Master in Artificial Intelligence

Master Thesis

A Learning from Demonstration Approach for Robot Trajectories through Motion-Sensing Human Demonstrations

Author:
Ángel Poc López

Supervisor:
Joan Aranda López (UPC)

Tutor:
Cecilio Angulo Bahon (UPC)

Universitat de Barcelona

UPC

Universitat Rovira i Virgili

April 20th 2020

Abstract

The objective of this thesis is to teach a Baxter robot to learn certain arm trajectories. The robot must be capable of generalizing the primitive movement of the trajectory to new unseen poses. The thesis is framed within a robotized kitchen project with aims to help people with mobility problems. To solve this problem end, a human will record demonstrations, which will be translated to the robots' morphology using an Inverse Kinematics (IK) module. For the learning part Dynamic Movement Primitives (DMP) will be used, due to their capability to take profit of human experience. The proposed system works in the majority of the scenarios, but, it would be expected to behave better when generalizing to new orientations of the arm. However a proposal has been suggested to correct this issue.

Acknowledgements

I would like to thank Professors Joan Aranda and Cecilio Angulo for giving me the opportunity of introducing myself to the world of robotics, a field in which I had always thought to be in contact with. Additionally, I would like to thank them for giving me the advice I needed to progress on my thesis. But, most importantly, I am very grateful to my parents, my brother, and Silvia for being always supportive with me.

Contents

List of Figures

List of Tables

List of Videos

Acronyms

DMP Dynamic Movement Primitives

DOF Deegrees of Freedom

GMM Gaussian Mixture Models

HMM Hidden Markov Models

IK Inverse Kinematics

IL Imitation Learning

InHANDS Interactive robotics for Human Assistance iN Domestic Scenarios

LfD Learning from Demonstration

MSE Mean Squared Error

NN Neural Network

PI^2 Path Integral Policy Improvement

RBF Radial Basis Function

RL Reinforcement Learning

Introduction

1.1 Motivation

Robots nowadays dispose of built-in systems that allow them to move their limbs from one to another position with the less effort possible. However, if you define an starting and an ending point for a movement, the robot will take the optimal path, without having into account if there are possible obstacles and ignoring if they are transporting objects whose content can be dropped, like liquids.

In favor of solving this kind of problems, Optimal Control Theory and, more recently, Reinforcement Learning (RL) communities, have spent a lot of their efforts. This is due, in fact, to the inherent complexity of the Robotics field. We can see this complexity solely observing that, the representations used to define the problems often use high-dimensional continuous states and actions. Different proposals have been taken into account in order to tackle this problems, as detailed in the following survey [1], however, Learning from Demonstration (LfD) and Imitation Learning (IL) can be some of the most interesting ones when it comes to a problem with a high number of restrictions, and, where human experience can be helpful.

Using this kind of approaches, and more precisely Dynamic Movement Primitives (DMP) [2][3], allows us to get rid of the difficulty of defining complex reward functions in RL, which would be needed to meet those hard restrictions, by presenting human demonstrations from which the algorithms can learn and generalize.

Unlike in an industrial environment, where every movement a robot makes is calculated with high precision and where no changes neither in the environment nor the task are expected, we aim to build an assistive technology. In such a field we expect that: the humans that are going to be assisted do not have any kind of technical knowledge; the environment changes; and new tasks can be needed in order to fulfill human's new requirements.

To this end, the technology created must be capable of generalizing to new situations and it must be easy to teach new tasks to the robot, even by non technical personnel.

1.2 Context

This thesis is framed within the Interactive robotics for Human Assistance iN Domestic Scenarios (InHANDS) project from the Automatic Control Department of the Polythecnic University of Catalonia. The part of the project this thesis is related to, aims to build a robotic kitchen in order to help people with mobility problems to achieve daily tasks. As it can be seen in Figure 1.1, there are several robotic devices in the laboratory which will help the user, among them, the Baxter Research Robot [4], is the one in which this work will be based on. A picture of the robot can be observed in Figure 1.2.

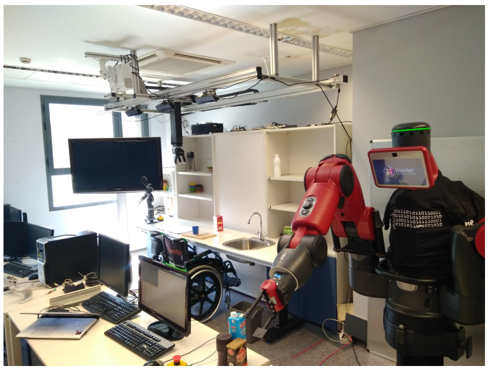


Figure 1.1: Kitchen laboratory.

1.3 Objectives

The main objective of this thesis is to design a LfD approach that allows the robot to learn certain trajectories. The goal is to generalize those trajectories in order to start or end in different positions than the specified originally.

Instead of using the robot arms to record the trajectories, these will be recorded though human demonstrations using a Kinect camera. This allows

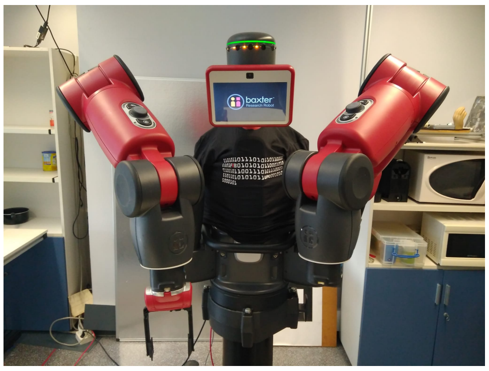


Figure 1.2: Baxter Research Robot.

anybody to record the trajectories they desire without having any technical knowledge on robotics. For sake of simplicity, this thesis will only work on trajectories for the right arm of the Baxter robot.

In order to accomplish the final goal, the following sub-goals must be achieved:

1. Create a system able to capture different joints of a human body in movement.

2. Make a conversion of reference systems from the camera center to the human base spine.

3. Preprocess the data to avoid errors caused by the sensing system.

4. Calculate the orientation of the hand, to a format understood by the Baxter robot.

5. Create a system that given a position and an orientation of the hand can resolve a configuration of the robot joints for the right arm. Additionally, this system should take into account previous states of the robot arm.

6. Create and tune a system via DMPs, able to learn proposed trajectories and to generalize to new non previously seen situations.

1.4 Outline

In Section 2 we will comment the State of the Art on the area. Later in Section 3 the solution developed in order to solve the problem will be explained. In section 4 the experimental setup will be commented to subsequently explain the results obtained in section 5. In section 6 we will explain the conclusions this thesis leads to and finally in section 7 we will give details of how to keep developing this work further from this thesis.

State of the Art

In this section we will discuss different solutions in the State of the Art to solve the problem of learning trajectories in a robot. More concretely we will review several solutions specifically designed for Baxter Research Robots.

2.1 Optimal Control Problems

Optimal Control is an area that calculates optimal control policies for dynamical systems optimizing cost functions defined by the user. Classical approaches to solving this problem often require complete knowledge of the system dynamics, however, Reinforcement Learning (RL) discipline has been developing several tools to overcome this problem. In RL, an agent learns a policy to optimize a long-term reward by means of interaction with the environment in order to accomplish a goal. This survey [5], shows different methods for optimal control problems using RL. It treats, additionally, the optimal tracking control problem, which is related to the topic of this thesis. In optimal tracking the controllers should ideally make the states or outputs of the system track a reference trajectory. In particular, [6] offers a solution to the optimal tracking control problem in continuous time systems using an off-policy Integral RL algorithm which allows to ignore the system dynamics. More algorithms specifically developed to cope with non-affine systems can be found in [7] and [8]. However, in this kind of solutions, it is necessary to still manually define complex reward functions which may be difficult to be defined, and above all, they do not take profit of the experience a human can contribute.

In this survey on Learning from Demonstration robotic systems [9], the authors recommend different solutions in order to convert the human skills and experience into something valuable for the problem. This kind of solutions offer a framework in which the user does not need to have programming skills, it just needs to create the movements that the algorithm will assimilate, either by kinesthesic, motion-sensor or teleoperated demonstration. The mentioned work proposes different approaches for LfD systems. The first of them is using Hidden Markov Models (HMM) given their ability to encapsulate both spatial and temporal variability. These systems can also be used to recognise and generate new motions at the same time, due to the nature of their generative model. Other proposed approach is the use of Gaussian Mixture Models (GMM), for systems with high noise or where not enough samples can be collected. Among those models, it is highlighted the use of Dynamic Movement Primitives (DMP) for being suitable for following a specific path, being robust to perturbations and also for being faster than HMM. Above all, DMPs are preferable for their favorable components for formalizing nonlinear dynamic movements.

Due to this facts, the solution proposed in this thesis will be based on DMPs.

2.2 Baxter Research Robot

In [10] the authors pursue the same goal of this thesis, teaching a Baxter robot through human demonstrations using a Kinect camera. However, their process is different, since they dispose of additional decives, like Myo Armbands. This armbands are capable of calculating the wrist and elbow angular velocities. Instead of using the Inverse Kinematics (IK) module of the Baxter robot, about which we will talk later, they create their own mapping of human to robot joints. Additionally, to solve the learning part they use Kalman Filters and Radial Basis Function (RBF) Neural Networks.

A different proposal can be observed in [11], where they teach a Baxter robot using Kinect sensors. Instead of using Kalman Filters and Neural Networks they propose an extension of DMPs which allows them to learn from multiple demonstrations, by means of modelling them using GMMs. They feed the DMP system with joint space trajectories, however they do not make explicit the conversion from the Human Kinect skeleton to the Baxter joints.

A more ambitious approach can be observed in [12]. Here, they introduce a novel LfD framework that divides the process in three independent but connected phases: the teaching phase, modelled by a GMM; the learning phase, which makes use of DMPs; and finally the reproduction phase, modelled using a RBF NN. This system is also feed with trajectories in the joint space.

We can appreciate how several approaches opt for using DMPs in some part of the learning process and how all of them make some previous conversion, if necessary, to the joint space before learning.

Proposal

The developed proposal can be divided into three parts: the data collection part, in which a sensing device, in our case a Kinect camera, will be used to record the human demonstrations; the preprocessing step, where the collected data is processed to avoid errors and the orientation of the demonstrator's hand is calculated; and finally the robot learning part, where the processed records are translated into a format the robot interfaces can use, feeding the DMP system with this data to be.

In LfD proposals there are different approaches to demonstrate the trajectories [9][13]. In Kinesthesic approaches the movements are recorded directly from the robot. In this cases, the user moves the robot's endpoint in zero gravity mode. This could be complicated as moving the arms of the robot is not easy given the 6 or 7 Deegrees of Freedom (DOF) they commonly have. Another approach is to use teleoperated systems, but this would lead to less natural movements, as it is difficult to move all the 7 DOF in real time. Finally, motion-sensing systems can be used to capture the position of a human's joints. While this could lead to more natural movements, the user relies on the accuracy of computer vision algorithms which may be low in some cases. Additionally, the correspondence problem [14] appears in this approach, since the human joints' and the robot joints' morphology is not the same.

However, the last approach is the one that requires less knowledge from the user to execute the demonstration, and therefore, it will be selected to develop the work.

For the purpose of solving the correspondence problem between the captured data and the robot interfaces, we will use the Inverse Kinematics system offered by the robot. This system, given a position in the Cartesian space and an orientation of the robot's endpoint with respect to its base, can calculate a set of joint angles that will allow the robot arm to get to this pose. To clarify, the demonstrator's hand will act as the robot's endpoint.

