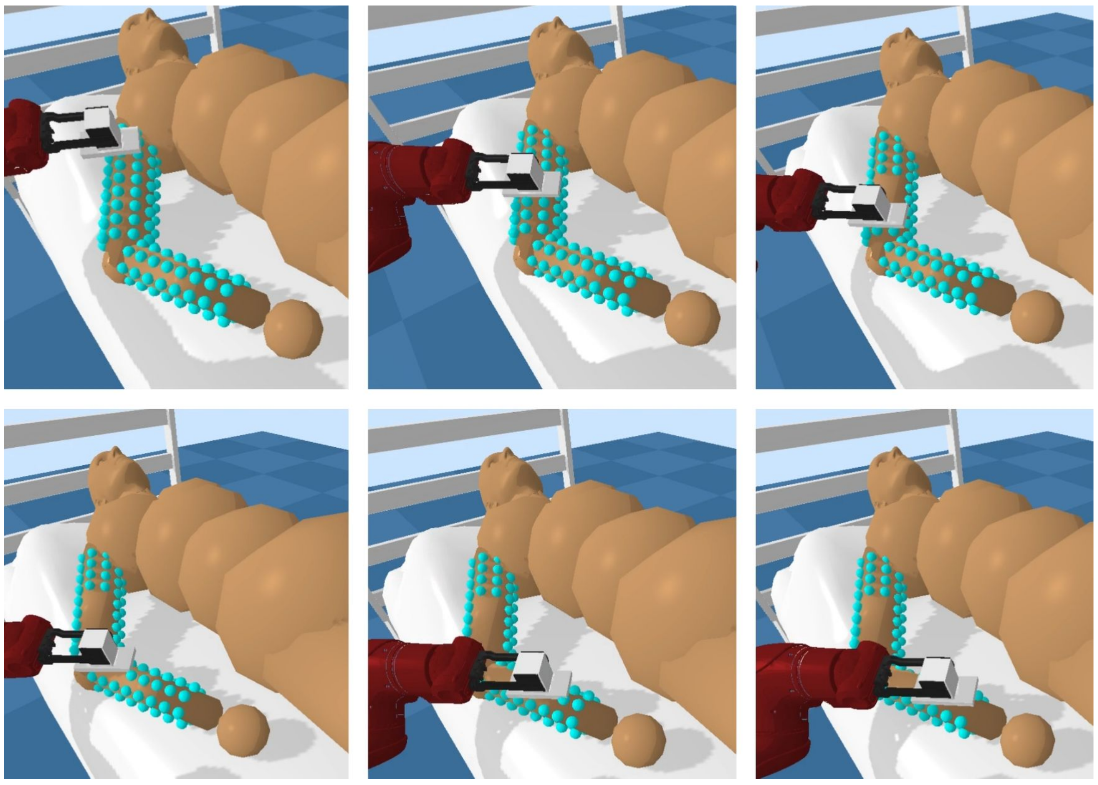
Figure 17: Performance of the first descending DMP in the Flexed Arm Scenario, from left to right.

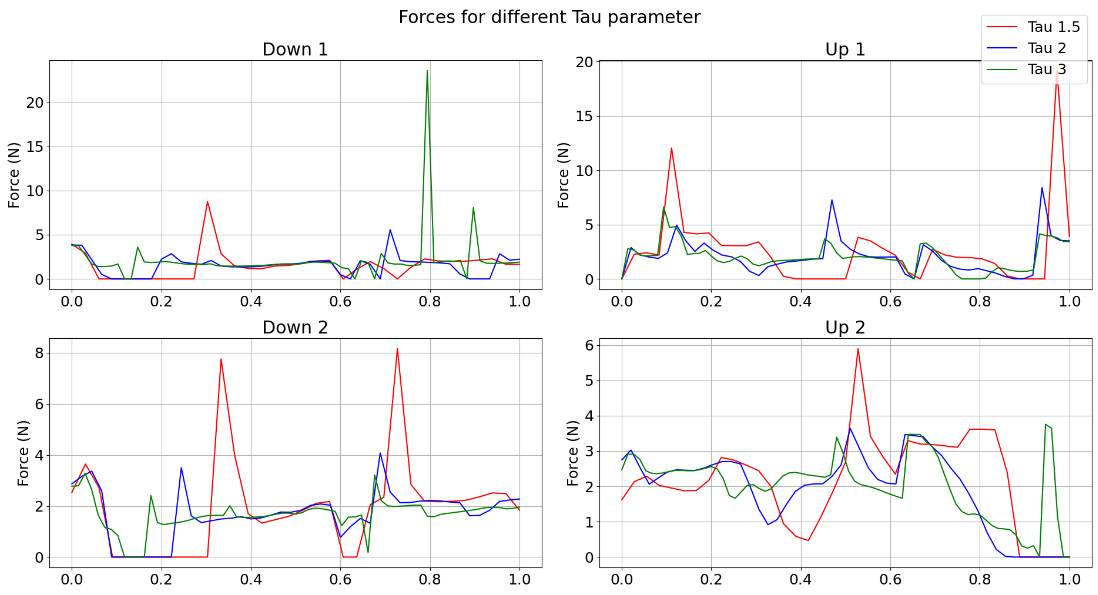

Figure 18: Forces for values of $\tau = \{1.5, 2, 3\}$ in the Flexed Arm Scenario.

The Figure 19 shows the velocities along the trajectory for the different values of the time scaling factors, the red lines correspond to the value $\tau = 1.5$, the blue lines to $\tau = 2$ and the green line to $\tau = 3$, the black line is for the net. The shapes of the trajectories computed with DMPs are different from the demonstration's, the reason is justifiable, the DMPs for this configuration have been computed with the same data than in the Straight Arm Section but splitted in two parts, as it has been explained, doing motions along the upper arm and the forearm, passing through the elbow. The current implementation of the DMPs assume a velocity at the end of the trajectory of 0, so when the tool is close to the elbow the velocity decreases. Making the velocity of the end-effector different from zero at that point would make the shapes of the velocities be more similar. The motion could be implemented by a demonstration of all the trajectory from the shoulder to the wrist but that would be useful for that configuration of the arm, the currently implemented is more universal.

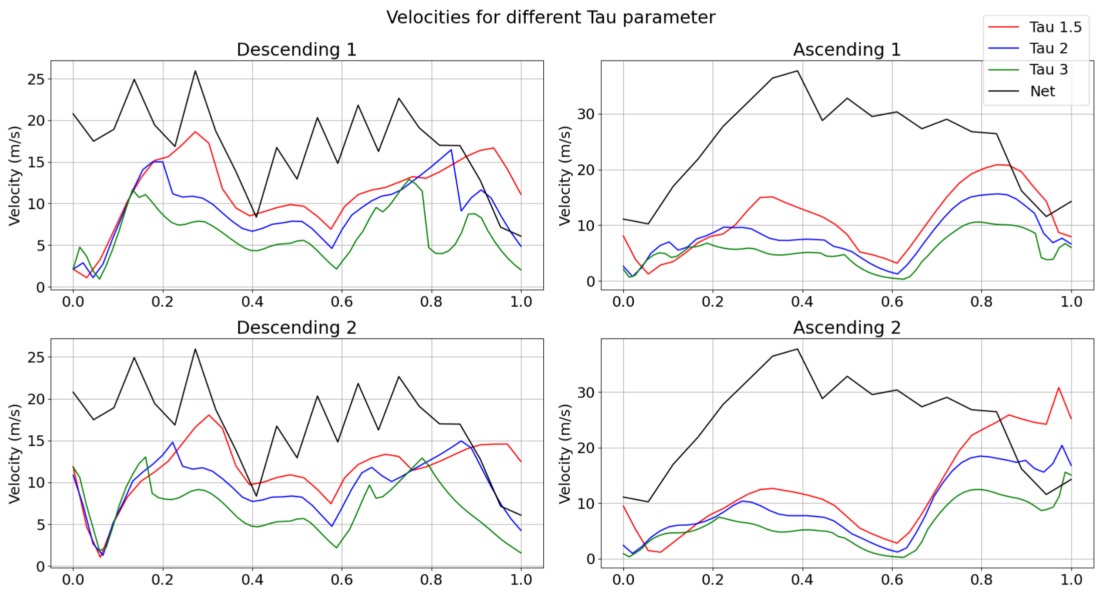

Figure 19: Velocities for values of $\tau = \{1.5, 2, 3\}$ and for the net in the Flexed Arm Scenario.

The following Table 6 compares the characteristic features, such as mean forces and velocities, maximum forces and rewards for every motion, for the DMPs and the trained model. To be rigorous with the human sensations, the mean force has been calculated by the mean of the points of contact.

Comparing different velocities for DMPs with compliance				
Feature	Action	$\tau = 1.5$	$\tau = 2$	$\tau = 3$
Mean force (N)	Descending 1	2.19	2.00	2.40
	Ascending 1	3.53	2.09	2.10
	Descending 2	2.76	2.02	1.77
	Ascending 2	2.72	2.19	2.03
Max force (N)	Descending 1	8.75	5.56	23.56
	Ascending 1	19.17	8.40	6.62
	Descending 2	8.16	4.08	3.28
	Ascending 2	5.89	3.64	3.76
Mean velocity (cm/s)	Descending 1	11.44	9.16	6.33
	Ascending 1	10.68	7.95	5.33
	Descending 2	11.53	9.56	7.01
	Ascending 2	12.44	9.37	6.16
Reward	Descending 1	88.34	113.27	127.93
	Ascending 1	32.91	23.06	7.79
	Descending 2	27.09	-1.75	-1.83
	Ascending 2	-1.92	-2.02	-2.17
Cumulative Reward	Descending 1	88.34	113.27	127.93
	Ascending 1	121.25	136.33	135.72
	Descending 2	148.34	134.58	133.89
	Ascending 2	146.42	132.56	131.72

Table 6: Relevant features for different DMP velocities and the net, with compliance and the Flexed Arm configuration.

Comparing this table with the Table 5 in the previous section it can be seen that the properties of the motions learned by Imitation Learning are similar to the same movement for the straight arm. Even the peaks of maximum forces created by collisions with the torso of the mannequin have disappeared for the current configuration of the arm. Doing the comparison with the trained net for the same task, the mean forces are similar but the reward is higher. This table doesn't include the net performance since it presents some irregularities and splitting the information in descending and ascending motions isn't possible. The overall performance of the trained net for this environment is shown in Table 3.

5.3.4 Discussion

From this experimentation some conclusions can be extracted. The first of them is that the DMPs can reproduce a trajectory respecting the shape of the movement, and velocities from a demonstrated motion to clean the right arm of a user, even if this demonstration comes from a trained net to perform this action and also with the implementation of forcing a soft contact along all the trajectory. From the same demonstration the implementation for the flexed arm can be done by splitting the DMPs by the elbow and the results are practically equal than in the straight arm configuration. With this methodology the robot has reached more surface of cleaning than the trained model, which has shown a worse performance compared to the straight arm with the same training time.

Also the DMPs present a more continuous and softer variations of velocities than the net does. It must be remarked that applying contact to the arm punctually isn't enough to perform a good clear action in real conditions, the reward calculated from the algorithm of the Bed Bathing environment only gives a reward for applying contact over a marker, then the marker disappears, this induces a problem where the net learns to apply instant pressure to the marker and go for the next, avoiding continuous forces, in real life there must be contact along all the trajectory, to ensure the correct cleaning of the user, so the rewards computed from the algorithm doesn't prove the correctly cleaning but proves the percentage of accessed parts of the arm. The introduction of the forcing contact algorithm increases the performance of the cleaning task because there exists contact in practically all the trajectory.

The performance of the DMPs still show some peaks of forces that are not excessively big but are over the human preferences definition, it is given by collisions with the elbow, torso and shoulder due to the rigidity of the mannequin and its oversized torso and shoulder. The collisions with the torso can be avoided by applying a short rotation of the arm in the Z axis.

5.4 Conflicting Zone Avoidance

The last improvement algorithm for the Imitation Learning performance is the one that, in case that the patient has some injury in a concrete zone of the arm, the mission of the robot will be to perform the same cleaning task but avoiding that zone during the trajectory of the movement. The avoidance will be an offset of the trajectory in the Z coordinate of ten centimeters over the conflicting zone, that zone is a circular zone of ten centimeters of radius around an specific point.

5.4.1 Setup

The implementation of this algorithm is in the same conditions than explained in the Section 5.3.1. It is programmed inside the same code the change of trajectory in case the robot is approaching to the conflicting zone, this means that the algorithm searches for the soft contact unless the end-effector is over the conflicting zone. Like in the previous experiment, four movements, two ascending and two descending, will be executed in the Straight Arm Scenario and in the Flexed Arm Scenario.