3.1 Data Collection

As mentioned before, the motion-sensing approach will be used in this work. In order to sense the human body to capture the data, we will use a Kinect device [15] connected to a Windows 10 system.

A Kinect camera is able to detect different joints of a human body applying Computer Vision algorithms to the depth and color images it can capture. In Figure 3.1 we can observe the different joints provided by the device. Given the joints we can capture and the fact that we have to provide a position and an orientation, the right wrist coordinates (x,y,z) will be saved for indicating the position of the robot endpoint, and the right hand and right thumb coordinates will be used together with the wrist coordinates to calculate the orientation of the hand with respect to the spine base coordinates of the body.

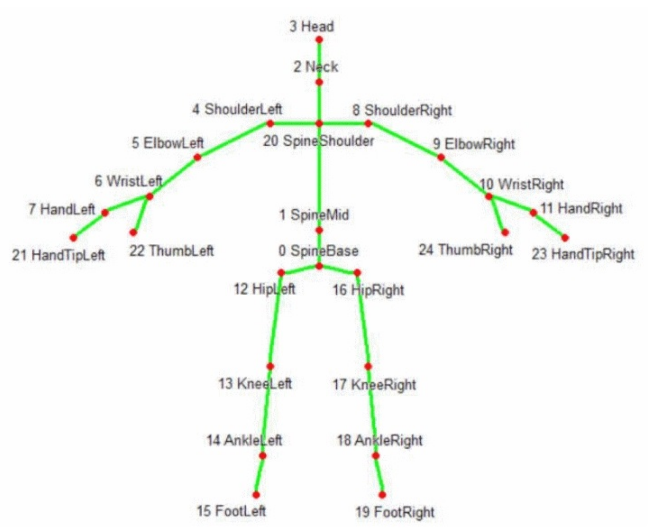


Figure 3.1: Different joints the Kinect camera can capture. [16]

To record the data, the Body Basics-WPF example program provided by the Kinect for Windows SDK 2.0 [17] has been modified using C#. The Kinect API offers a variable telling whether the pose of the hand has been captured with high confidence or not, but after testing this variable and seeing that there were large periods of up to 8 seconds with missing data we decided not to use it. Therefore, the timestamp of every record will be saved for later manual preprocessing, making sure the poses used in the learning process are clean.

A picture of the Kinect camera can be seen in Figure 3.2 along with its coordinate system. Noticing during early tests the low reliability of the camera for capturing the hand and, above all, the thumb coordinates, different takes were recorded per every demonstration, having the camera in different positions but always maintaining the Y axis of the camera parallel to the demonstrator body, in front of him. This way we make sure that some of the takes captured will be correct.

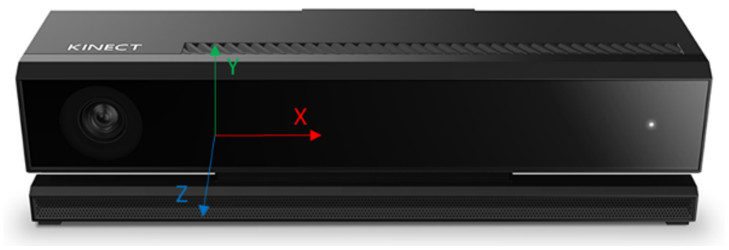


Figure 3.2: Coordinate system of the Kinect devices. [18]

3.2 Preprocessing

At this stage of the project, the data is being collected with respect to the Kinect coordinate system. However, if we observe the Baxter coordinate system in Figure 3.3, we can see that it does not correspond with the Kinect coordinate frame. To solve this, it will be necessary to create a rotation matrix which transforms the points from the Kinect to the Baxter coordinate frame, assuming the center of the new coordinate frame is the base spine of the human demonstrator. After this change of reference systems, we have to clean the data to avoid the robot to do unexpected movements that could damage itself. To end this stage, we have to calculate the hand orientation with the data we dispose of.

The change of reference system is done before saving the Kinect data using C# and the rest of the work is done using Python.

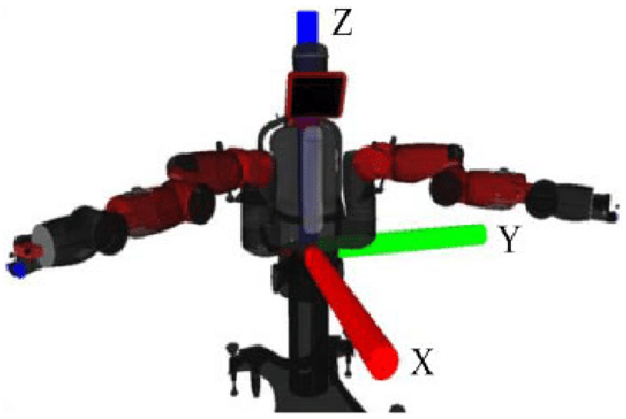


Figure 3.3: Coordinate system of the Baxter Research Robot. [19]

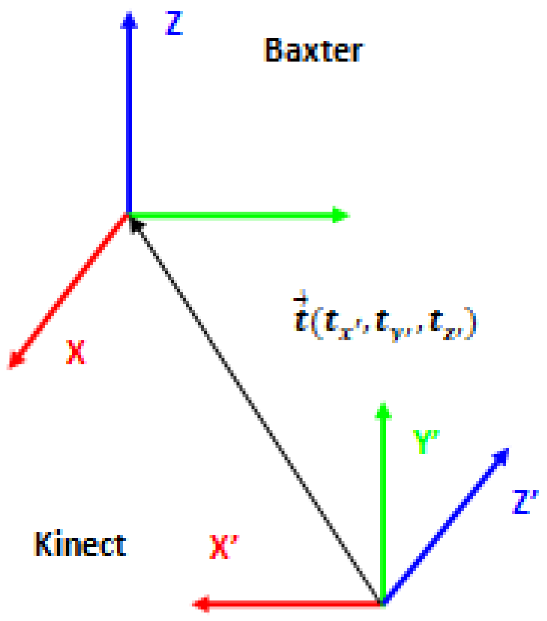


Figure 3.4: Comparison of the two coordinate systems.

Kinect to Baxter transformation

Considering that the Baxter coordinate system is in front of the Kinect one, and assuming that all the axis of one frame are parallel to one of the axis of the other, as we can see in Figure 3.4, we will have to apply several transformations to the Kinect system in order to convert it to the Baxter scheme. First, we will translate it a vector $\mathbf{t}(t_{x'}, t_{y'}, t_{z'})$, corresponding to the coordinates of the base spine. Then a rotation of -90º will be applied around the X' axis to later apply another rotation of 90º around the Z' axis.

To do this transformation, we can resort to books like [20], that explain algebraic methods for robotics. The translation can be defined algebraically with the following transformation matrix:

$$\mathbf{T}(\mathbf{t}) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.1}$$

The rotation of -90º around the X' axis can be defined as:

$$\mathbf{Rotx}'(-90^\circ) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

And finally, the rotation around the Z' axis of 90 degrees is determined as:

$$\mathbf{Rotz}'(90^\circ) = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

If we multiply the matrices, we obtain the following transformation matrix:

$$\mathbf{T} = \mathbf{T}(\mathbf{t})\mathbf{Rotx'}(-90°))\mathbf{Rotz'}(90)) = \begin{bmatrix} 0 & -1 & 0 & t_{x'} \\ 0 & 0 & 1 & t_{y'} \\ -1 & 0 & 0 & t_{z'} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

Thus, if we multiply $\mathbf{T}$ by a vector of coordinates in the Baxter system, we obtain the coordinates in the Kinect system:

$$\begin{bmatrix} r_{x'} \\ r_{y'} \\ r_{z'} \\ 1 \end{bmatrix} = \mathbf{T} \begin{bmatrix} r_x \\ r_y \\ r_z \\ 1 \end{bmatrix} \tag{3.5}$$

However, as what we want is the opposite, to obtain the Baxter coordinates given a vector of Kinect coordinates, we will calculate the inverse of the transformation matrix $\mathbf{T}$, what will help us to solve the system:

$$\mathbf{T^{-1}} = \begin{bmatrix} 0 & 0 & -1 & t_{z'} \\ -1 & 0 & 0 & t_{x'} \\ 0 & 1 & 0 & -t_{y'} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

$$\begin{bmatrix} r_x \\ r_y \\ r_z \\ 1 \end{bmatrix} = \mathbf{T^{-1}} \begin{bmatrix} r_{x'} \\ r_{y'} \\ r_{z'} \\ 1 \end{bmatrix} \tag{3.7}$$

With this transformation, we can convert the wrist, hand and thumb vectors from the Kinect reference system to the human body system, based at the base spine, or which is equivalent, the Baxter robot coordinate system.

Data Cleaning

Once the wrist, hand and thumb coordinates are in the Baxter reference, and given that we have sufficient data (records at 30 Hz), several measures are going to be taken in order to avoid possible errors and outliers during the data capturing.

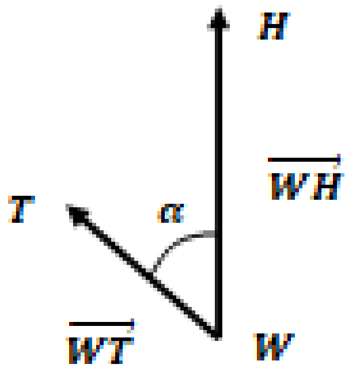


Figure 3.5: Representation of the hand vectors.

Let $\mathbf{W}(\mathbf{W_x}, \mathbf{W_y}, \mathbf{W_z})$ be the coordinates of the wrist, $\mathbf{H}(\mathbf{H_x}, \mathbf{H_y}, \mathbf{H_z})$ the coordinates of the hand, and $\mathbf{T}(\mathbf{T_x}, \mathbf{T_y}, \mathbf{T_z})$ the coordinates of the thumb w.r.t. the base spine, as it can be seen in in Figure 3.5, we are going to define the vector going from the wrist to the hand ($\vec{\mathbf{WH}}$) and the vector going from the wrist to the thumb ($\vec{\mathbf{WT}}$) as:

$$\vec{\mathbf{WH}} = \mathbf{H} - \mathbf{W} = \langle H_x, H_y, H_z \rangle - \langle W_x, W_y, W_z \rangle \tag{3.8}$$

$$\vec{\mathbf{WT}} = \mathbf{T} - \mathbf{W} = \langle T_x, T_y, T_z \rangle - \langle W_x, W_y, W_z \rangle \tag{3.9}$$

In the first place, the records will be grouped every 0.5 seconds (2 Hz). As during early tests with the camera it was noticed that the thumb coordinate was very susceptible to errors, in order to discard obvious outliers, the records within the 0.5 seconds periods whose cosine between the $\vec{\mathbf{WT}}$ and $\vec{\mathbf{WH}}$ vectors is higher than 0.95 ($\approx 18^\circ$) will be removed. This value has been chosen because it gives rise to a degree from which we know the sensing device has made a calculation error.

$$\cos\alpha = \frac{\vec{\mathbf{WT}} \cdot \vec{\mathbf{WH}}}{\left\|\vec{\mathbf{WT}}\right\| \cdot \left\|\vec{\mathbf{WH}}\right\|} \tag{3.10}$$

For the remaining records in each group, a median filter will be applied in time for every dimension independently, assuming that the variations from a record to the following one should not be big, and that the variance in 0.5 seconds is low. This filter has been chosen instead of, for example, a mean filter, because of the following two reasons:

- It allows to discard records that are further from the mean, that is, outliers.
- It avoids selecting the mean, which could have been displaced by errors in the sensing device, but gives a strong estimation supported by the most repeated value.

Orientation computation

Once we have a record per 0.5 seconds, it is time to convert the data to a format the Robot Inverse Kinematics (IK) system can assimilate. To this end, it is needed to specify a cartesian position and an orientation, indicated as a quaternion [4]. Quaternions are an extension of the complex numbers which allow to define rotations, analogolously to other representations like rotation matrices or euler angles, but which prevent the Gimbal lock problem [21]. This problem occurs when two of the axis that define a 3 dimensional rotation are in a parallel configuration, reducing the transformation to a two dimensional space, which is less than the required one. The quaternions can be represented in the form $q_w + q_x\mathbf{x} + q_y\mathbf{y} + q_z\mathbf{z}$, where q_w, q_x, q_y and q_z are real numbers and $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ are quaternion units.

As we only have two vectors to define the orientation and we need to express it as a quaternion, we must make a conversion. We know that a rotation in a three dimensional space can be defined by three orthogonal vectors or basis, with which we can create a rotation matrix. Therefore, these three basis will derive from vectors $\vec{\mathbf{WH}}$ and $\vec{\mathbf{WT}}$. To finalize, the rotation matrix ${}^{\mathbf{B}}_{\mathbf{E}}\mathrm{R}$ will be created to define the orientation of the hand (endpoint) with respect to the base spine (baxter reference system).

$$ {}^{\mathbf{B}}_{\mathbf{E}}\mathrm{R} = \begin{bmatrix} \mathbf{n} & \mathbf{o} & \mathbf{a} \end{bmatrix} \tag{3.11} $$

As vector $\mathbf{a}$ we will use the vector $\vec{\mathbf{WH}}$ normalized to make it unitary:

$$ \vec{\mathbf{a}} = \frac{\vec{\mathbf{WH}}}{\left\| \vec{\mathbf{WH}} \right\|} \tag{3.12} $$

As we need three vectors to create a reference system and $\vec{\mathbf{WT}}$ is not orthogonal w.r.t. $\vec{\mathbf{a}}$, we will create orthogonal vectors as combinations of them. The vector $\vec{\mathbf{n}}$ will be the normal vector defined by the plane formed by the vectors $\vec{\mathbf{a}}$ and $\vec{\mathbf{WT}}$:

$$ \vec{\mathbf{n}} = \frac{\vec{\mathbf{a}} \times \vec{\mathbf{WT}}}{\left\| \vec{\mathbf{a}} \times \vec{\mathbf{WT}} \right\|} \tag{3.13} $$

And, finally, the vector $\vec{\mathbf{o}}$ will be defined by the cross product of vectors $\vec{\mathbf{a}}$ and $\vec{\mathbf{n}}$ (the cross product of two unitary vectors should be unitary, but we divide it by the norm to avoid numerical problems afterwards):

$$ \vec{\mathbf{o}} = \frac{\vec{\mathbf{a}} \times \vec{\mathbf{n}}}{\|\vec{\mathbf{a}} \times \vec{\mathbf{n}}\|} \tag{3.14} $$

After computing the vectors, we can define the Rotation matrix ${}^{\mathbf{B}}_{\mathbf{E}}\mathrm{R}$ as:

$$
{}^{\mathbf{B}}_{\mathbf{E}}\mathrm{R} = \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_y & a_z \end{bmatrix} \tag{3.15}
$$

Now, the rotation matrix to quaternion transformation can be calculated according to [20] as:

$$
q_w = \frac{1}{2}\sqrt{(n_x + o_y + a_z + 1)} \tag{3.16}
$$

$$
q_x = \frac{1}{2}\sqrt{(n_x - o_y - a_z + 1)} \tag{3.17}
$$

$$
q_y = \frac{1}{2}\sqrt{(-n_x + o_y - a_z + 1)} \tag{3.18}
$$

$$
q_z = \frac{1}{2}\sqrt{(-n_x - o_y + a_z + 1)} \tag{3.19}
$$

Offsets and smoothing

Due to the fact that the morphology of the human and the robot arms is not the same (the robot arms are bigger, inter alia), some offsets will be applied for some of the wrist coordinates. These offsets will be calculated experimentally for each of the future demonstrations.

To finalize, a Gaussian filter will be applied along the time axis for each of the three coordinates of the wrist, to smooth the movement and remove some high frequency details that could have been introduced by errors in the sensing system.

3.3 Robot Learning

In this section, once the demonstration is specified giving a cartesian position and an orientation, a system is developed to convert each cartesian trajectory to a trajectory in the joint space for then calling the DMP module which will learn it.

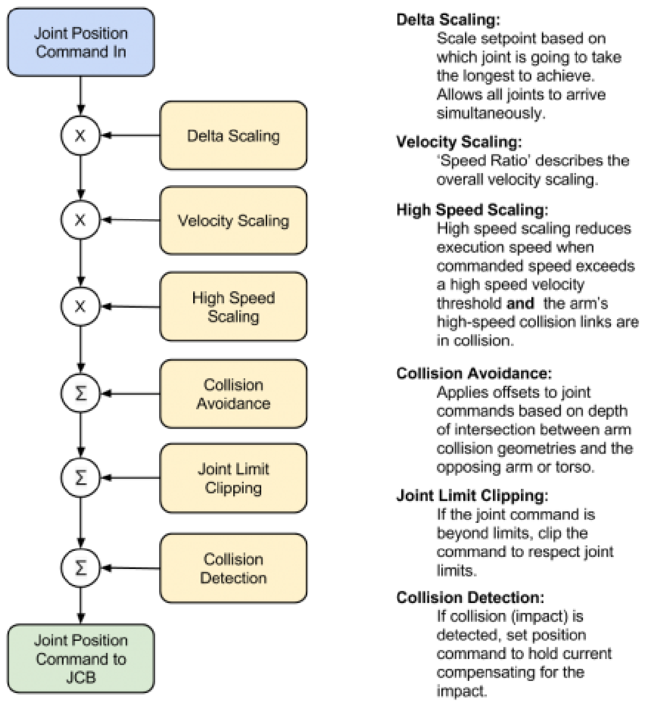
To interact with the robot we have used ROS Indigo through Python 2.7 in a Linux 14.04 LTS environment.

Inverse Kinematics

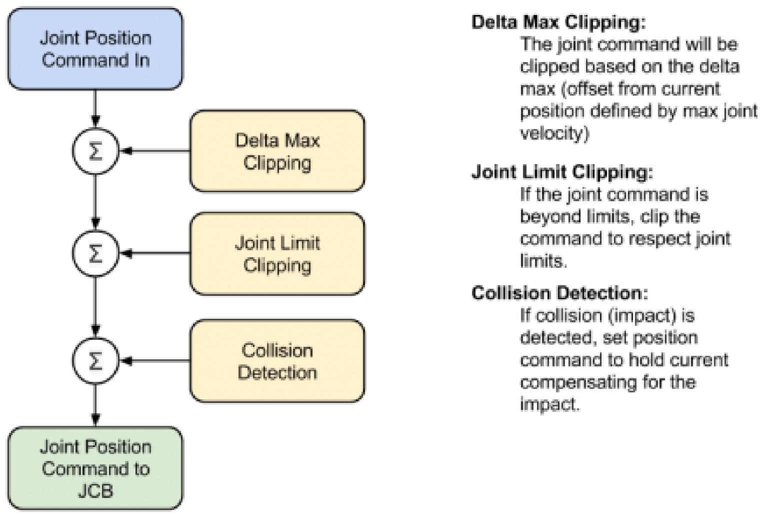
Up to now, for every demonstration we have N records, $\{r_0, ..., r_{N-1}\}$ in the form $r_i = \{p_i, q_i\}$, where $p_i = \{p_{ix}, p_{iy}, p_{iz}\}$ corresponds to the i-th cartesian position of the wrist and $q_i = \{q_{ix}, q_{iy}, q_{iz}, q_{iw}\}$ corresponds to the i-th orientation of the hand. Each record r_i defines the pose of the robot's endpoint at every moment. However, we cannot program the robot directly using these records.

The Baxter Research Robot disposes of four different arm control modes [4]: the joint position control mode, where we specify the angle at which we want each joint of the robot to be; the "raw" joint position control mode, similar to the previous one but offering a more direct control over the joints; the joint velocity control mode, where we can specify the velocity we want every joint to reach; and the joint torque control mode, where we can define the torques we want the joints to achieve. A comparison between the raw and normal joint position control modes can be observed in Figure 3.6. To manage the robot, we will use the "raw" position control mode. This mode has been selected because the Inverse Kinematics system offers the solution in joints and because it leads to more natural movements.

Nevertheless, we have a set of Cartesian positions and orientations r_i, and, what we want to achieve a list of joint positions $\{j_0, .., j_{N-1}\}$, where $j_i = \{j_{i0}, ..., j_{i6}\}$. Whether converting from a set of joint positions to the pose in the Cartesian space of the endpoint is something straightforward given


Joint Position Command In
X
Delta Scaling
X
Velocity Scaling
X
High Speed Scaling
Σ
Collision Avoidance
Σ
Joint Limit Clipping
Σ
Collision Detection
Joint Position Command to JCB
Delta Scaling: Scale setpoint based on which joint is going to take the longest to achieve. Allows all joints to arrive simultaneously.
Velocity Scaling: 'Speed Ratio' describes the overall velocity scaling.
High Speed Scaling: High speed scaling reduces execution speed when commanded speed exceeds a high speed velocity threshold and the arm's high-speed collision links are in collision.
Collision Avoidance: Applies offsets to joint commands based on depth of intersection between arm collision geometries and the opposing arm or torso.
Joint Limit Clipping: If the joint command is beyond limits, clip the command to respect joint limits.
Collision Detection: If collision (impact) is detected, set position command to hold current compensating for the impact.

(a) Joint Position Control Mode. [4]


Joint Position Command In
Σ
Delta Max Clipping
Σ
Joint Limit Clipping
Σ
Collision Detection
Joint Position Command to JCB
Delta Max Clipping: The joint command will be clipped based on the delta max (offset from current position defined by max joint velocity)
Joint Limit Clipping: If the joint command is beyond limits, clip the command to respect joint limits.
Collision Detection: If collision (impact) is detected, set position command to hold current compensating for the impact.

(b) "Raw" Joint Position Control Mode. [4]

Figure 3.6: Comparison of Baxter control modes.

the robot's structure, the opposite problem, i.e., given an endpoint pose, calculate the configuration of the joints is not a trivial question. This is attributable to the fact that the relationship is not bijective, and, given an arm joint configuration there is only one possible endpoint pose, but given an endpoint pose there are several configurations of the joints that could accomplish that pose. This issue is caused by the redundancy provided by the 7 DOF of the arm. We can appreciate the different joints of the system from Figure 3.7.