The objective of this experiment is to test the capability of the algorithm and the robot to perform the task with the desired specifications. The time scaling factor selected is $\tau = 2$, and the conflicting zone is the elbow.

5.4.2 Straight Arm Scenario

In Figure 20 it is shown the execution of the Bed Bathing task avoiding a zone around the elbow for the Straight Arm Scenario. The displayed order in the figure is from the left to the right and from the top to the bottom.

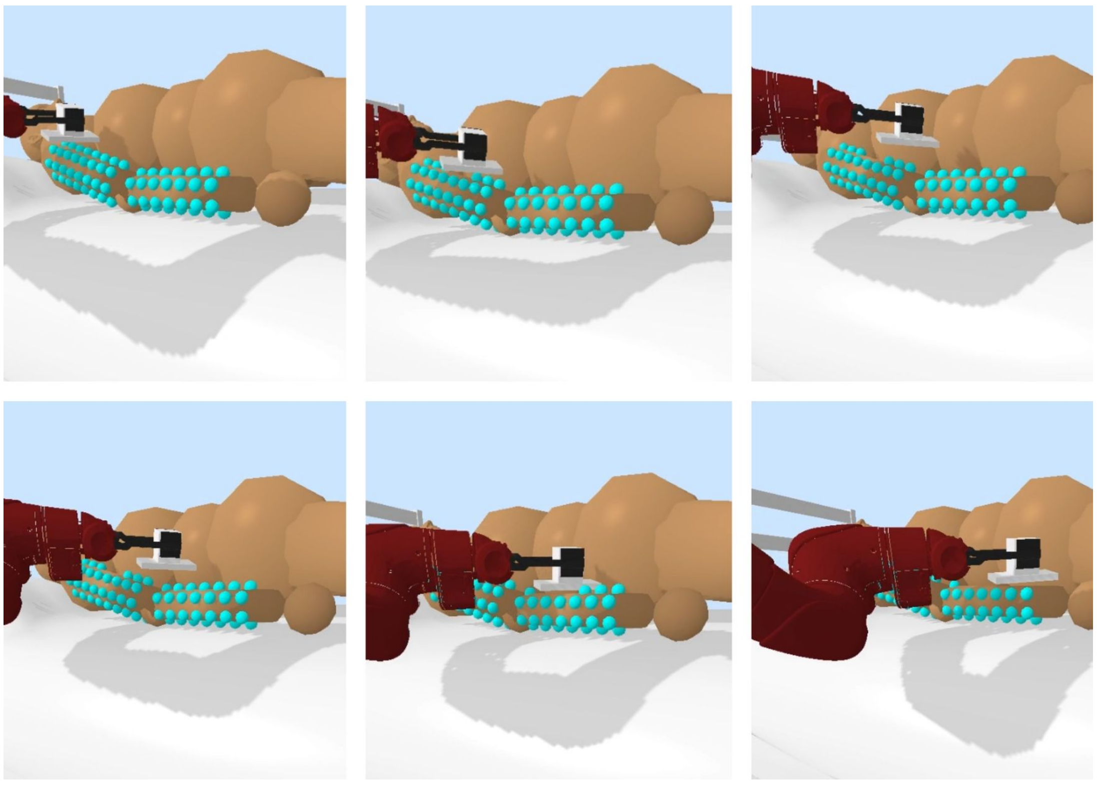

Figure 20: Performance of the first descending DMP in the Straight Arm Scenario avoiding contact over the elbow.

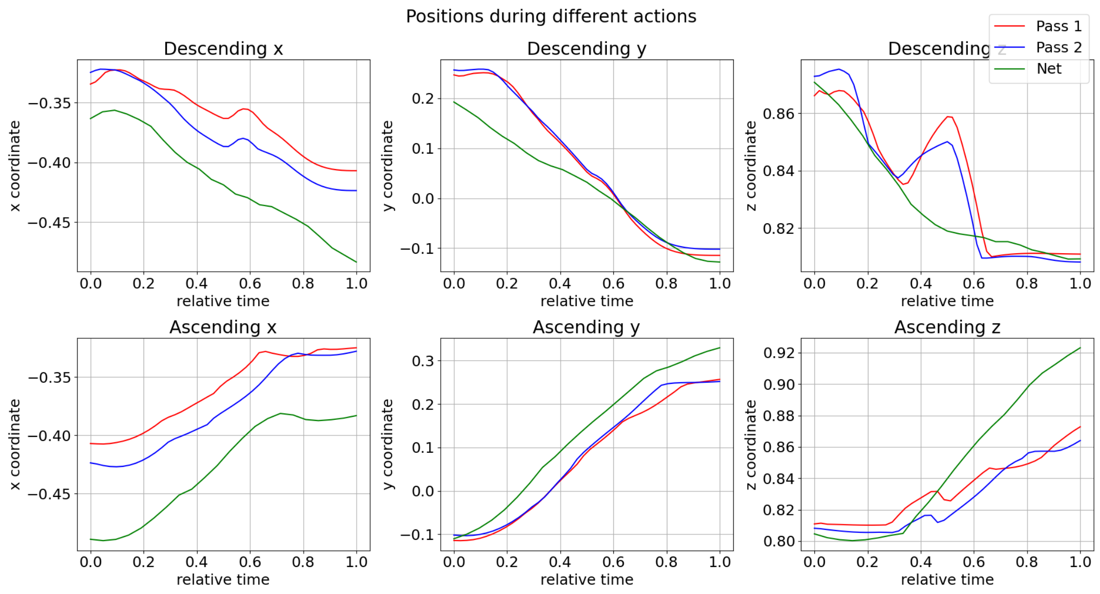

Figure 21: Trajectory coordinates for the DMPs and the net for the different motions for the straight arm in the Avoiding Zone algorithm.

The behavior of the trajectory can be checked in Figure 21, the red and blue lines are for the first and second motion DMPs respectively, and the green line is the net trained model. The trajectory from the DMPs still behaves, as expected, like the demonstration, excepting for the avoidance of the conflicting zone, that it mostly affects the z coordinate, but it does to the x and y lightly.

As the behavior for the DMPs is the same for different time scaling factor, excepting the velocity that is proportional, the study of the forces has been computed only for $\tau = 2$. The important characteristic of the Figure 22 is the amount of time steps around the elbow where the force is zero, as it should be. This proves that the algorithm is well implemented and that the robot is capable to perform the task avoiding touching a concrete zone. When the tool recovers the contact with the arm after avoiding the conflicting zone, it doesn't do it progressively, but the resulting forces are low enough to provide safety to the user.

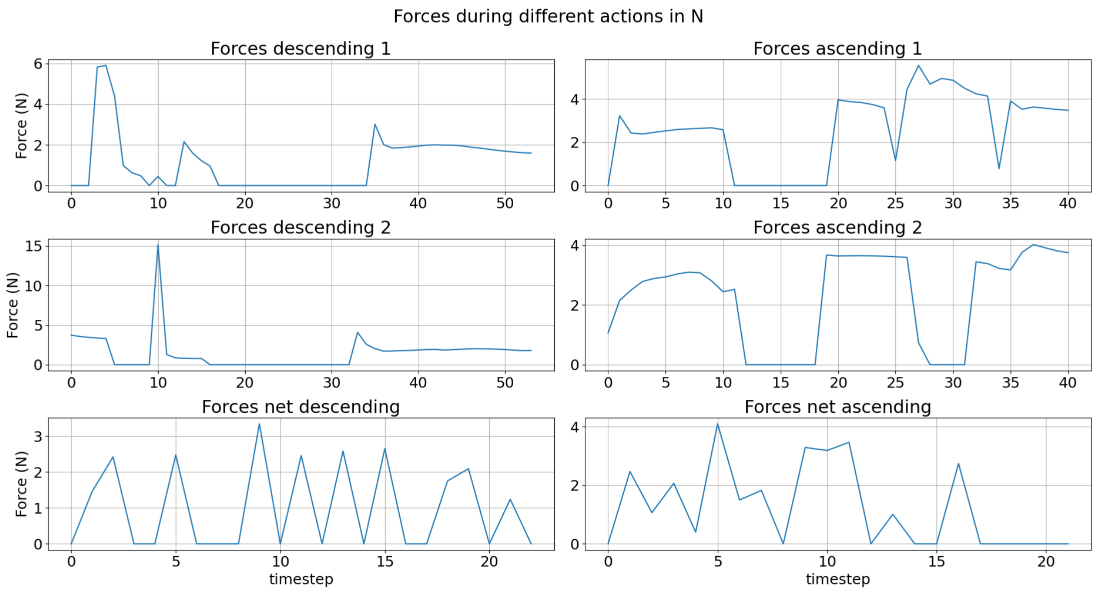

Figure 22: Forces of the DMPs and the Net along the trajectories in Avoiding Zone scenario for the Straight Arm.

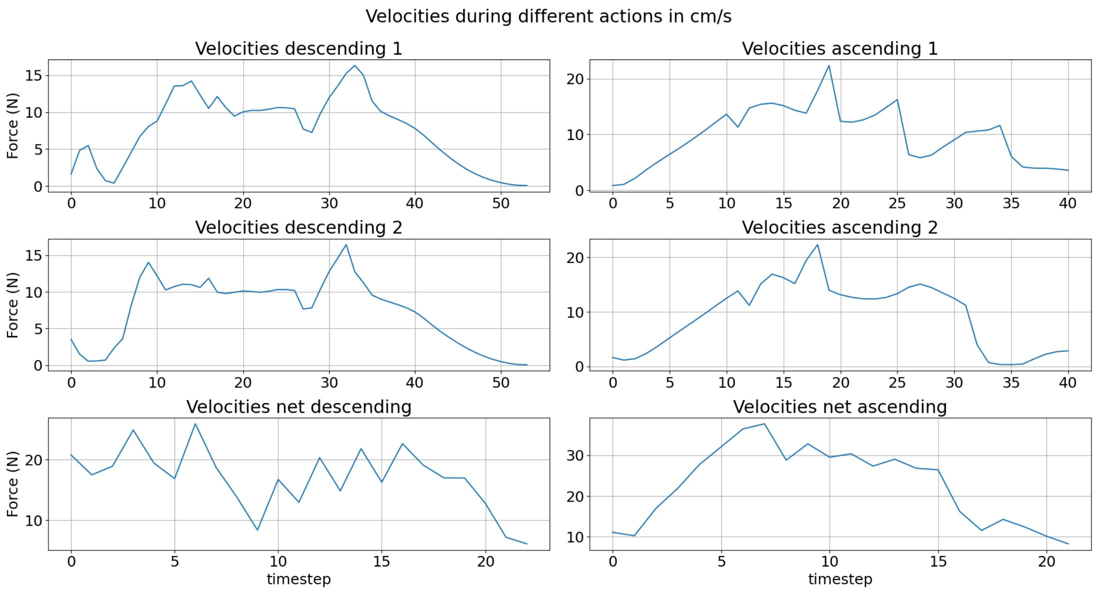

Figure 23: Velocities of the DMPs and the Net along the trajectories in Avoiding Zone scenario for the Straight Arm.

The velocities of the trajectory are shown in Figure 23, they show a similar behavior than the net does, excepting for the zones where there is no contact force and the trajectory adapts its positions and velocities. In this Figure it is appreciable that when velocities are relatively high when there is no contact, when touching the tool with the arm it creates a peak of force.

5.4.3 Flexed Arm Scenario

In Figure 24 the execution of the Bed Bathing task avoiding a zone around the elbow for the Flexed Arm Scenario can be seen, the performance is done from the left to the right and from the top to the bottom of the Figure.

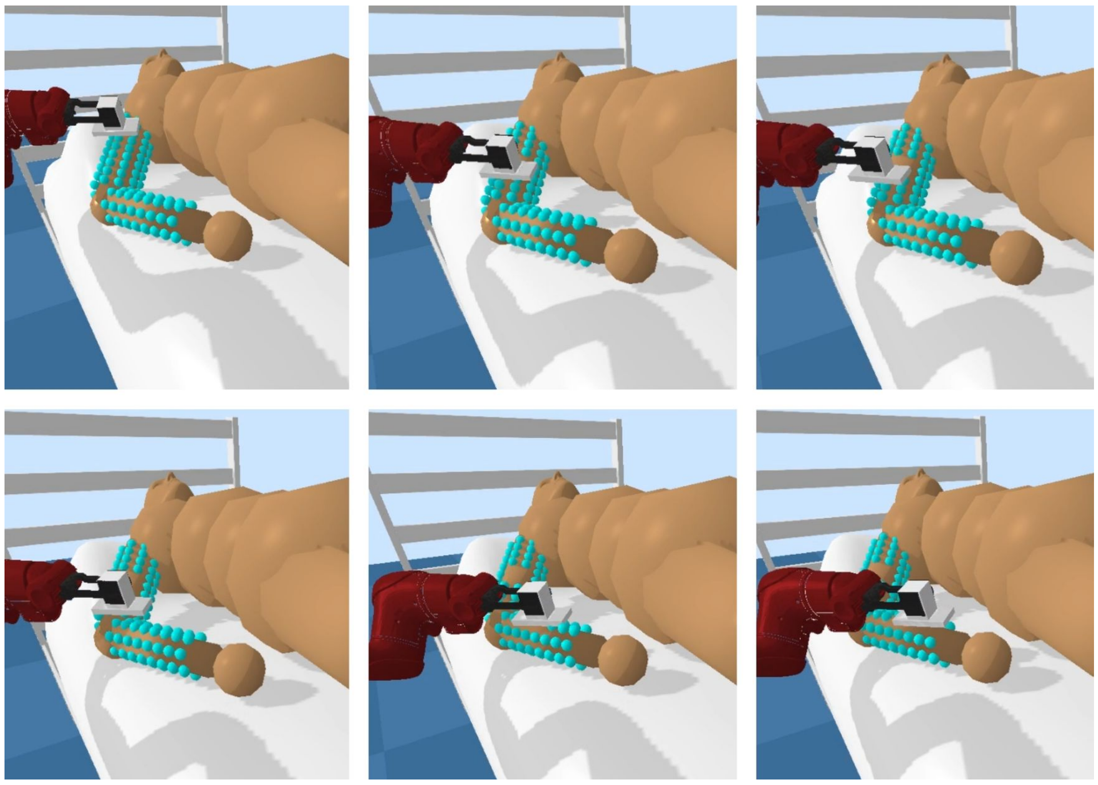

Figure 24: Performance of the first descending DMP in the Flexed Arm Scenario avoiding contact over the elbow, from left to right.

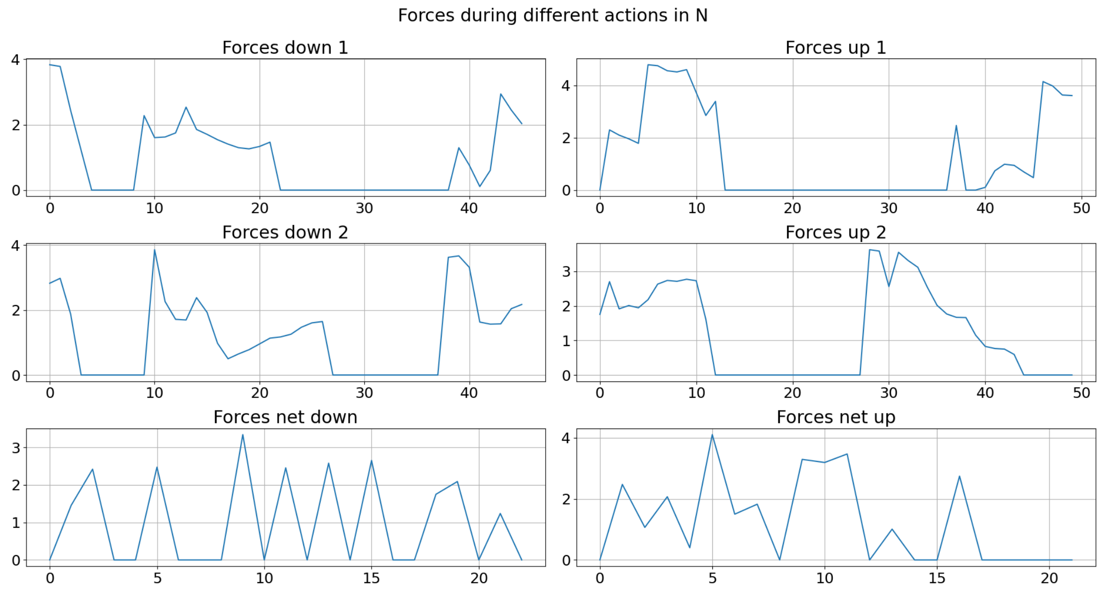

Figure 25: Forces of the DMPs and the Net along the trajectories in Avoiding Zone scenario for the Flexed Arm.

From Figure 25 some facts can be extracted, the first one is that the force of the DMPs show a similar behavior than the Nets by looking at the peaks without considering the elbow proximity. The second

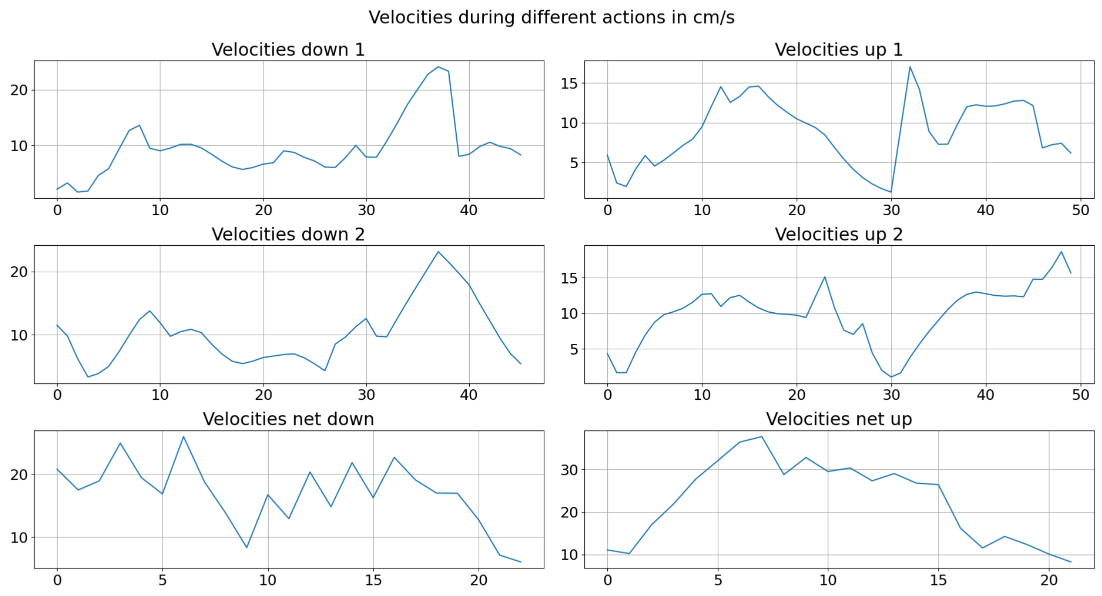

Figure 26: Velocities of the DMPs and the Net along the trajectories in Avoiding Zone scenario for the Flexed Arm.

one is that the robot successfully performs the task, avoiding the elbow proximities and keeping as much contact as possible without exceeding forces. No peaks of an excessive force appear. The difference of time steps from the first motion to the second motion occurs for the points of the trajectory, the first motion follows the arm at the top of it while the second does the same but a bit displaced to an outer side of the arm and a bit lower, as it has been shown in Figure 9. Having the arm flexed makes the first motion reach the conflicting zone, which is a circle of five centimeters around the elbow, earlier.

As it has been explained in Section 5.3.3, the velocity for the Flexed Arm configuration diminishes when approaching the elbow, although taking this into account the shapes of the DMPs velocities and the net are similar.

5.4.4 Discussion

This experimentation demonstrates the capacity of the algorithm, for both scenarios, to generate suitable motions to avoid a conflicting zone using Imitation Learning while the active compliance control is activated. The performances maintain the demonstration's trajectory shape, in terms of velocities and positions, even with the compliance control and the avoidance of a conflicting zone implemented. The recovery of the contact of the tool with the arm must be done progressively to avoid harming or scaring the user.

5.5 Lessons Learned

In this section the suitability of each algorithm is analyzed from the experimental results of the simulations. The proposed algorithms are evaluated for different situations and strengthens and drawbacks to implement each method are commented. Approaches are catalogued as easy, medium and difficult to implement or if it is needed to retrain the model. The following Table 7 shows the knowledge acquired for different situations.

Situation	RL approach	IL approach
Adaptability to user preferences	Easy	Easy
Adaptability to user handicaps	Easy	Difficult
Changing the scenario	Easy	Medium
Complex movements	Difficult	Medium
Control of the trajectory	Difficult	Easy
Expression detection	Difficult	Medium
Generalization for different robots	Requires Retraining	Easy
Low learning time	Difficult	Easy
Modification of the performance	Requires Retraining	Easy

Table 7: Cataloguing the difficulty of different situations for each approach.

Concerning to the adaptability of the user preferences, it exists the possibility that the user is not comfortable with the current performance and it prefers a different velocity or contact forces during the trajectory. When using Reinforcement Learning algorithm, the model has to be retrained with different human preferences parameters. If the Imitation Learning approach is used, the needed change is to modify the τ parameter and the limit force parameter. The Reinforcement learning algorithm will take longer to be able to perform the action as it needs more time for the training.

If the training of the net with Reinforcement Learning includes user handicaps, such as tremors or joint limits, the robot will have learned the best actions, in every observation, taking that into account to perform the action. In the Imitation Learning approach an estimator of the movement of the human needs to be introduced and that adds difficulty in the programming.

The Assistive Gym provides different scenarios and a tutorial to create and train new policies and scenarios. It doesn't require much programming skills to create a new environment and its reward, since the machine learning algorithm is already implemented, with Reinforcement Learning the only needed change is to create the new environment. However, in IL depending on the complexity of the action to perform, more knowledge of the motion is required to perform and adapt the DMPs trajectory to make them safe.

With Reinforcement Learning, the robot has achieved worse results in scenarios where the robot has to perform complex movements to complete the action successfully. It is easier to implement in Imitation Learning, since the procedure is the same than in simple movements but needs more supervision for safety.

Learning the actions with Imitation Learning requires demonstrations. These demonstrations are done by the programmer or the supervisor of the robot and the trajectory that the robot will do is manually controlled. On the contrary, in Reinforcement Learning the net explores to learn the best performance, but the result cannot be controlled as the robot performs the best movements learned.

As the results of the Sections 5.3 and 5.4 have shown, changes in the trajectory are easy to implement in the DMPs trajectory created with Imitation Learning, while the model trained with Reinforcement Learning has to be retrained to implement features such as avoiding a conflicting zone.

As every robot is different and the net uses characteristic points of the robot for the training, it is needed to retrain the model for new robots in the Reinforcement Learning algorithm. Meanwhile in Imitation Learning, having the DMPs computed in Cartesian space, the trajectory is learned and it is adaptable for all the robots, the only requirement is that the robot can reach all points of the trajectory with natural joint configurations.

Introducing expression recognition is a hard task but it has been already studied and implemented, with help in programming and studying articles about it, the recognition can be applied easily in the Imitation Learning algorithm, checking the expression of the user and reacting to it. On the contrary, it is more difficult to introduce feelings or expressions and physical reaction movements of the user in case of fright in the simulations where the model is trained with Reinforcement Learning. The training can be done in the real world with a human, but it is unsafe, expensive and time consuming.

The trained net with Reinforcement Learning took 60h to perform the task correctly, and it even performed the task worse in a complex configuration of the arm. Meanwhile in Imitation Learning, with the data extraction and introduction, it can be done in a few hours.

6 Project Impact

6.1 Social Impact

Assistive robotics is a fast growing field and so does the controversy around opinions of this paradigm.

On the one hand, the research on assistive robots is focused on complementing the job of health care workers in hospitals, rehabilitation centers and nursery homes, as well as empowering dependant people by giving autonomy to the people with a dependency in ADL, this way the feeling of dependency from other people will decrease. The implementation of assistive robots does that healthcarers can focus their energy on patients who need it more. The actual social and sanitary context with the apparition of the global pandemic virus SARS-CoV-2 that provokes the COVID-19, has shown that there can surge situations where the sanitary capacity gets overtaken. This situations reflex the possible need of assistive robots to improve everyone's safety and the healthcarers's mental health.

On the other hand, the assistive robots are integrated to areas where robots had never entered before, such as decision-making, feeling and relationships, the regularization of this areas for the public benefit has lead to an establishment of a new discipline: Roboethics [24], and it has created a fast growing and wide field of study that is currently taking course.

The change from a human to a robot for performing this task represents a psychological impact for the patient by the lack of socialization during it. This is specially critical in case of vulnerable groups, such as elder, children or mentally disabled people.

There are many issues to be considered before implementing robot assistance in eldercare, they lose opportunities to socialize. Furthermore they could be more neglected by the society and their families. There is risk of objectification in case that the robot decides to move people, or some part of its body, without consulting. The feeling of losing privacy by having a robot controlling them. The restriction of personal liberty. The infantilization coming from encourage them to interact with a robot as if it was a companion, and finally the attribution of responsibility if things went wrong.