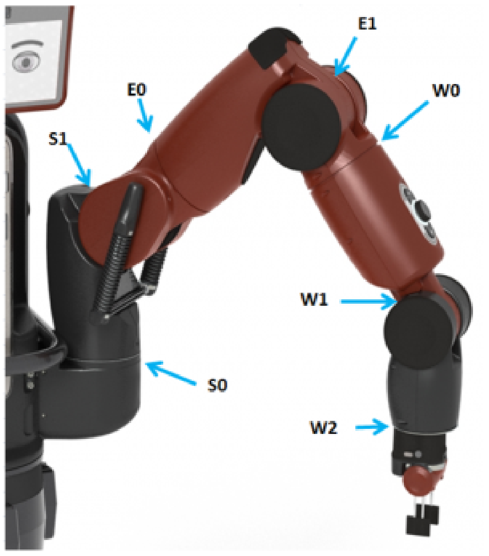


Figure 3.7: Names of the Baxter Robot joint names. [4]

To solve this, we will use the IK service of the robot, that, given an endpoint pose, returns a configuration of the joints that can accomplish it. This service, additionally, allows to input a list of joint seeds, so if we use it, the structure of the joints the IK service outputs will be as much similar as possible to the structure of some the input seeds. The manually defined seeds are important because using random seeds could lead to two potential problems:

- No solution is found starting from those random seeds.
- A solution is found, but the configuration of the arm is complex and it will not facilitate to continue the trajectory.

So, in order to convert the cartesian points p_i to angles j_i we will make iteratively requests to the IK service, but the same seeds will not be used during the whole conversion process:

- In the first iteration, a manually designed seed will be input to the service. This initial seed has to be designed by someone with sufficient knowledge about the robot, and it can be selected for example using the zero gravity mode of the arm. It has to be designed setting the initial position of the arm in such a pose that allows the trajectory to be completed without big changes in the arm structure.

- The rest of the iterations, to calculate the conversion of r_i we will use j_{i-1} as the seed.

Dynamic Movement Primitives

Once the records are represented as joint positions we can start the learning process. For the learning process we will use DMPs, a proposal coming from the dynamical approach rather than from the computational one. In detail, we will use an evolution of the original DMPs, described in [22], that allows to improve the control over a goal position different than the trajectory shows.

In DMPs, a movement is represented using non-linear differential equations. As any recorded movement can be represented using a set of differential equations, we can create a library of movements which could be concatenated in order to create more complex actions. They idea is that we do not have to demonstrate every possible movement, but instead, we demonstrate basic movements, creating basic DMP blocks, and then, we extrapolate the initial and final positions.

A one dimensional movement can be generated integrating the following differential equations, whose behavior is similar to a spring modified by an external force:

$$\tau \dot{v} = K(g - x) - Dv - K(g - x_0)s + Kf(s) \tag{3.20}$$

$$\tau \dot{x} = v \tag{3.21}$$

Where x is the position of the system, v the velocity, x_0 the starting position, g the goal position and τ a temporal scaling factor. K behaves like a spring constant and D is selected to allow the system to damp. Finally f is a non-linear forcing term that must be learnt to produce generalization. This term, f, can be defined as:

$$f(s) = \frac{\sum_i w_i \psi_i(s) s}{\sum_i \psi_i(s)} \tag{3.22}$$

Here, $\psi_i(s)$ are Gaussian basis functions with mean c_i and width h_i, and w_i are the weights to be optimized for learning. We can notice that f does not depend on time, instead, it depends on a phase variable s, which monotonically decreases from 1 to 0 during the execution:

$$\tau \dot{s} = -\alpha s \tag{3.23}$$

This system, where α is a user defined constant, is referred as the canonical system.

The equations defined above meet several properties: the convergence to the goal is guaranteed thanks to the canonical system; the weights can be optimized to learn every trajectory; and the movements are similar changing the start and goal points.

To learn from a recorded demonstration x(t) we must first calculate its derivatives $\dot{v}(t)$ and $\ddot{v}(t)$. Then, we must integrate the canonical system, computing $s(t)$ for a given τ. Later, a target forcing function $f_{target}(s)$ is calculated for every s according to:

$$f_{target}(s) = \frac{\tau\dot{v} + Dv}{K} - (g - x) + (g - x_0)s \tag{3.24}$$

Finally, we have to solve the weights in Equation 3.22 to minimize the cost function, which is the sum of the Euclidean distance between each pair of f and f_{target}:

$$J = \sum (f_{target}(s) - f(s))^2 \tag{3.25}$$

The optimization of the weights (w_i) can be posed as a linear regression problem, whose solution can be calculated using, for instance, least squares.

To explain the behavior of the system in a more abstract way, we can say that there is a spring-like mechanism that makes the system to converge from the start position to the attractor of the system, the end position; and a force function that learns the details of how it should converge. The canonical system is introduced to help the system to converge.

At generalization time, the previously calculated weights are reused, setting a new x_0 and g and declaring $s = 1$. Also, the canonical system has to be integrated, evaluating $s(t)$. In Figure 3.8 we can see how the canonical system drives the forcing function which modifies the trajectory to output the desired final values.

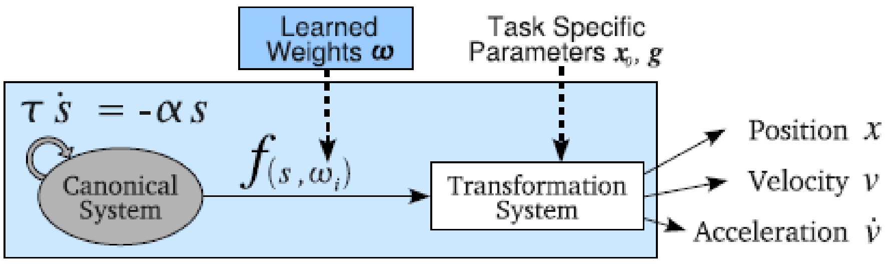


Figure 3.8: Canonical system driving the DMP. [22]

This schema is for a one dimensional DMP, but it can be extended for as many dimensions as desired, declaring several differential equations and forcing functions, but being cautious of linking all of them to the same canonical system. In our project, the angle position of each joint along time would be the input to the system.

Obstacle avoidance can be incorporated in this framework, by means of a vector specifying the position of the obstacle, and, modifying the differential equations. However, we will not incorporate it in our project as the implementation we are going to use does not dispose of it.

To develop the DMPs we have used the implementation published in ROS [23]. This implementation contains three functions:

- **learn_dmp_from_demo**: where you input the initial trajectory and introduce the hyperparameters.
- **set_active_dmp**: to set a DMP as active in the server.
- **get_dmp_plan**: to make a query to the server to generate a new plan given the learnt trajectory and the new goals.

So first, the records in joint space $\{j_0, .., j_{N-1}\}$ are passed through the *learn_dmp_from_demo*. Then the DMP is marked as active. And later a new start and final positions are introduced to create a new plan, using *get_dmp_plan*.

The start and final positions are specified in cartesian space, as a poses (position and orientation) and then converted to the joint space using the IK service. For the starting position and final positions, the seeds used are j_0 and j_{N-1} respectively.

After this, the learning process is completed, obtaining a trajectory of length M, $\{t_0, .., t_{M-1}\}$, which can be sent to the Baxter's joint trajectory server to be executed.

Experiments

In order to find out if the system created accomplishes its goals, two demonstrations will be used to test it. We will use demonstrations of trajectories which are not trivial and could not be replicated alone by the Baxter services unless they were manually programmed.

The first demonstration resembles to a pick and place movement. A lot of robots are programmed in the industry with aims to repeat this movement once and again, however, frequently, these robots do not have the capability of generalizing doing the same movement for new starting or ending positions. It is such a useful movement in a lot of environments that can be used in numerous situations. Specially, assistive operations often require to pick and place objects. This demonstration will start with the humans' hand in a low position, and will make an inverted V transition, first moving the hand to a left up direction and later lowering it to to the bottom left. The orientation of the hand only varies lightly.The idea is that this movement mimics a situation in which you want to move something from one position to another knowing that there will be an obstacle in the middle. This obstacle could be a little wall or a water tap, something that is always there and therefore must be always avoided.

It could be thought that this demonstration can be substituted by commanding the robot to go to three different positions, the initial one, one at the top of the curve, and one at the end, however, programming that would not guarantee us neither of these two things:

- The orientation of the endpoint is the one that we desire along the trajectory.
- The endpoint avoids the obstacle. This latter one is not guaranteed due to the fact that from one point to another, the robot services decide which path to follow, and nothing guarantees us the obstacle is in the middle.

The second demonstration, represents a situation with changes in position and orientation. It mimics a situation where a human has an object in a high position, then lowers it, moves it to the left, and finally turns the hand 90 degrees. This could be similar to a human taking some recipient with liquid in it, which wants to pour it inside the sink. It is very important to maintain the orientation of the hand when lowering the recipient to avoid dropping the content. This demonstration has been designed thinking on people with mobility problems that often need help objects which are in high positions, adding some complexity later with a manipulation of the object, to see the limits of our proposal.

To carry the experiments, N and D are set according to the ROS DMP [23] package documentation to make sure the spring system converges. The rest of the parameters are set manually by experimentation. They can be consulted in Tables A.1 and A.2. Parameter τ is set so as the generated plans have the same length than the original demonstrations.

For every demonstration we have measured the number of records per period that were discarded due to having a cosine of the angle lower than 0.95, we will measure the quality of the recorded data, and if the algorithm generalizes, querying new unseen positions.

The experiments have been accomplished first in the simulator and later on the real robot, however, we will only show demos on the simulator due to the inability to access the laboratory the two months previous to the presentation of this thesis.

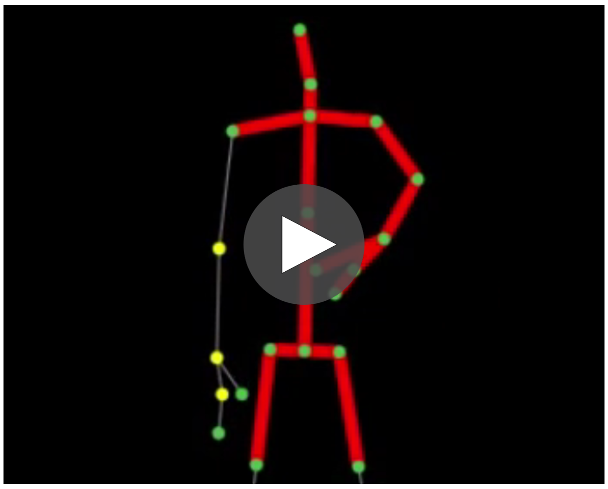
For both demonstrations, several takes are going to be recorded and after visual reviewing and revision of the different measures, the best take for each of both demonstrations will be selected for analysis.

Results

In this section we are going to explain the results obtained for the two recorded demonstrations. The different aspects defined in the previous section will be discussed.

5.1 First Demonstration

As we explained before, this movement has characteristics in common with the pick and place movement. On Video 5.1 we can watch the user performing the demonstration. Besides, in Figure 5.1 we can observe the evolution of the coordinates for every human joint. From this graph we can analyze several things. First, we can easily see that there are undesired pikes, above all in the X coordinates of the wrist, that we would like to avoid in the final movement. If we do not have into account this pikes, the group of the X coordinates is the one that has less variation. We can also see the inverted V structure on Z coordinates that we commented about when designing this demonstration. Last, we can perceive the increment on the Y coordinates along time.



Video 5.1: Kinect recording of the first demonstration.

As mentioned before, in order to smooth the trajectories and discard outliers, we preprocessed the data. First, we discarded those records whose corresponding cosine of the angle between vectors was higher than 0.95.