Even if assistive robots can provide safety, success and adaptability to every user for the tasks, there will be needed a regularization of the field together with an increment of confidence from the society.

6.2 Budget

The cost of the project has been separated in two parts. The first one is the depreciation and energy consumption related with the computer that has been used, C_C, the second one is the human resources cost of the student and the two supervisors, C_{HR},

$$Total\ cost = C_C + C_{HR}$$

The software used for the simulations, Assistive Gym, Python and ROS are open source so their cost is zero.

The project has been developed during 6 months, working approximately 35 hour per week, resulting in a total of 840 hours. The computer has been running constantly for all that time. Considering using the computer eight ours a day, five days a week and forty-eight weeks a year, during five years, results in a useful life of 9.600 hours. From the price of the computer, which is 1.500€, and its life expectancy the price of the depreciation per hour used can be calculated, this price is 0,15625€/h. The price of the depreciation of the computer for this project is 131.25€.

For the human resources cost, a base salary of 15€/h for the junior engineer is adjudicated, and 30€/h for the supervisors. The time spent in the project for the student is 840 hours, this supposes a total cost of 12.600€. The supervisors have dedicated approximately 3 hours a week during the same period of time, what gives a total of 72 hours for both, the total cost for each supervisor is 2.160 €, and the total for both is 4.320€.

Finally, since the project has been done telematically, the electrical cost can be calculated considering the energy consumption of the computer during its working time. From the battery it can be extracted that the potency the laptop uses is of 0,065kW, meaning that the energy consumed is 54.6kWh. The price of the energy consumed is 0,104778 Eur/kWh, the total electrical cost is 5.72€.

The total cost of the project considering all the parts is of 17.056,97€. Which can be approximated to 17.200€ to cover factors that might not have been considered.

In Table 8 the total cost of the project is shown:

Cost factor	**Variable cost (€/h)**	**Time (h)**	**Total cost (€)**
Computer depreciation	0,15625	840	131,25
Computer energy consumption	0,00681	840	5,72
Project managers ($\times$2)	30	72($\times$2)	4,320
Engineer	15	840	12,600
TOTAL COST			17.056,97

Table 8: Project cost.

In the case of performing the experiments in the real-world, the cost of the robot selected will have to be considered for the amount of hours dedicated to the project.

7 Conclusions

The results of the experimentation prove that both methodologies are capable of performing a Bed Bathing task motion successfully, and as expected, they differ from each other.

On the one hand, Reinforcement Learning algorithms have high adaptability to user preferences, and to train new models for the same action the only needed change to do is changing the value of the weights. It is also easy to take into account and adapt to possible diseases of the users, like head and arm tremors, joint weakness or limited range of motions. Co-optimization still has to be further investigated but has promising results. On the other hand, generating the trajectories by training nets with Reinforcement Learning creates a dependency of the computational power for a faster training. The fact that training in simulation is faster, cheaper and safer than doing it with real robots and users, but if the user isn't comfortable with the real robot performance trained with Reinforcement Learning it will have to be retrained with different parameters of the reward, which means losing time. Every user is different, or even the same user can change physically or mentally and this implies the need of retraining the net. It is also highly difficult to introduce feelings of the user during the training simulations to make the robot more agreeable.

On the other hand, Imitation Learning is a very fast method to learn an action, and the velocity and trajectory coordinates are easily and fast modifiable. Once the robot has learned to perform a movement, it doesn't have to be trained anymore and can be adapted relatively fast to another user by modifying the necessary parameters to warrant the user's welfare. It also ensures the shape of the motion that wants to be done for a task. This algorithm is generalized in Cartesian coordinate system, what means that other robots can perform the learned action. The data collected can be easily used to perform similar actions: for example, we shown that the robot has been capable to clean a flexed arm using a demonstration of cleaning a straight arm just by splitting the data. Moreover, the implementation of compliance control algorithms and controlling corporal and facial expressions of the user is easier than with Reinforcement Learning, it can also be more prepared for unexpected situations. Another benefit is the possibility of the online actualization of the execution of the task in case the user moves the arm. The drawbacks of Imitation Learning are that to reprogram new explicit situations need an expert to change the code and it has a worse adaptability for a single concrete user than Reinforcement Learning does, also the adaptability to the diseases like tremors adds difficulty to the algorithm. In Imitation Learning the logic of the task has to be implemented with the demonstration while in Reinforcement Learning the logic is learned.

Once the policy trained with Reinforcement Learning has been learned, the robot has performed the movements from the policy and it has been possible to adapt that movements with DMPs for different velocities and small modifications of the trajectory.

In conclusion, Reinforcement Learning algorithms produce robot behaviors that are better suited for users that have a long term dependency, such as particular users or people from the hospital who will spend a long period there. And the most suitable situations for Imitation Learning would be for short term users that will present different symptoms and preferences, such as clinics or hospitals, also places where the need of the robot is urgent and there is not enough time to train with Reinforcement Learning, or particular users with unstable preferences or needs, for example someone who has fragile bones and has injured some part that must be avoided.

8 Future Work

Due to the current social-sanitary situation of the world, it couldn't be checked if the methodology of virtual training of the robots and then transferring and adapting the results to real robots is more efficient and effective than training directly with real robots. This is an important fact that must be tested because it would have a great impact in the assistive robotics field. The performance of both implementations should be tested with real robots and users to verify if the hypotheses from this project is true in real scenarios. The next step would be to test it at the Perception and Manipulation Lab in the *Institut de Robòtica i Informàtia Industrial* with the TIAGo robot.

Both algorithms can be exponentially improved and tested, taking into account that there are more methodologies that are currently researched, they practically have no limits. An interesting test to perform would be to train a policy where in every iteration the human preferences would change, this would implicitly train the mode for different kinds of users and this would provide more autonomy and efficiency to the Reinforcement Learning methodology. Other small changes can be ones like creating a limit speed control to avoid high collision forces, provide a progressive force when the tool touches the arm, checking for possible collisions with the torso or the shoulder and avoid them, increasing the range of cleaning doing it also on a leg, or the whole body.

Currently the robots are static and their base do not move in the simulation. Giving them mobility (a.e. using the mobile base in the Tiago robot) to improve the results and to be able to reach more difficult parts of the body would be interesting. It could also do friendly motions for children and having different interfaces. And also it would be interesting to compare both methods in a much more complex scenarios, where for example, the user is moving, and showing its feelings.

There are lots of scenarios that need to be checked to know which of the methodologies studied in this project suits better for it, like environments where the task consists on very complex movements such as lacing the shoes of a user, or fasten a belt. Another possible future job to do could be testing the viability, in terms of efficiency and effectiveness, once the policy trained with Reinforcement Learning is learned, that the robot learns from simulation demonstrations the motion of that task, improves it with supervision and adapts it for every user with DMPs implementing all the current safe and efficient methodologies to perform assistive tasks.

Acknowledgements

I would to give my biggest gratitude to both my tutors, for finding time to support me and the project, given the difficulties that can have occurred due to the current social and sanitary situation that we are living through.

References

[1] G. Canal, E. Pignat, G. Alenyà, S. Calinon, and C. Torras. Joining high-level symbolic planning with low-level motion primitives in adaptive hri: Application to dressing assistance. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3273–3278, 2018.

[2] Gerard Canal, Guillem Alenyà, and Carme Torras. Personalization framework for adaptive robotic feeding assistance. In *Social Robotics - 8th International Conference, ICSR 2016, Proceedings*, volume 9979, pages 22–31, 11 2016.

[3] A. Clegg. *Modeling human and robot behavior during dressing tasks.* PhD thesis, The Georgia Institute of Technology, 2019.

[4] Z. Erickson, A. Clegg, W. Yu, G. Turk, C. K. Liu, and C. C. Kemp. What does the person feel? learning to infer applied forces during robot-assisted dressing. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6058–6065, 2017.

[5] Z. Erickson, H. M. Clever, V. Gangaram, G. Turk, C. K. Liu, and C. C. Kemp. Multidimensional capacitive sensing for robot-assisted dressing and bathing. In *2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR)*, pages 224–231, 2019.

[6] Zackory Erickson, Henry Clever, Greg Turk, C. Liu, and Charles Kemp. Deep haptic model predictive control for robot-assisted dressing. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, 05 2018.

[7] Zackory Erickson, Vamsee Gangaram, Ariel Kapusta, C. Liu, and Charles Kemp. Assistive gym: A physics simulation framework for assistive robotics, 10 2019.

[8] David Feil-Seifer and Maja Matarić. Defining socially assistive robotics. In *Proceedings of the IEEE 9th International Conference on Rehabilitation Robotics*, volume 2005, pages 465 – 468, 07 2005.

[9] D. Gallenberger, T. Bhattacharjee, Y. Kim, and S. S. Srinivasa. Transfer depends on acquisition: Analyzing manipulation strategies for robotic feeding. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 267–276, 2019.

[10] P. M. Grice, A. Lee, H. Evans, and C. C. Kemp. The wouse: A wearable wince detector to stop assistive robots. In *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, pages 165–172, 2012.

[11] Wu C. Yue Y. et al. Gu, S. Real-time compliance control of an assistive joint using qnx operating system. *International Journal of Automation and Computing*, 10:506–514, 12 2013.

[12] Steven Klee, Beatriz Ferreira, Rui Silva, Joao Costeira, Francisco Melo, and Manuela Veloso. Personalized assistance for dressing users. In *Social Robotics - 7th International Conference, ICSR 2015, Proceedings*, pages 359–369, 10 2015.

[13] Yash Mehta, Rohit Khot, Rakesh Patibanda, and Florian Mueller. Arm-a-dine: Towards understanding the design of playful embodied eating experiences. In *CHI PLAY '18: Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play*, pages 299–313, 10 2018.

[14] D. Park, H. Kim, Y. Hoshi, Z. Erickson, A. Kapusta, and C. C. Kemp. A multimodal execution monitor with anomaly classification for robot-assisted feeding. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5406–5413, 2017.

[15] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, pages 763–768, 2009.

[16] Emmanuel Pignat and Sylvain Calinon. Learning adaptive dressing assistance from human demonstration. *Robotics and Autonomous Systems*, 93, 04 2017.

[17] T. Rhodes and M. Veloso. Robot-driven trajectory improvement for feeding tasks. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2991–2996, 2018.

[18] Niekum S. DMPs. `http://wiki.ros.org/dmp`, 2015 (accessed April 15, 2020).

[19] A.S. Sadun, Jamaludin Jalani, and J.A. Sukor. An overview of active compliance control for a robotic hand. *ARPN Journal of Engineering and Applied Sciences*, 11:11872–11876, 01 2016.

[20] S. Schröer, I. Killmann, B. Frank, M. Völker, L. Fiederer, T. Ball, and W. Burgard. An autonomous robotic assistant for drinking. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6482–6487, 2015.

[21] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 07 2017.

[22] T. Tamei, T. Matsubara, A. Rai, and T. Shibata. Reinforcement learning of clothing assistance with a dual-arm robot. In *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pages 733–738, 2011.

[23] A. R. Tilley. *The measure of man and woman: human factors in design.* John Wiley & Sons, 2002.

[24] Carme Torras. Assistive robotics: research challenges and ethics education initiatives. *Dilemata*, 30:63–77, 2019.

[25] W. Yu, A. Kapusta, J. Tan, C. C. Kemp, G. Turk, and C. K. Liu. Haptic simulation for robot-assisted dressing. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6044–6051, 2017.

A Co-optimization. Additional Information

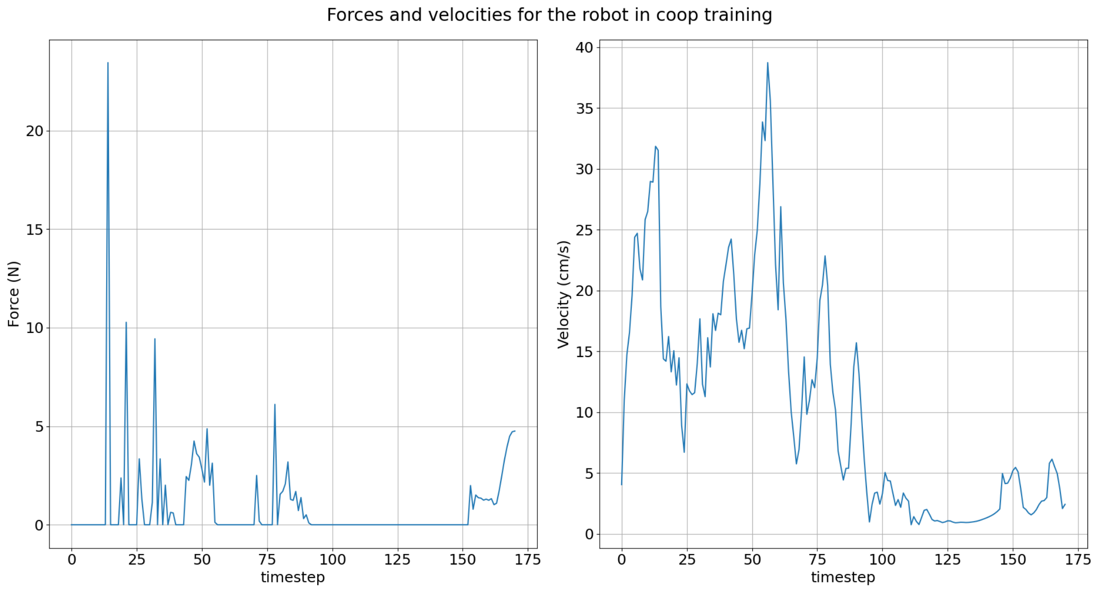

Figure 27: Forces and velocities of the end-effector of the robot for the co-optimized trained net.

B Reinforcement Learning Results for Flexed Arm Scenario

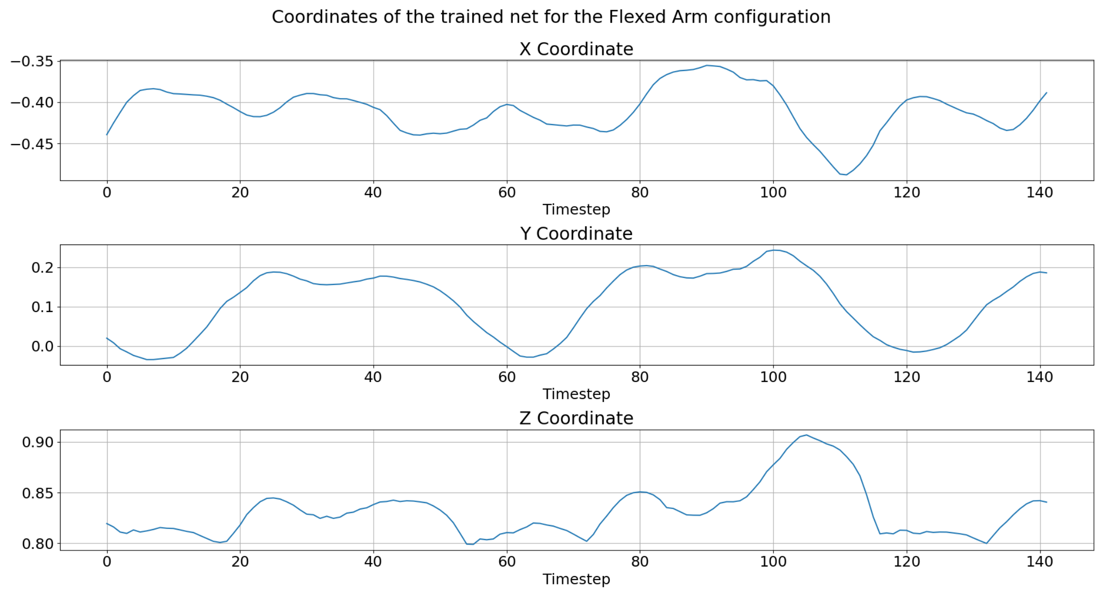

Figure 28: Evolution of the x, y and z coordinates of the trained net with the arm flexed.

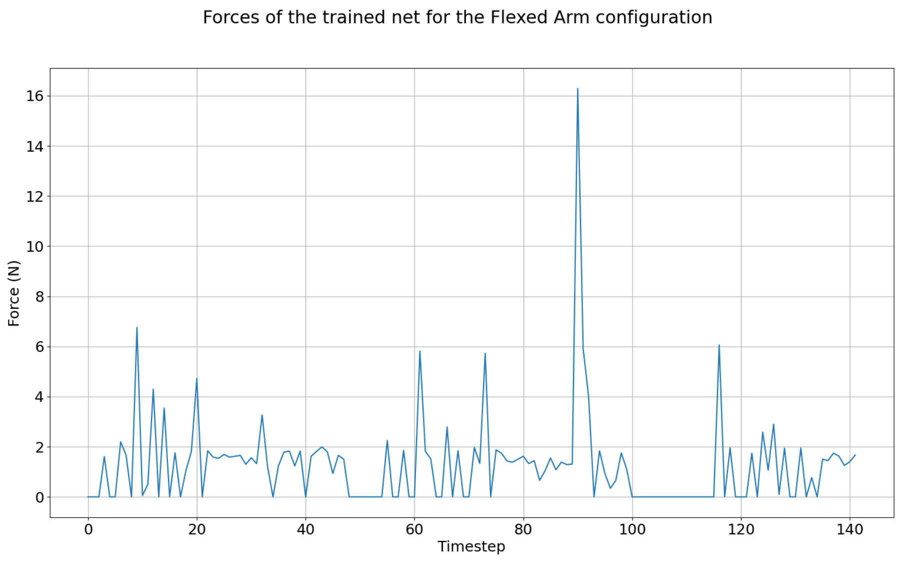

Figure 29: Forces along the trajectory of the trained net with the arm flexed.

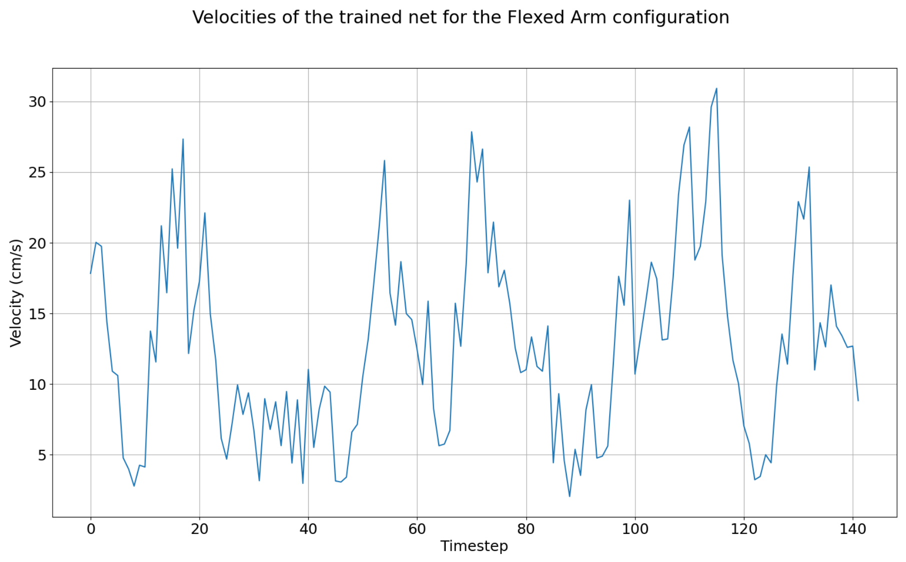

Figure 30: Velocity along the trajectory of the trained net with the arm flexed.

C Implementation of DMPs. Supplementary Information

C.1 Figures from the No Compliance method for different time scaling factor values

C.1.1 $\tau = 1.5$

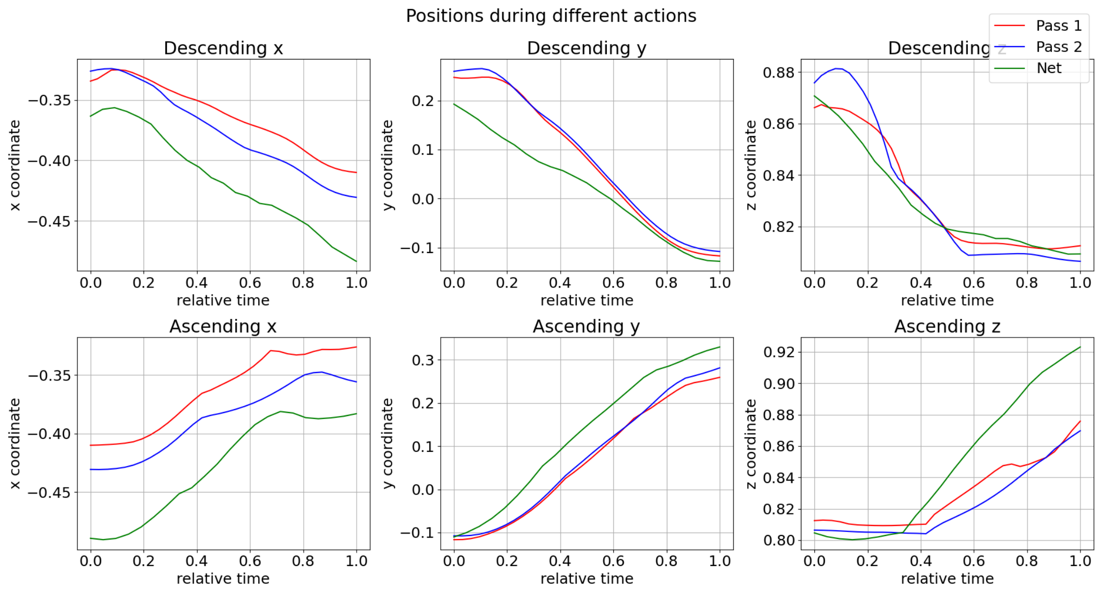

Figure 31: Evolution of the x, y and z coordinates of the DMPs for $\tau = 1.5$ compared with the net for the No Compliance scenario.

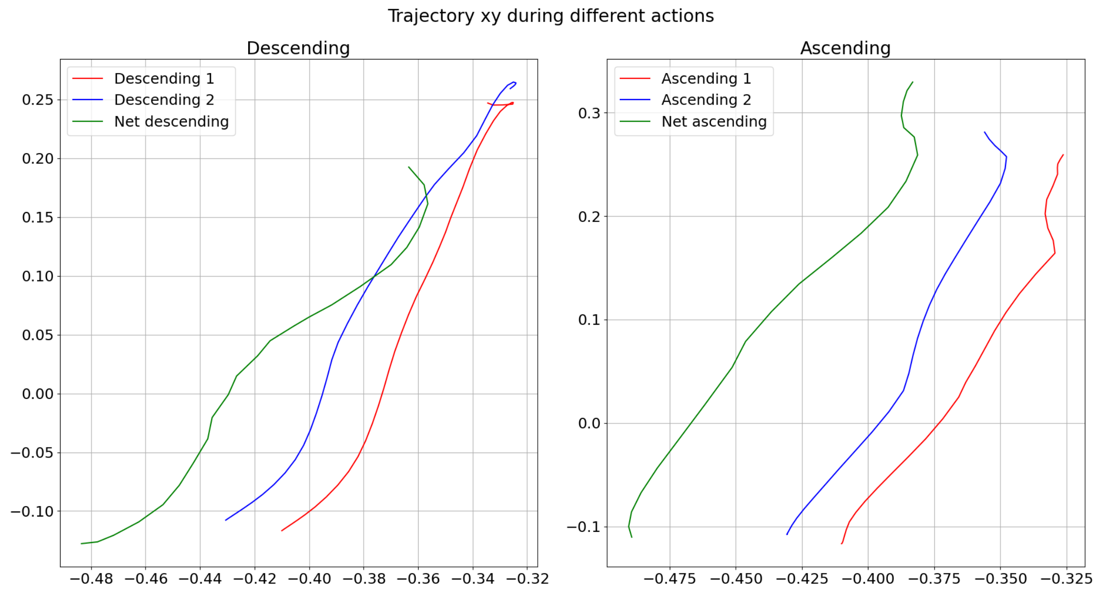

Figure 32: Trajectory in the z plane of the DMPs for $\tau = 1.5$ compared with the net for the No Compliance scenario.

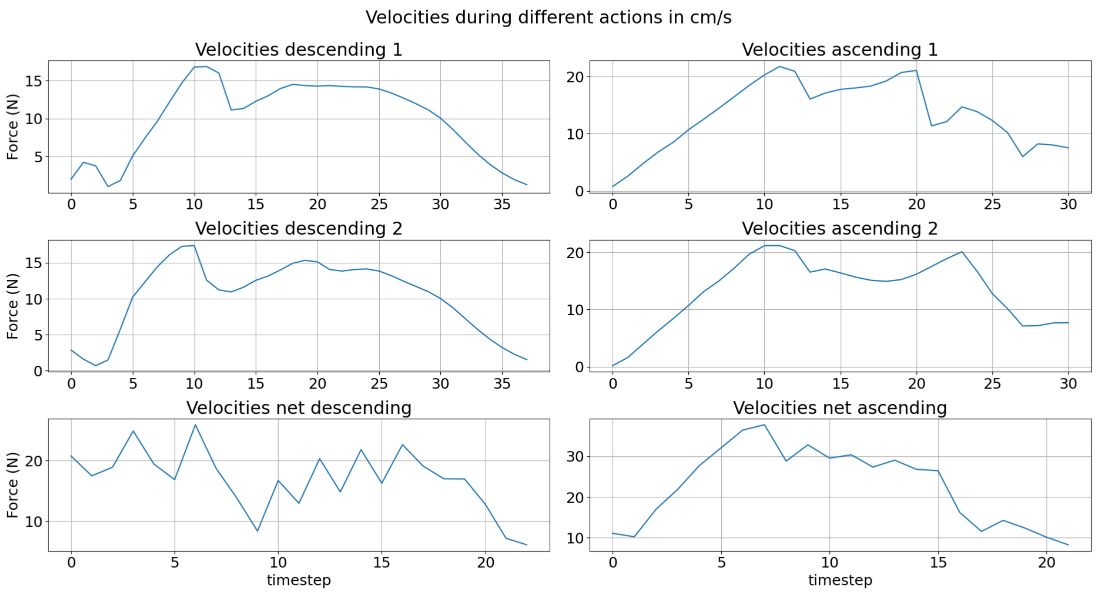

Figure 33: Velocity evolution of the DMPs for $\tau = 1.5$.