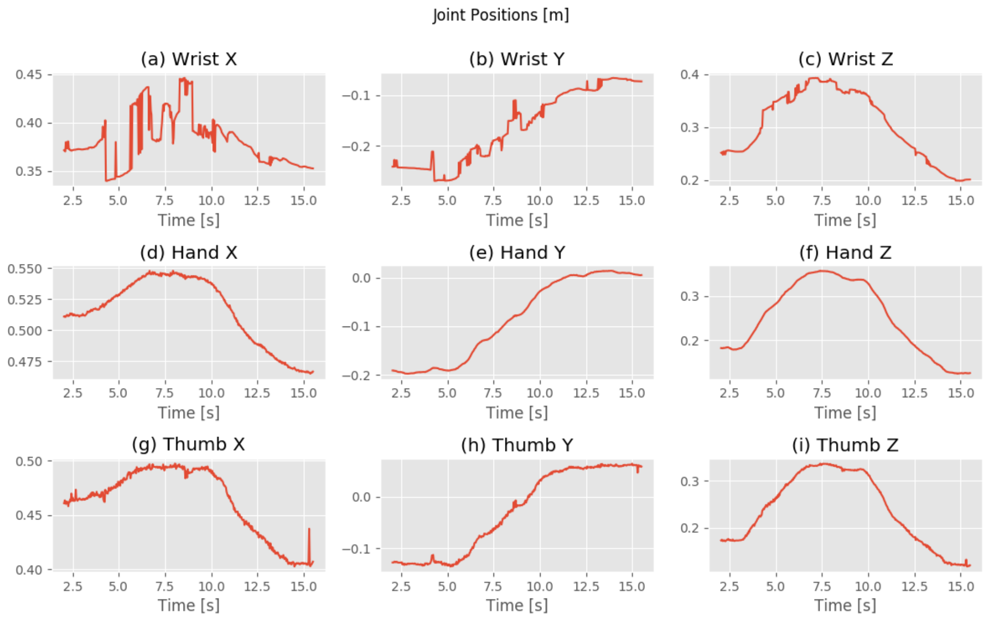


Figure 5.1: Coordinate positions of the joints for the first demonstration.

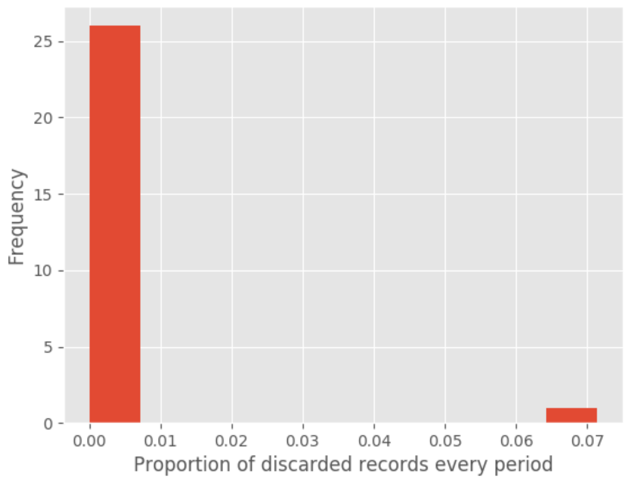


Figure 5.2: Histogram showing the proportion of records discarded by the cosine filter every period. Records taken at 30 Hz and grouped every 0.5 s.

In Figure 5.2 it can be seen how almost all the records were considered valid and only a small amount of them was removed, in one of periods of 0.5 seconds.

Also, in Figure 5.3, we can observe a histogram on the difference between the minimum and maximum coordinate values per every period. There, we can see, that, whereas in the majority of the time periods the distance between the records coordinates was minimal, there were some cases in which this distance was more noticeable. We can see, for example, that Y and Z coordinates suffer more from this issue, and how this difference sometimes reaches up to 6 cm. Additionally, in Figure 5.4 we can see the histogram on the standard deviation for every period. There, we can see that the standard deviation sometimes reaches the 3 cm, meaning that there is a lot of variability in the measures, but in most of the periods it is low.

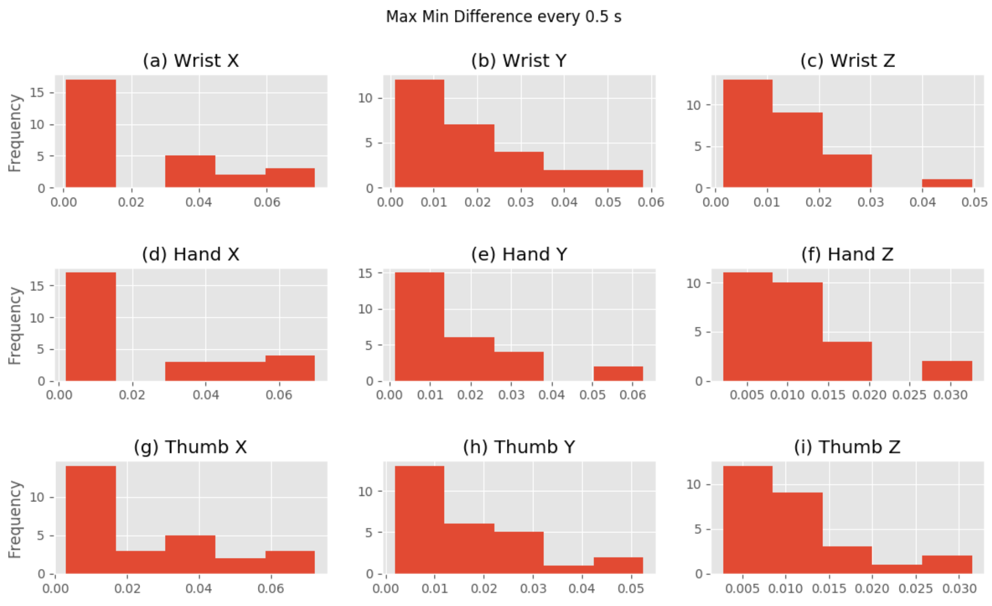


Figure 5.3: Histogram showing the difference between the coordinates at every period. Records taken at 30 Hz and grouped every 0.5 s.

After applying those filters, and doing the calculation of the orientation, the signals appearing in Figure 5.5 were obtained, showing the robot's endpoint pose. It can be perceived how the signals are now much smoother than before.

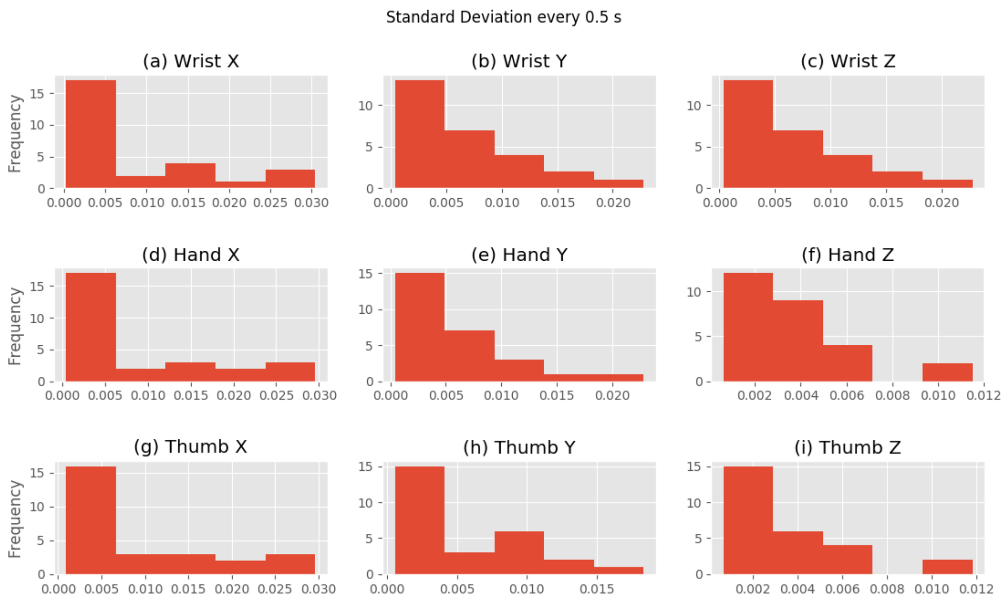


Figure 5.4: Histogram showing the standard deviation of the coordinates at every period. Records taken at 30 Hz and grouped every 0.5 s.

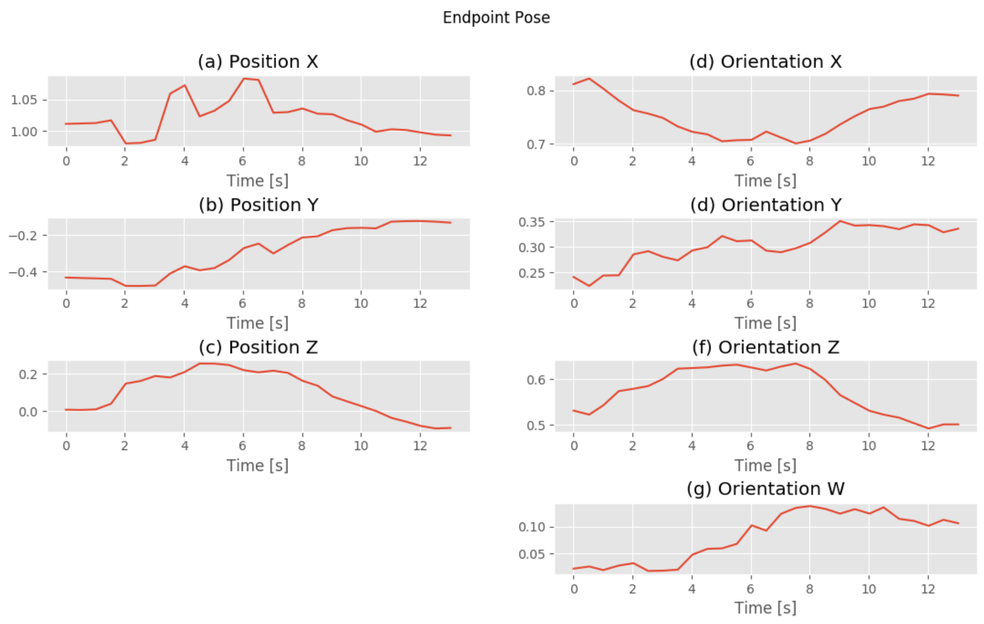
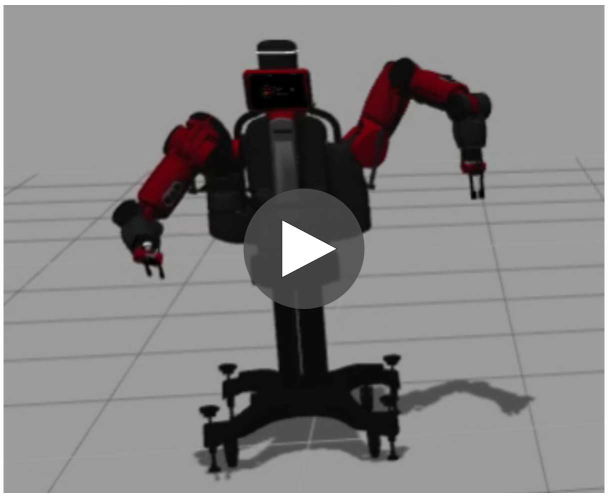


Figure 5.5: Pose as a position and an orientation.