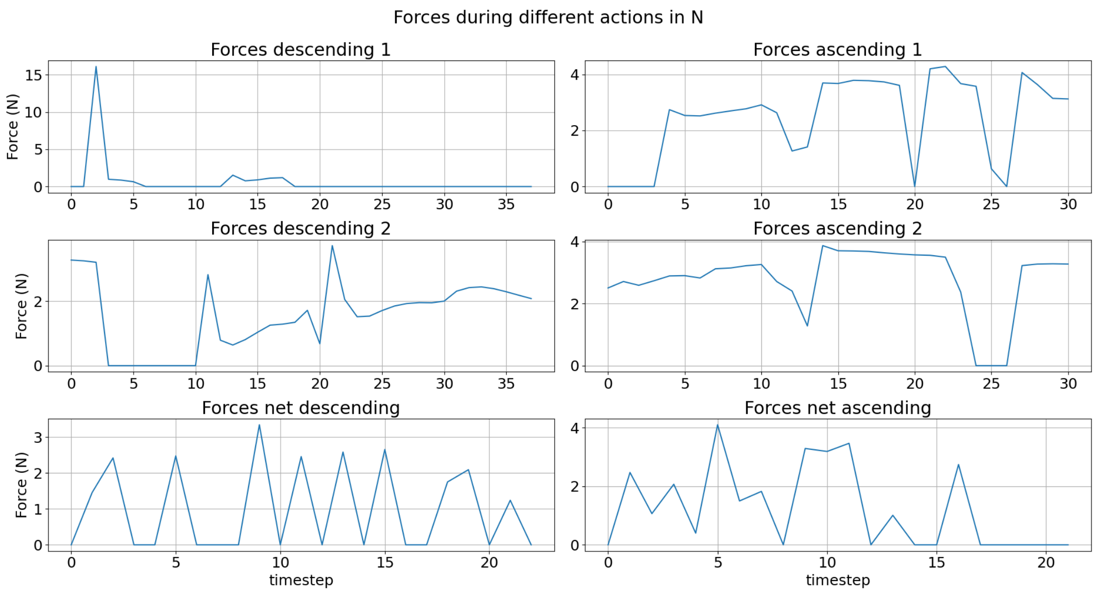

Figure 34: Forces evolution of the DMPs for $\tau = 1.5$ for the No Compliance scenario.

C.1.2 $\tau = 2$

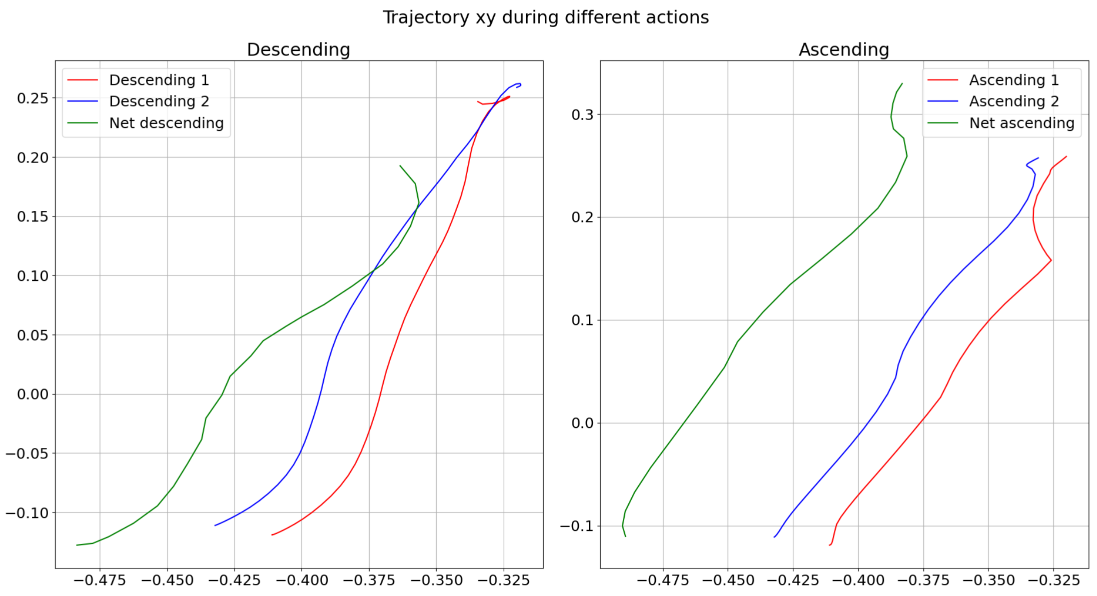

Figure 35: Trajectory in the z plane of the DMPs for $\tau = 2$ compared with the net for the No Compliance scenario.

C.1.3 $\tau = 3$

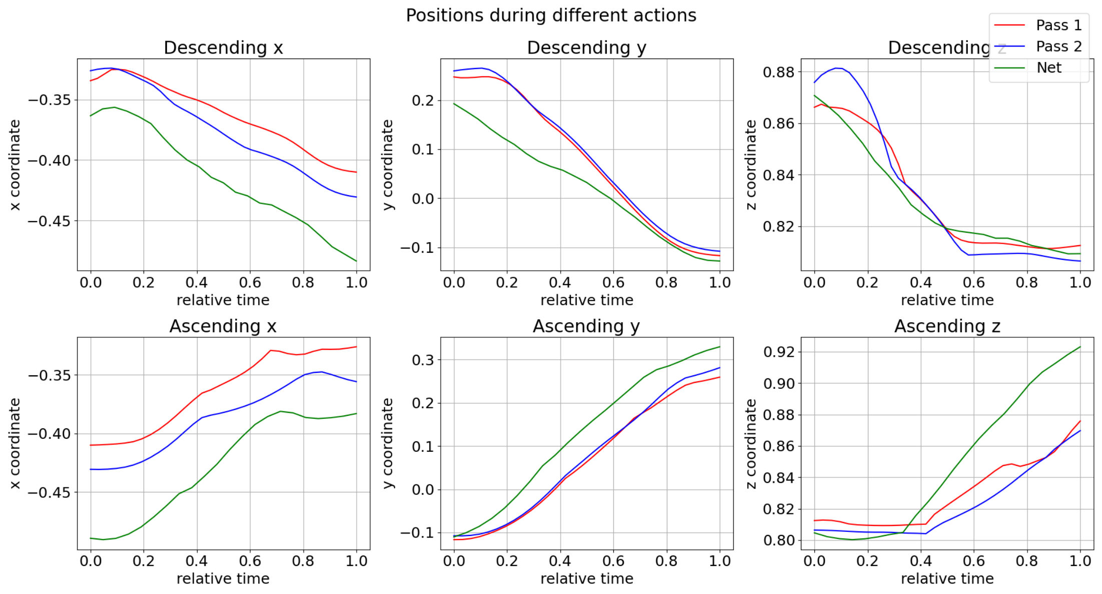

Figure 36: Evolution of coordinates of the DMPs for $\tau = 3$ compared with the net for the No Compliance scenario.

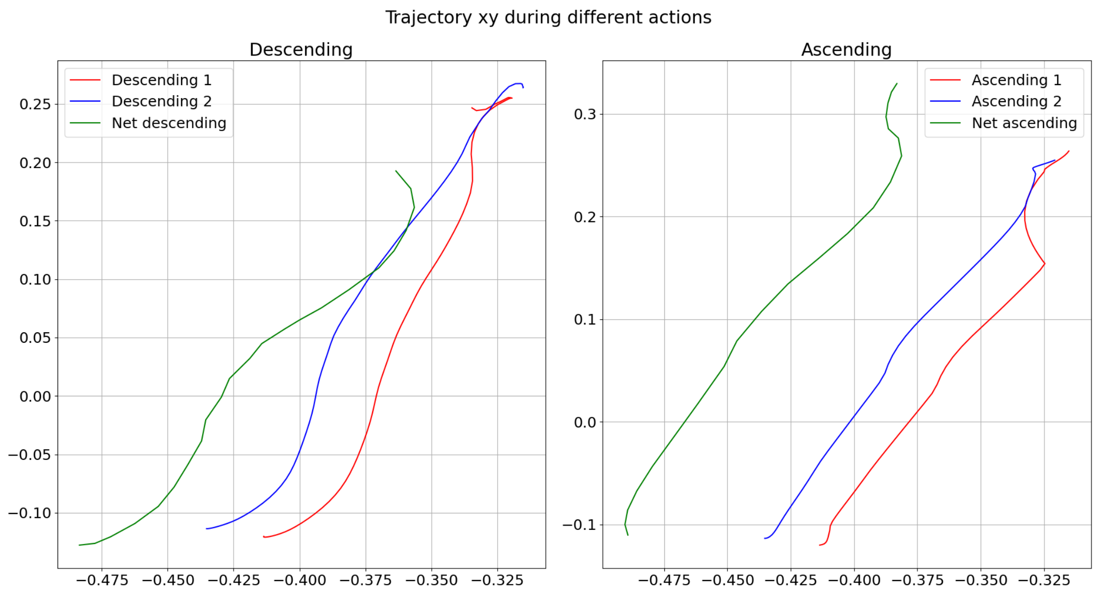

Figure 37: Trajectory in the z plane of the DMPs for $\tau = 3$ compared with the net for the No Compliance scenario.

Figure 38: Velocity evolution of the DMPs for $\tau = 3$ for the No Compliance scenario.

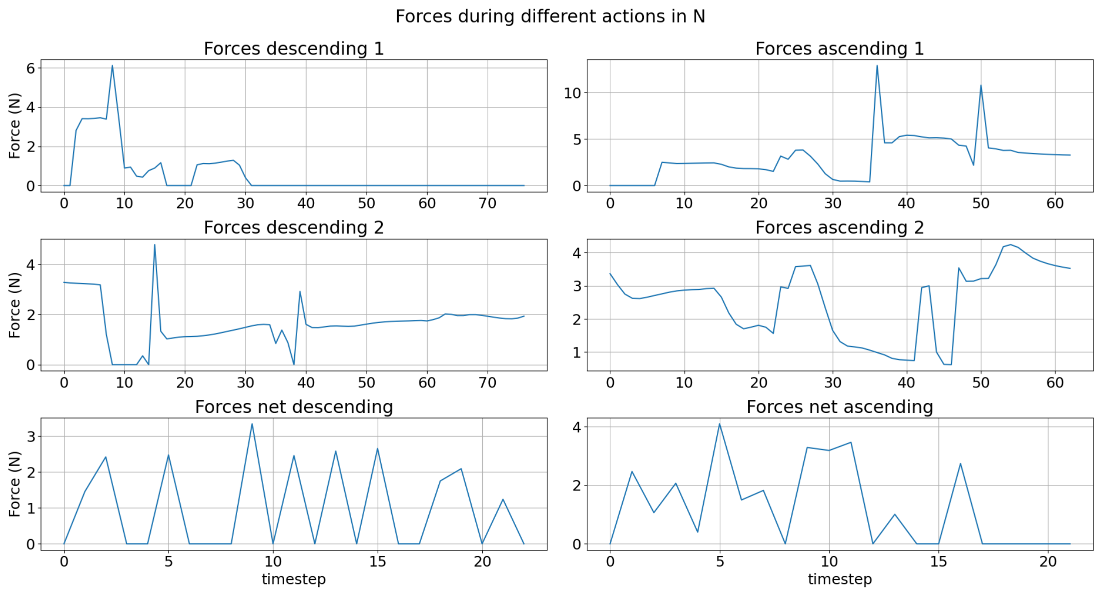

Figure 39: Forces evolution of the DMPs for $\tau = 3$ for the No Compliance scenario.