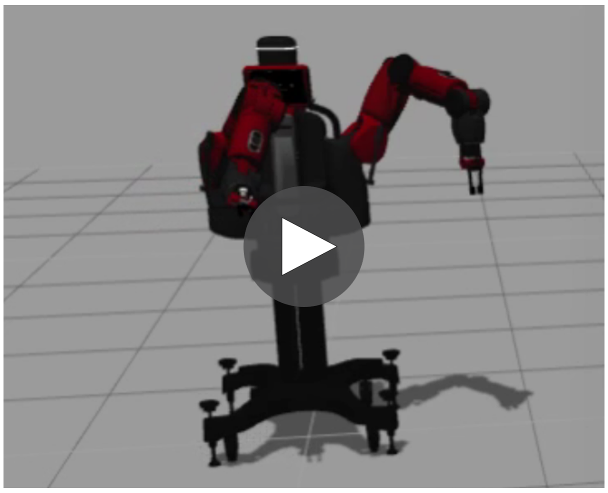
As a proof of concept, we will demand the system to learn the demonstrated trajectory, and to replicate it, in order to see if it is capable of mimicking it starting from the same start and end positions. From Video 5.2 and Figure 5.6 we can see that it is capable of accomplishing this simple goal obtaining a Mean Squared Error (MSE) with respect to the source (demonstrated) trajectory of 0.08, which is very low.



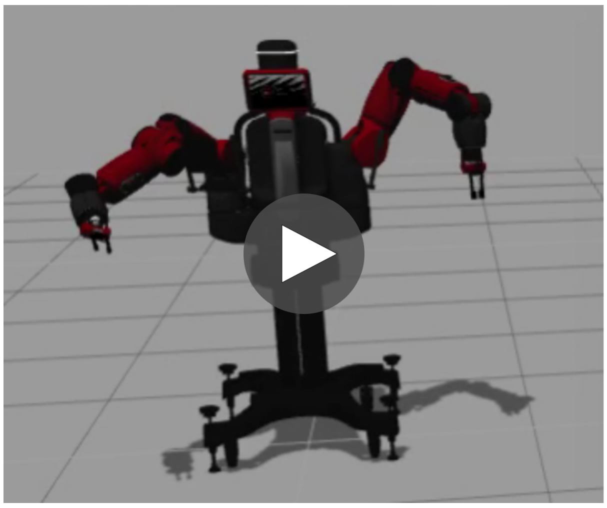
Video 5.2: Demonstration 1. Execution of the source movement.

The next stage, once we have seen it is capable of learning this trivial task, is to test if the system generalizes. We will first see if its capable of generalizing changing only the finishing state, to later modify both start and end states. As the movement is similar to a pick and place one, we applied some offsets so as the hand is near the table we had in the laboratory both in the starting and end positions. However, now, we will ask the system to end in a pose maintaining the final orientation but moving with the endpoint 17 cm to the left and 13 cm up.

We can see from Video 5.3 that the system succeeds at achieving its goal. From Figure 5.7 it is seen that, for some of the joints, at the beginning the curves are similar but as they approach to the end of the movement they diverge. However, other joints, as, for example, right_e1, follow a similar trajectory to the demonstrated movement.



Video 5.3: Demonstration 1. Execution of the plan associated to modifying the end pose.



Video 5.4: Demonstration 1. Execution of the plan associated to modifying the start and end poses.

Finally, we will ask the system for generalizing both starting and ending positions. To this end, we will use the same end pose as in the previous example. The start position, on the contrary, will be moved 20 cm to the left and 10 cm up, which is a substantial difference. Nevertheless, we can see from Video 5.4 that it is able to realize the new movement following a similar trajectory to the one learned. From Figure 5.8 we can perceive that this time the curves diverge in general in both the beginning and end of the movement.

Discussion on Demonstration 1

Now, we are going to analyze several important aspects deduced from the analysis of the results.

First, from the analysis of the histograms about the signal processing part, we can see that if we had not corrected those signals with the filters we applied, it would have leaded to an undesired behavior with the arm moving to opposing directions at every moment.

Thanks to the histograms, we can also see that the selection of the median filter, is an improvement over a mean filter, as the latter one would have modified the output through the outliers, even though most of the data is near the mean.

Even though we have shown examples in which the system works, there are some cases, the less, where it does not. This may be due to two causes:

- The system outputs a trajectory that is out of the limits of some of the joint's domain.
- The inputs to the system are very different to the learnt trajectories and, thus, it does not generalize well.

However, in the majority of tests executed the system performed well.

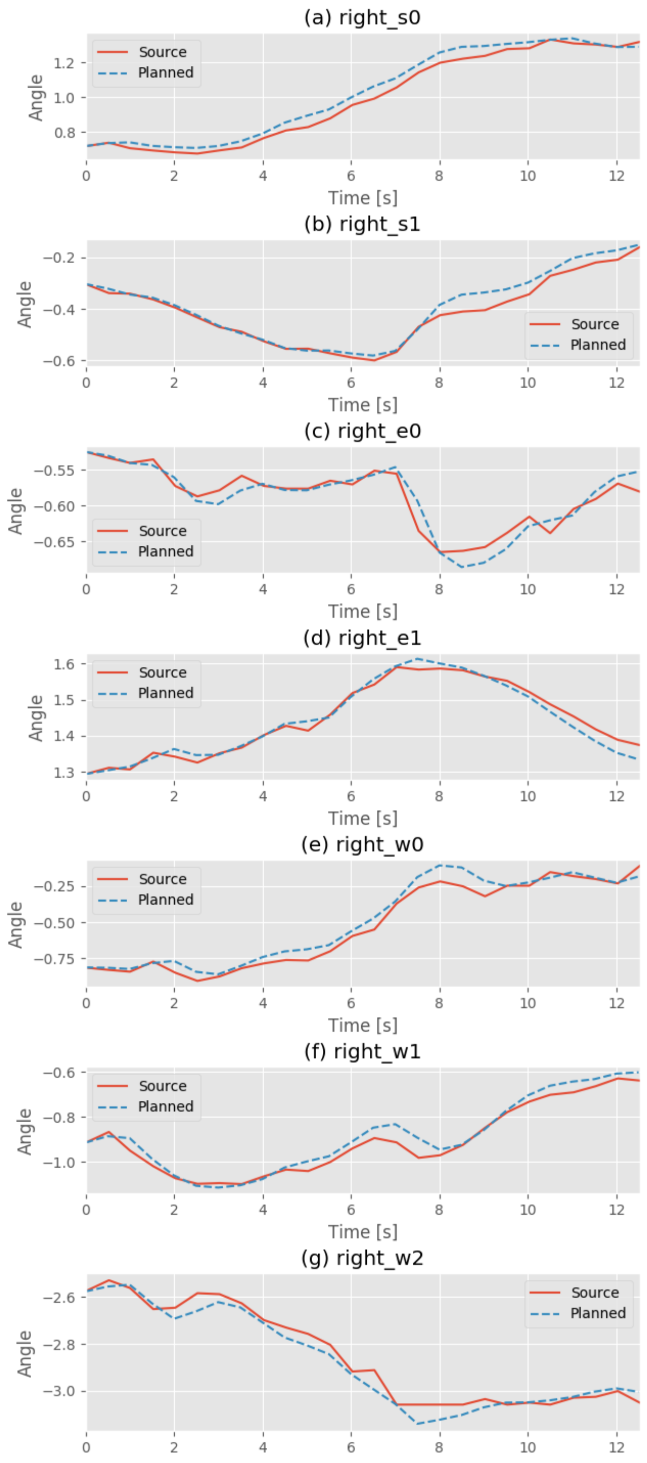


Figure 5.6: Comparison of source and planned trajectories.

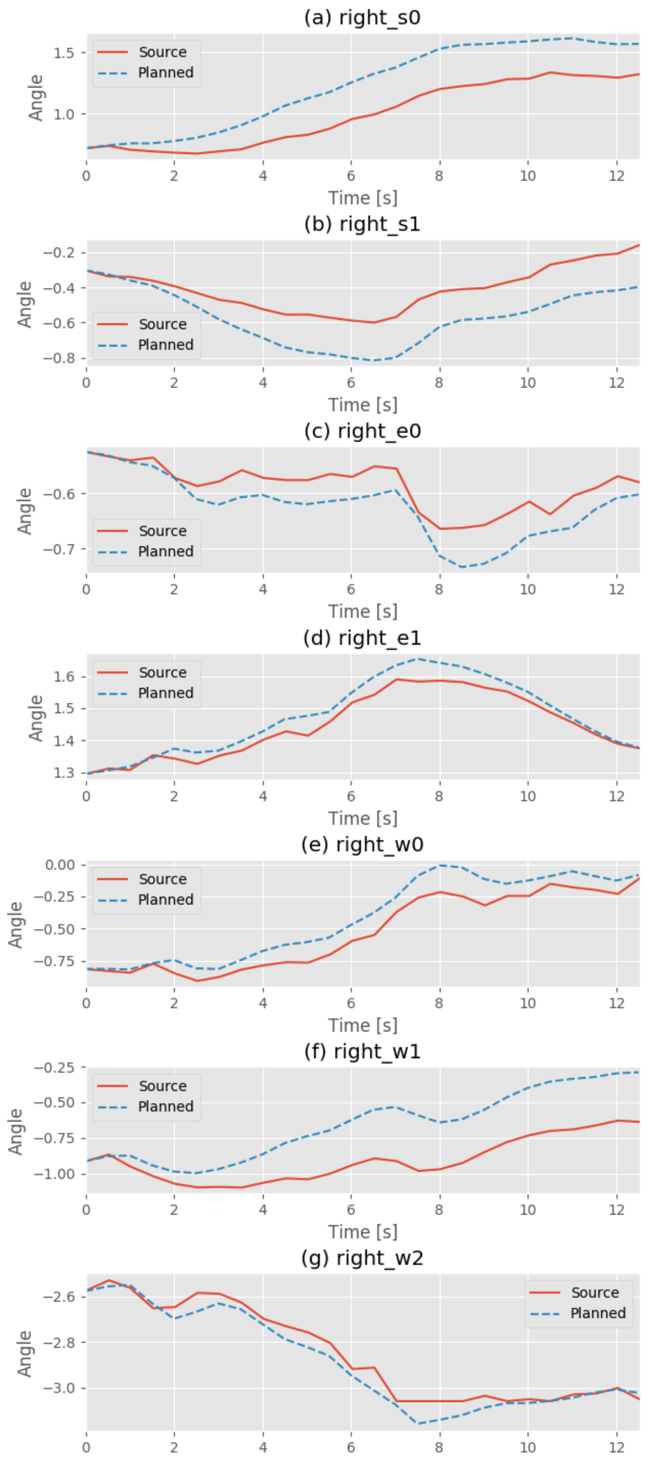


Figure 5.7: Planned trajectory modifying the end position.