C.2 Figures from the Compliance method for different time scaling factor values for the Straight Arm Scenario

C.2.1 $\tau = 1.5$

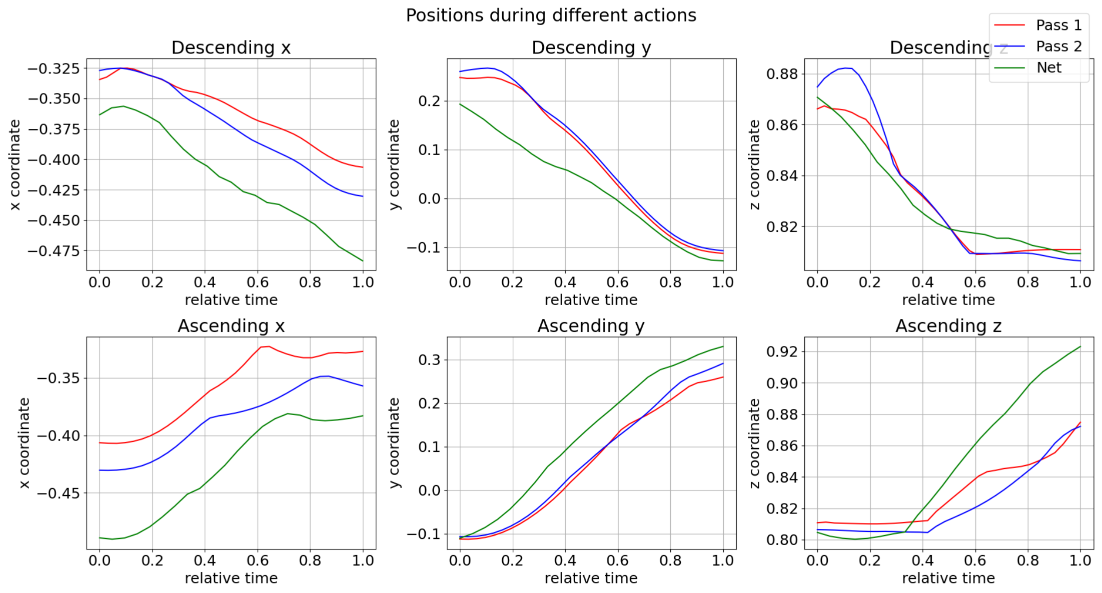

Figure 40: Evolution of coordinates of the DMPs for $\tau = 2$ with compliance and the net's.

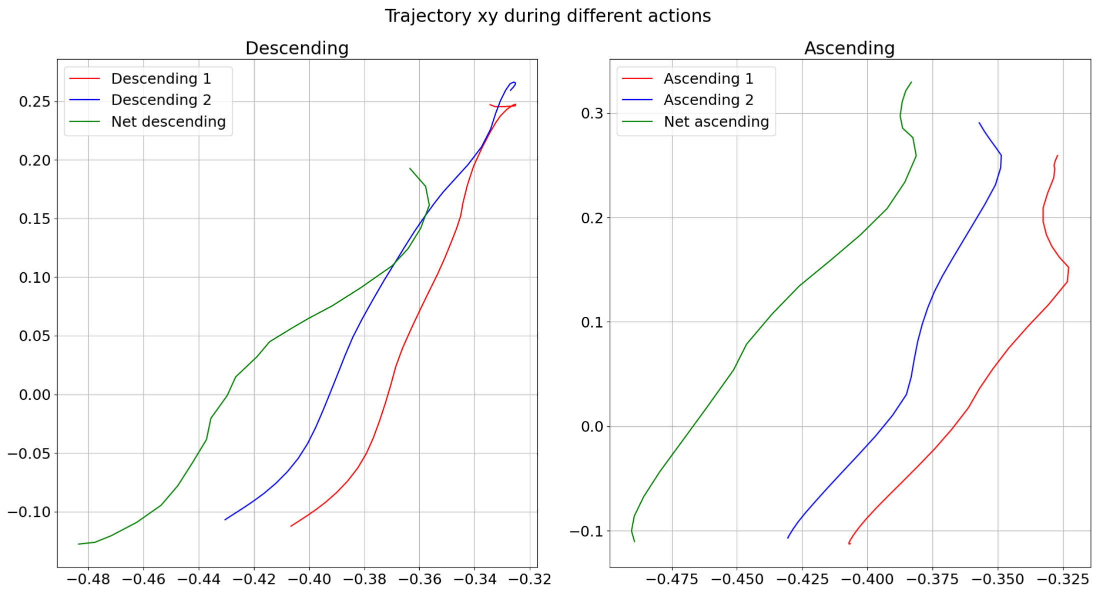

Figure 41: Trajectory in the z plane for $\tau = 1.5$ with compliance and the net.

C.2.2 $\tau = 2$

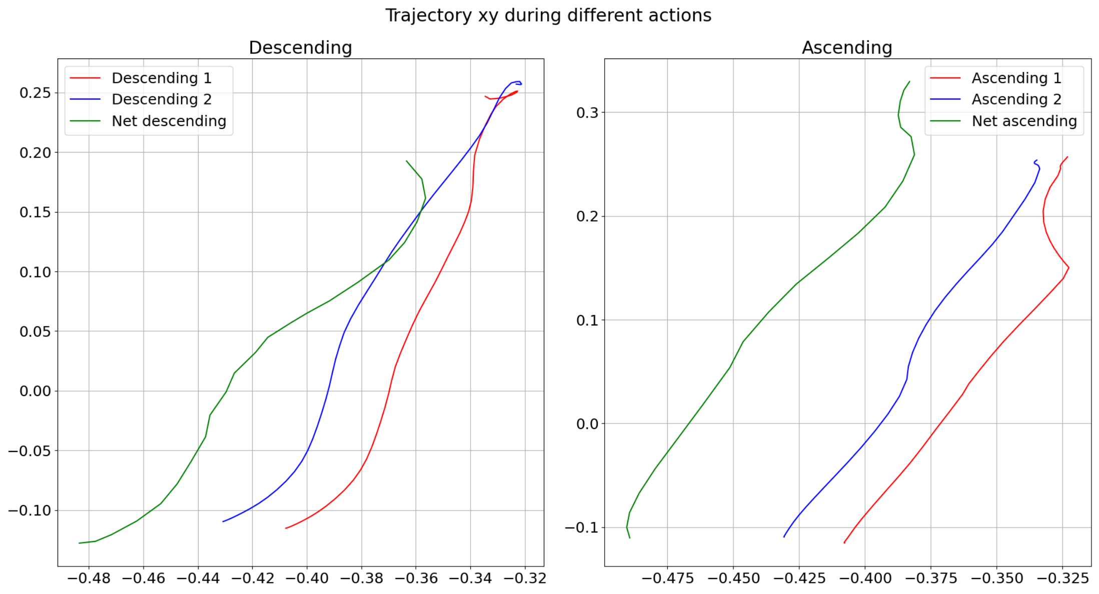

Figure 42: Trajectory in the z plane for $\tau = 2$ with compliance and the net.

C.2.3 $\tau = 3$

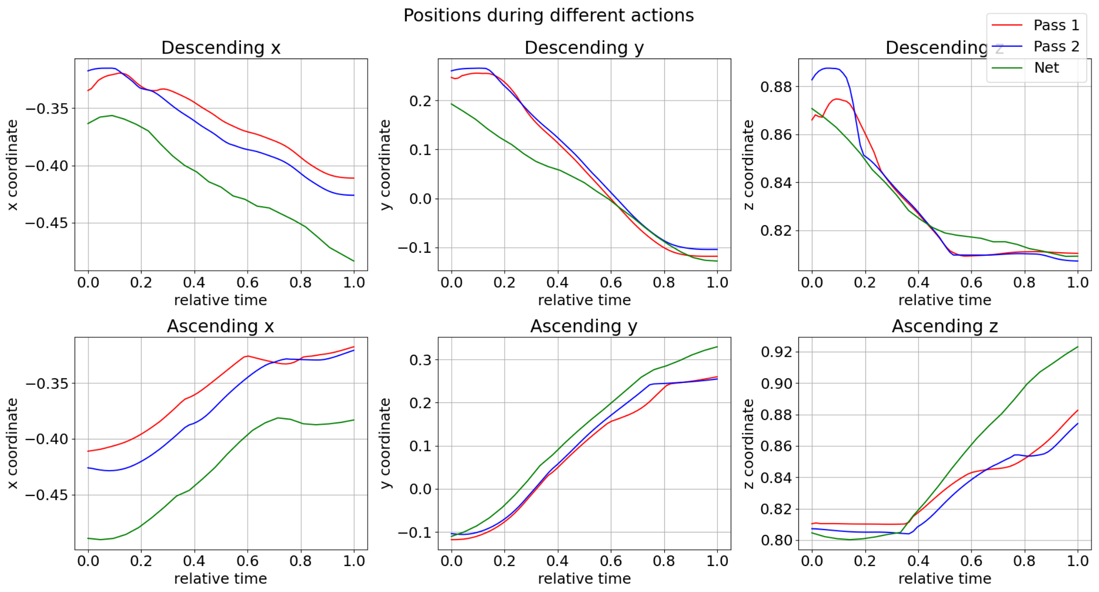

Figure 43: Evolution of coordinates of the DMPs for $\tau = 3$ with compliance and the net's.

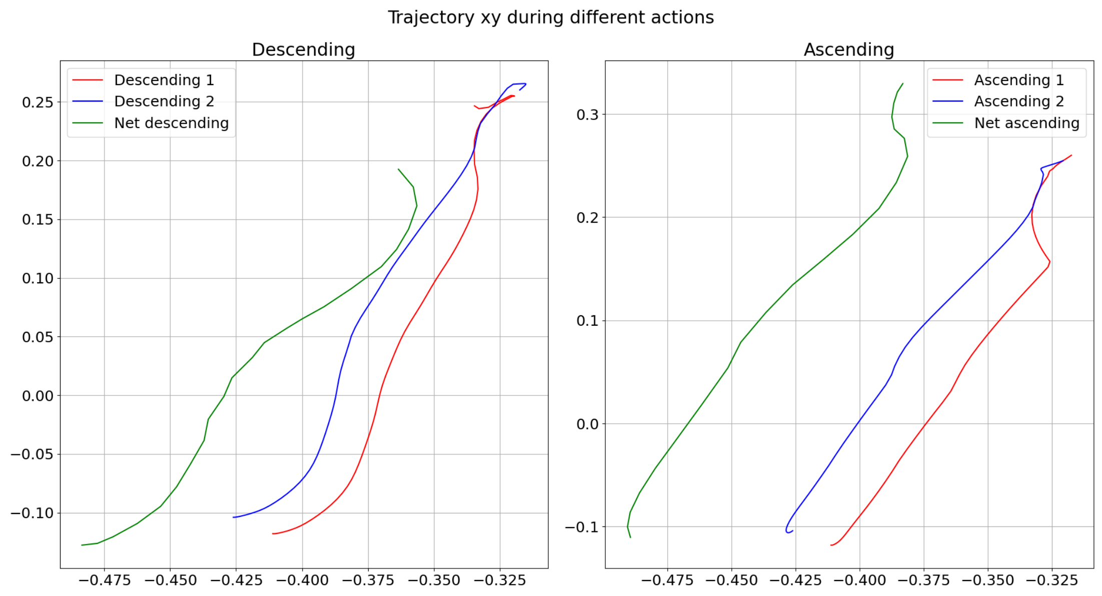

Figure 44: Trajectory in the z plane for $\tau = 3$ with compliance and the net.

C.3 Figures from the Compliance method for different time scaling factor values for the Flexed Arm Scenario

C.3.1 $\tau = 1.5$

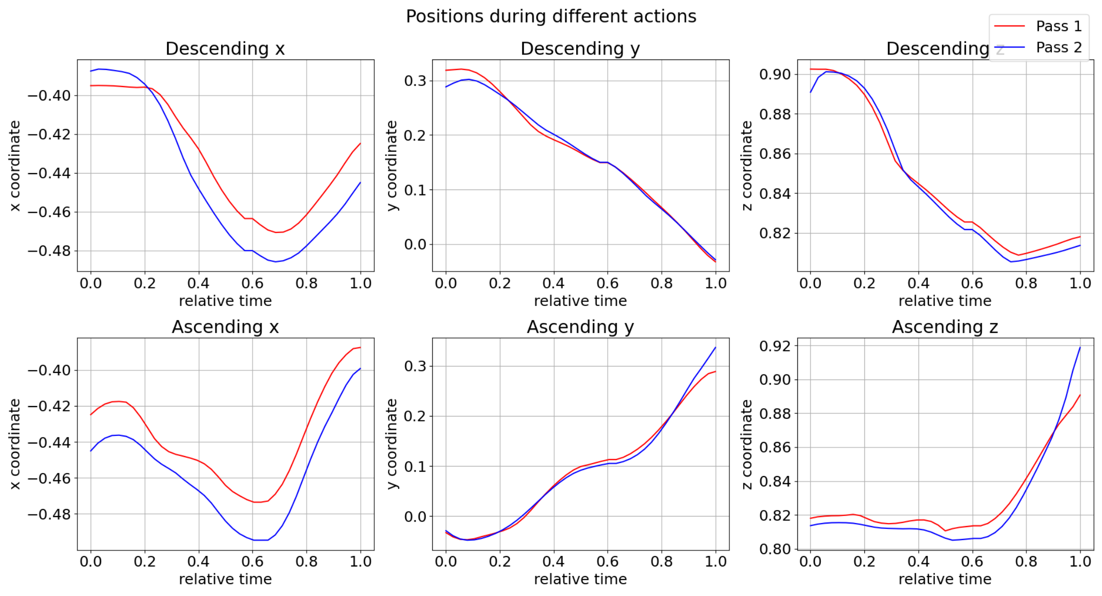

Figure 45: Evolution of coordinates of the DMPs for $\tau = 1.5$ with compliance for the Flexed Arm Scenario.

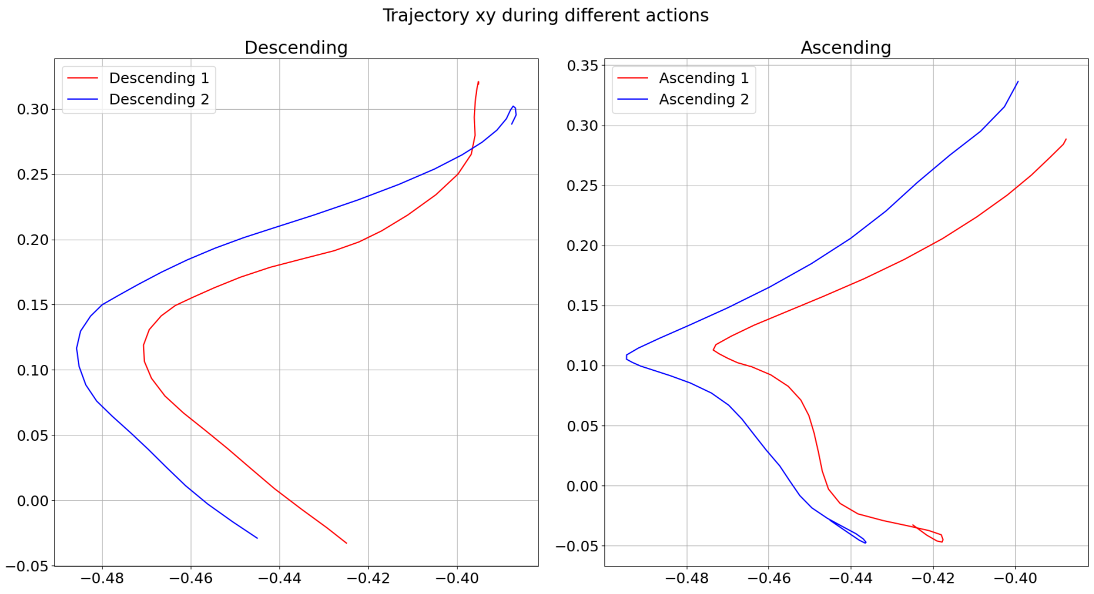

Figure 46: Trajectory in the z plane for $\tau = 1.5$ with compliance for the Flexed Arm Scenario.

C.3.2 $\tau = 2$

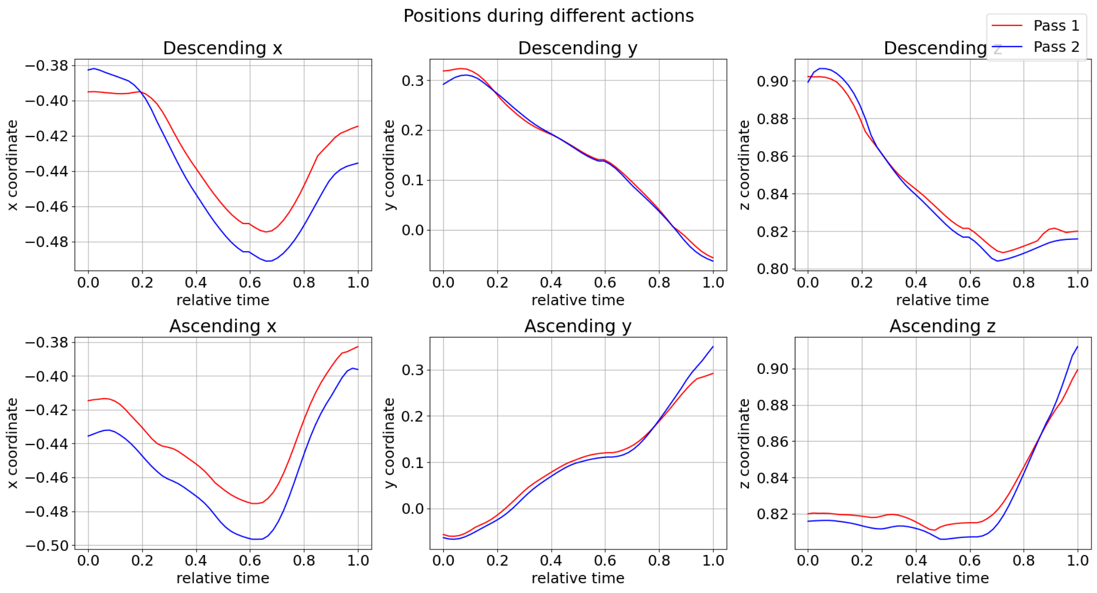

Figure 47: Evolution of coordinates of the DMPs for $\tau = 2$ with compliance for the Flexed Arm Scenario.

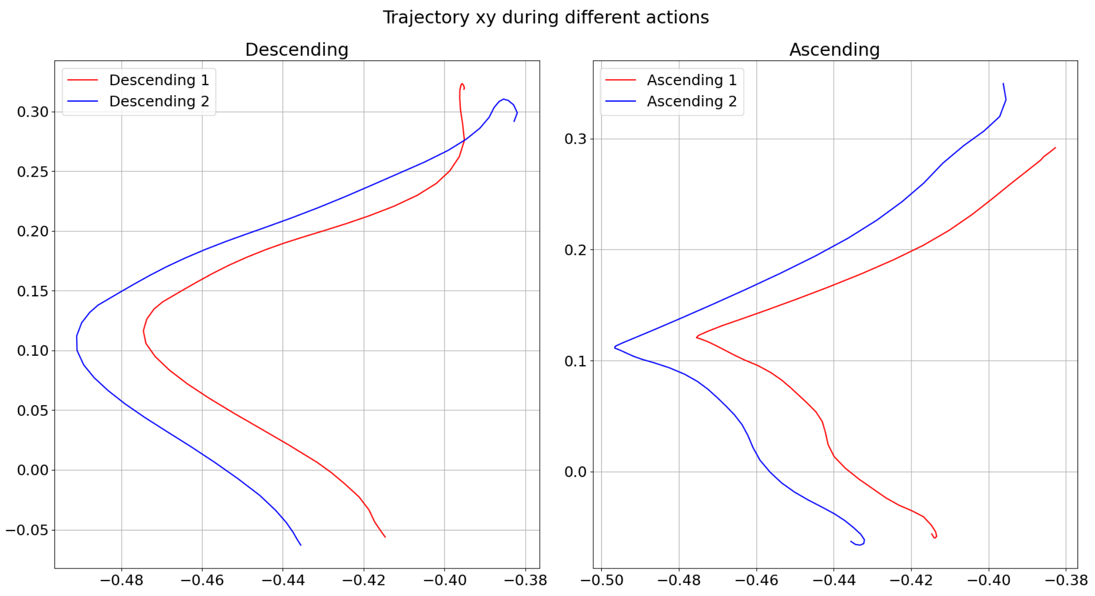

Figure 48: Trajectory in the z plane for $\tau = 2$ with compliance for the Flexed Arm Scenario.

C.3.3 $\tau = 3$

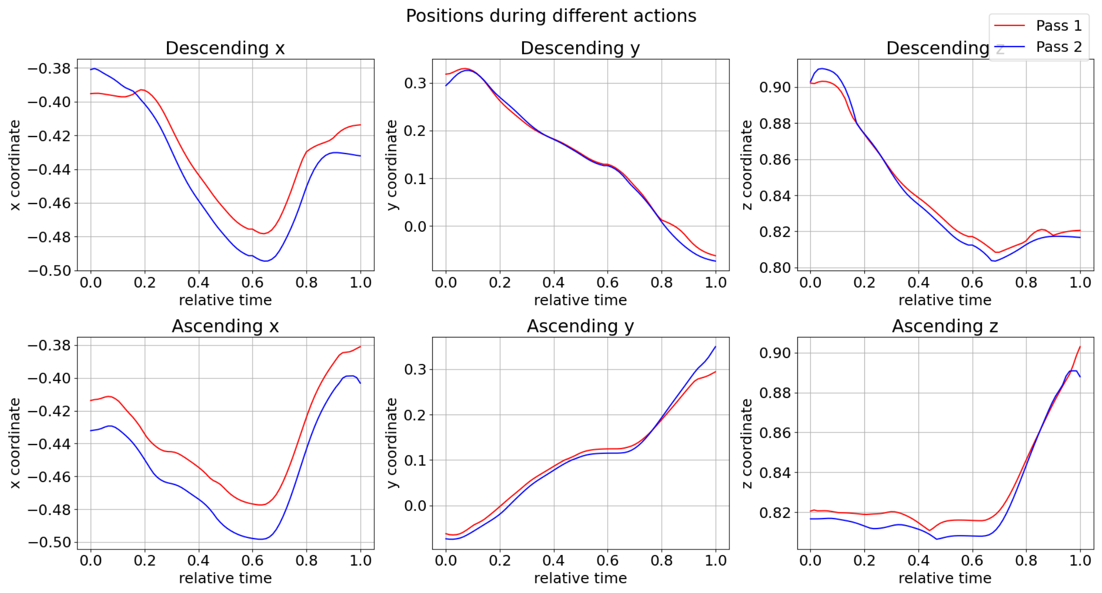

Figure 49: Evolution of coordinates of the DMPs for $\tau = 3$ with compliance for the Flexed Arm Scenario.

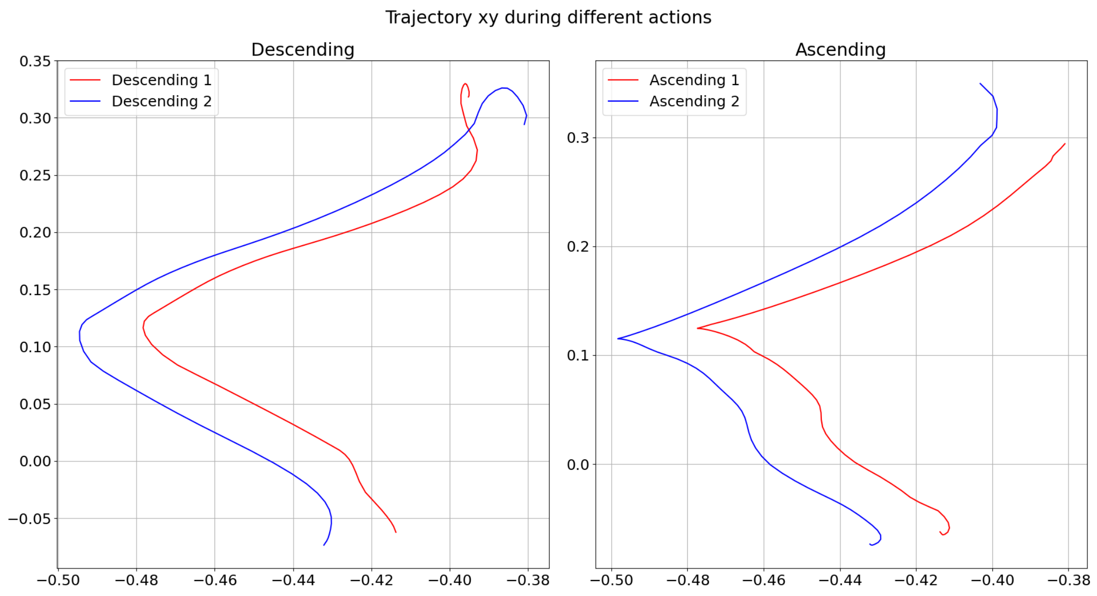

Figure 50: Trajectory in the z plane for $\tau = 3$ with compliance for the Flexed Arm Scenario.