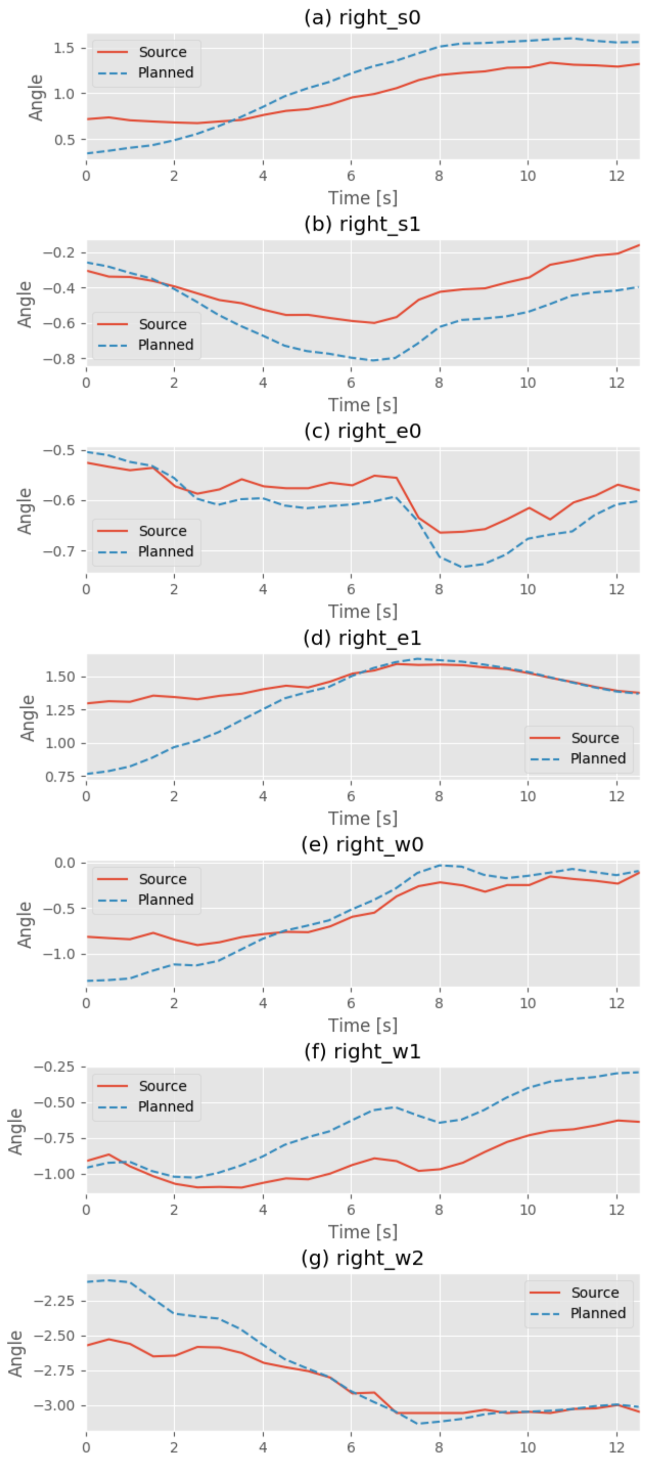


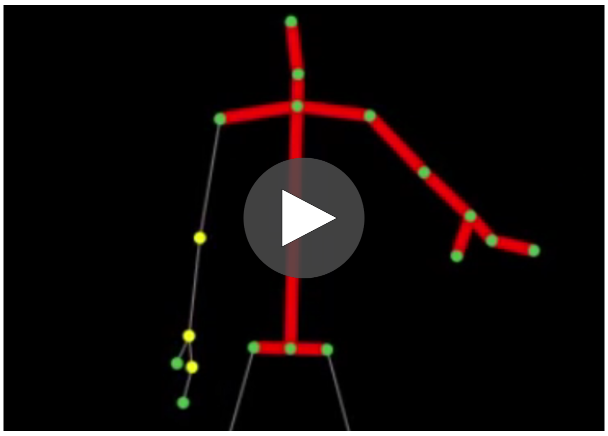
Figure 5.8: Planned trajectory modifying the start and end positions.

5.2 Second Demonstration

Again, for this demonstration, we will comment the different results obtained and the capability of generalization of our proposal.

Something remarkable for this demonstration is that, due to the fact that the Kinect camera was not sensing the thumb correctly at high number of frames, the camera position was changed several times and the demonstrations had to be done very carefully and slowly.

On Video 5.5 we can see the gestures recorded by the user, and, on Figure 5.9 we can observe the translation of this demonstration to coordinates for every joint.



Video 5.5: Kinect recording of the second demonstration.

It can be noticed again that the trajectories have numerous pikes, more pronounced in the X coordinate of the wrist, that could lead to problems if they were not processed properly. In fact, in this case the pikes are bigger than in the previous demonstration.

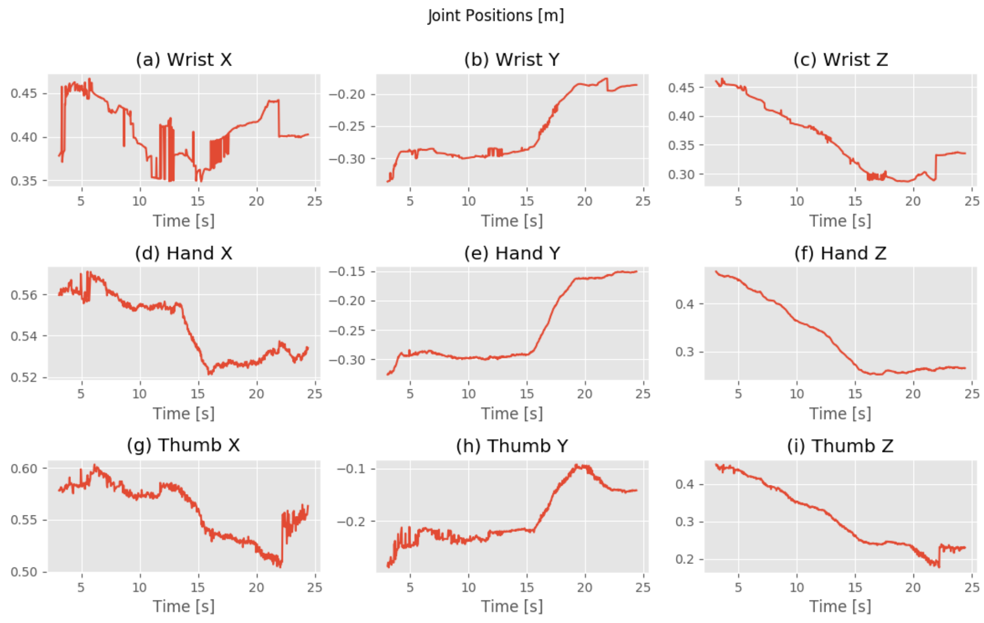


Figure 5.9: Coordinate positions of the joints for the second demonstration.

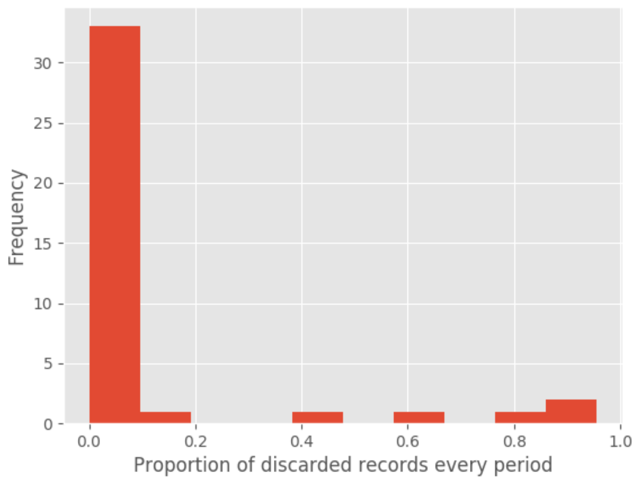


Figure 5.10: Histogram showing the proportion of records discarded by the cosine filter every period. Records taken at 30 Hz and grouped every 0.5 s.

On Figure 5.10, opposing to the behaviour of the previous demonstration, we can see how, in this case there are more periods having coordinates discarded, and in fact, it can be seen how in one of them, the ninety percent of the records did not meet the cosine rule we imposed before.

As for the quality of the records that pass this first filter, we can observe the histograms on Figures 5.11 and 5.12. It may be perceived how the maximum difference between coordinates of the same period can be sometimes up to 5 cm, observing that the standard deviation never surpasses the 2 cm, which is a high value. We can even see how, for the Y coordinate of the Thumb joint, which is very problematic, as stated before, the bin with higher frequency on the histogram is not the one with minimum standard deviation. However, as the demonstrations are being done slowly, most of the coordinates in the periods of 0.5 seconds are representative of the movement and therefore processing with the median filter succeeds if the camera sensed correctly.

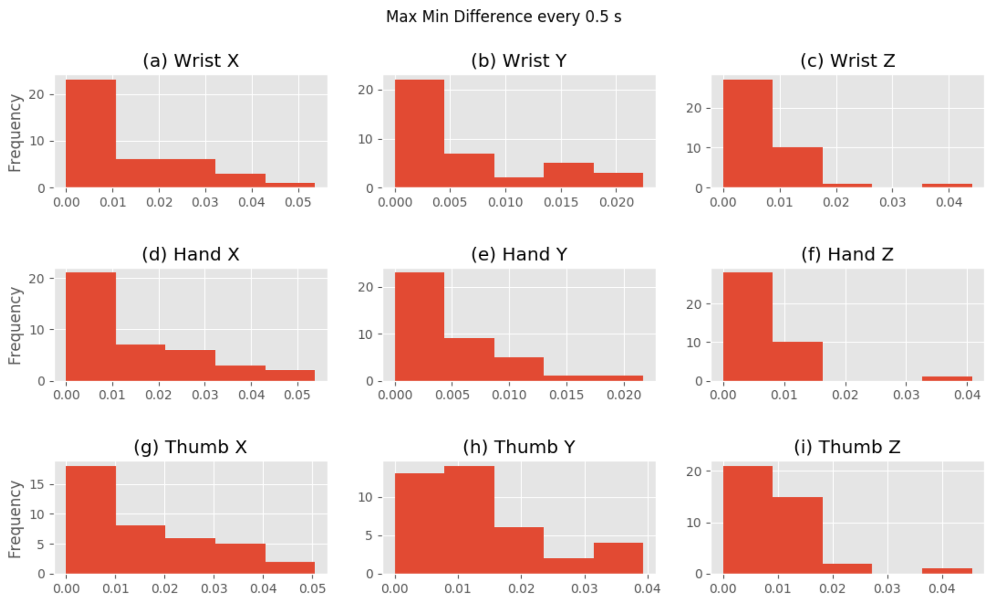


Figure 5.11: Histogram showing the difference between the coordinates at every period. Records taken at 30 Hz and grouped every 0.5 s.

The result after processing the signal with the commented filters, calculating the orientation and applying the final Gaussian filter results in a much smoother curve, as it can be appreciated from Figure 5.13. However, we can still find undersired peaks in the X Position coordinate, which does not seem to follow a natural trajectory, like the rest of the signals.

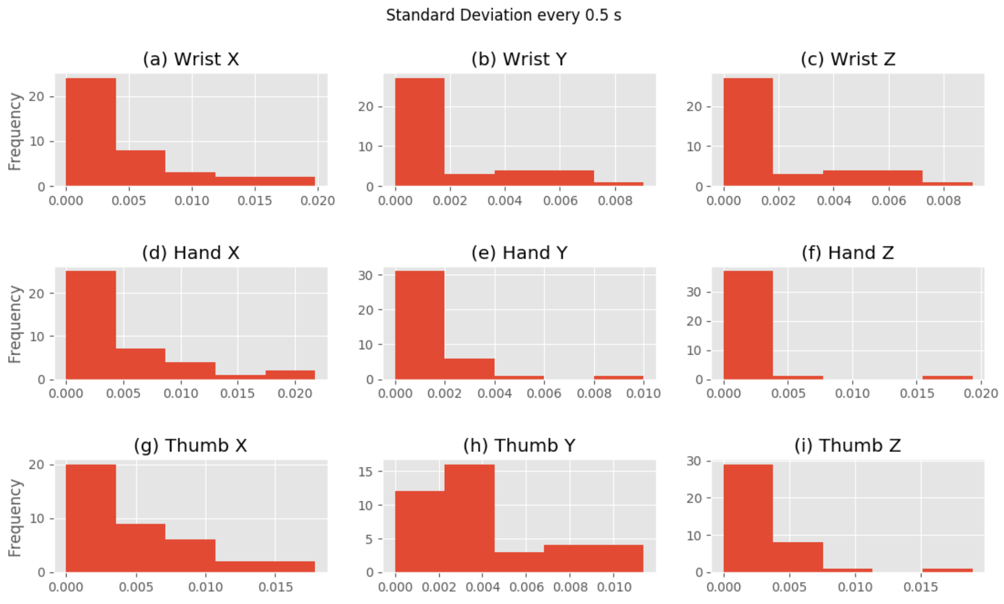


Figure 5.12: Histogram showing the standard deviation of the coordinates at every period. Records taken at 30 Hz and grouped every 0.5 s.

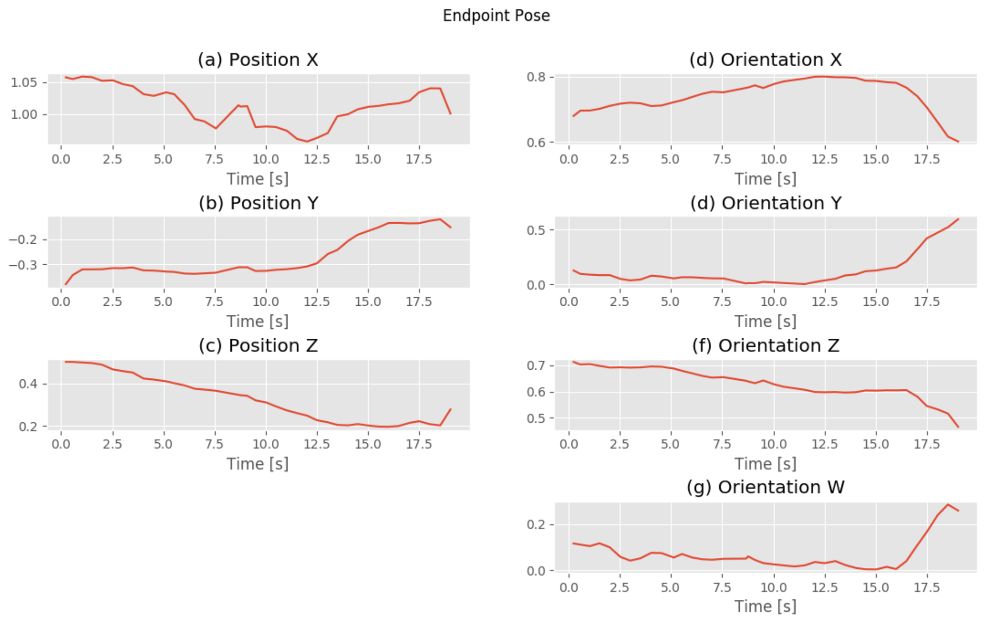
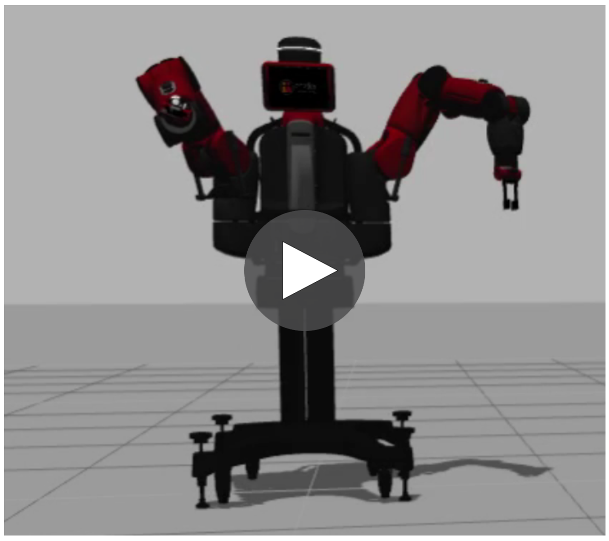


Figure 5.13: Pose as a position and an orientation.

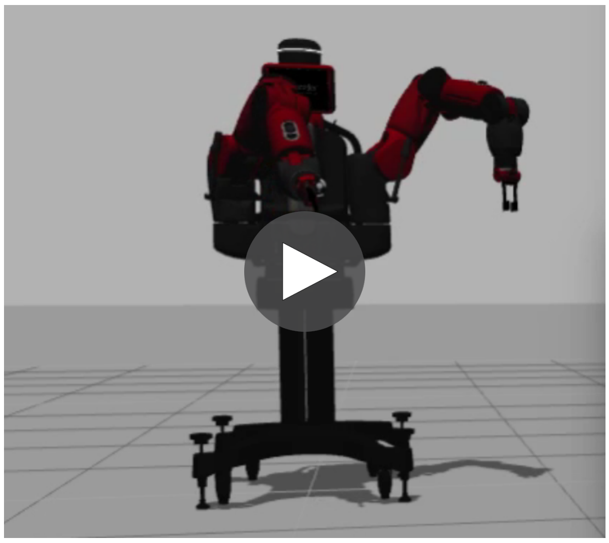
After analyzing the signals obtained and how they have been processed we are going to do several experiments to see how the algorithm works on the desired task. As in the first demonstration we will ask the system to learn the source trajectory, planning on starting and finishing in the same poses, after using the Inverse Kinematics system of the robot. As we can see from Video 5.6 the system accomplishes its objective. However, from 5.14 we can see that the planned trajectory needs some more seconds to achieve the goal. The MSE calculated only for the length of the source signal is just 0.075.



Video 5.6: Demonstration 2. Execution of the source movement.

Once we see that the system is able to learn the source movement, we are going to request the system to modify the end position. For this, we will ask the robot to position the endpoint 23 cm more to the left and 23 cm lower than in the source trajectory. As we can see from Video 5.7 the system accomplishes its objective. Also, in Figure 5.15, we can observe analogously to the previous demonstration how the source and planned signals follow similar trajectories at the beginning but diverge at the end of the movement.

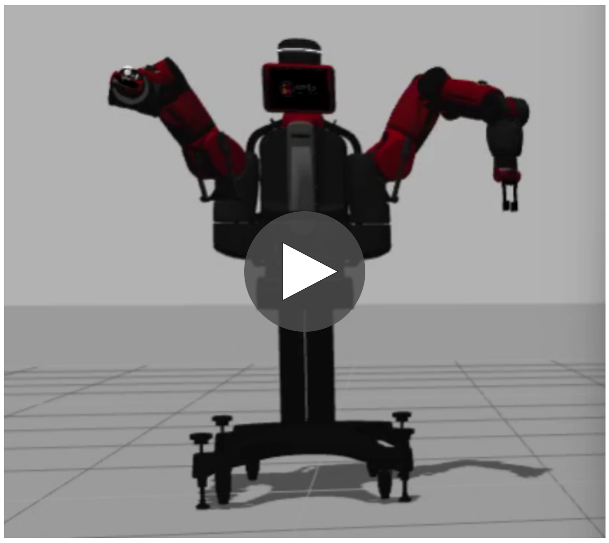
After this, we will ask the robot to modify the starting position as well. We will request the endpoint to start 15 cm more to the right and 10 cm upper than in the IK source signal. Again, the system achieves the execution of the trajectory, as recorded in Video 5.8. In Figure 5.16 we can see that the end



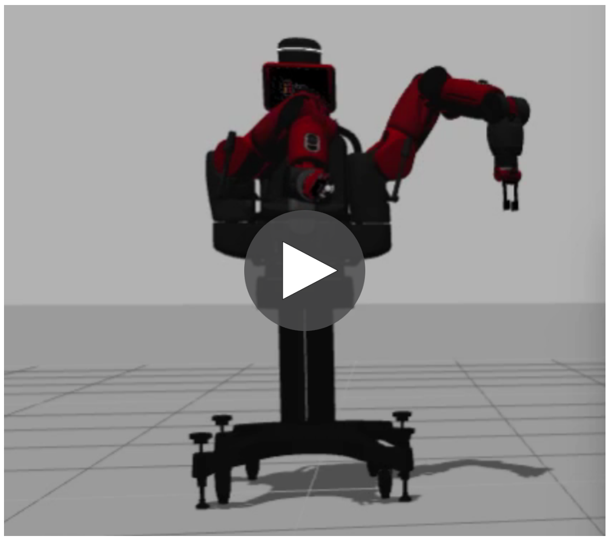
Video 5.7: Demonstration 2. Execution of the plan associated to modifying the end position.

of the movement is equal for some of the joints related to the wrist (right_w1 and right_w2), compared to the source signal, meaning that some joints are not affected by these changes. An important note is that, when requesting the system an increment in the starting pose to the right larger than 21 cm, the Inverse Kinematics system failed at disposing a joint configuration that could mach that pose.

Finally, an experiment maintaining the same start and end positions of the previous demonstration but requesting to modify the end orientation has been performed. Here, we have asked the system to end earlier the final twist of the hand, turning it only about 45 degrees instead of the 90 initially demonstrated. This time the DMP technique fails at doing what we expected from it. Even though the robot starts and ends in the pose we requested, we can observe from Video 5.9 that the robot starts turning the wrist earlier than desired, which would lead to dropping some liquid in the designed environment. This can be seen more exactly from joints right_w1 and right_w2 in Figure 5.17, where we would have expected the curves to be similar at the beginning and to start diverging just at the end of the movement.



Video 5.8: Demonstration 2. Execution of the plan associated to modifying the start and end positions.



Video 5.9: Demonstration 2. Execution of the plan associated to modifying the start and end positions besides the end orientation.

5.3 Discussion on Demonstration 2

Again we are going to comment some interesting aspects related to the demonstration. When doing the experiment planning the learned trajectory, we could see that the planned signal was larger than the original one. We think that this may be due to the fact that some source signals have a sudden change in trajectory at the end, and this may difficult the learning process.

Additionally, we have seen that the system has an unexpected behavior at orientation generalization time. One reason for this behaviour could be the mismatch between indicating the poses in Cartesian trajectories and then learning in robot joint trajectories. We are first indicating the endpoint end pose, which later has to be converted to a robot joints configuration. Even if we are using the final end pose of the source trajectory as a seed to calculate this joints configuration with the IK system, an small change in the pose may result in a bigger change in the joints, which might lead to indicating the DMP system to start modifying the trajectory earlier to accomplish the objective.

With the purpose of improving this orientation generalization, and thanks to the modularity of the implementation, a system was developed where the DMPs, instead, learned in Cartesian space (position and orientation), and the conversion to joints was performed posteriorly. However, when asking the IK service to convert the trajectory generated by the DMP system, it stated that not joint configuration was found meeting the trajectory requirements.

This, indeed, may be due to the fact that, opposing to previously, where the DMPs were learnt in the joint space, implicitly giving information of which is the domain of every joint, now, we are asking the system to learn in a virtual space that the robot is unaware of. And thus, leads to generating plans whose joint positions are out of the domain specified by the Baxter's manufacturer.

As a solution, we propose learning in robot joints but splitting this movement into two simpler and more primitive ones: a first movement doing the translation, and a second one doing the change in the orientation. Chaining those two DMP blocks whenever requested, following the principles explained in subsection 3.3 and in [22]. We think that, this way, the movement would have no problem to generalize the position, as tested previously, and it would be easier for it to learn the change of orientation, which would start for sure at the end of the change of position, due to the chaining process, solving the problem.

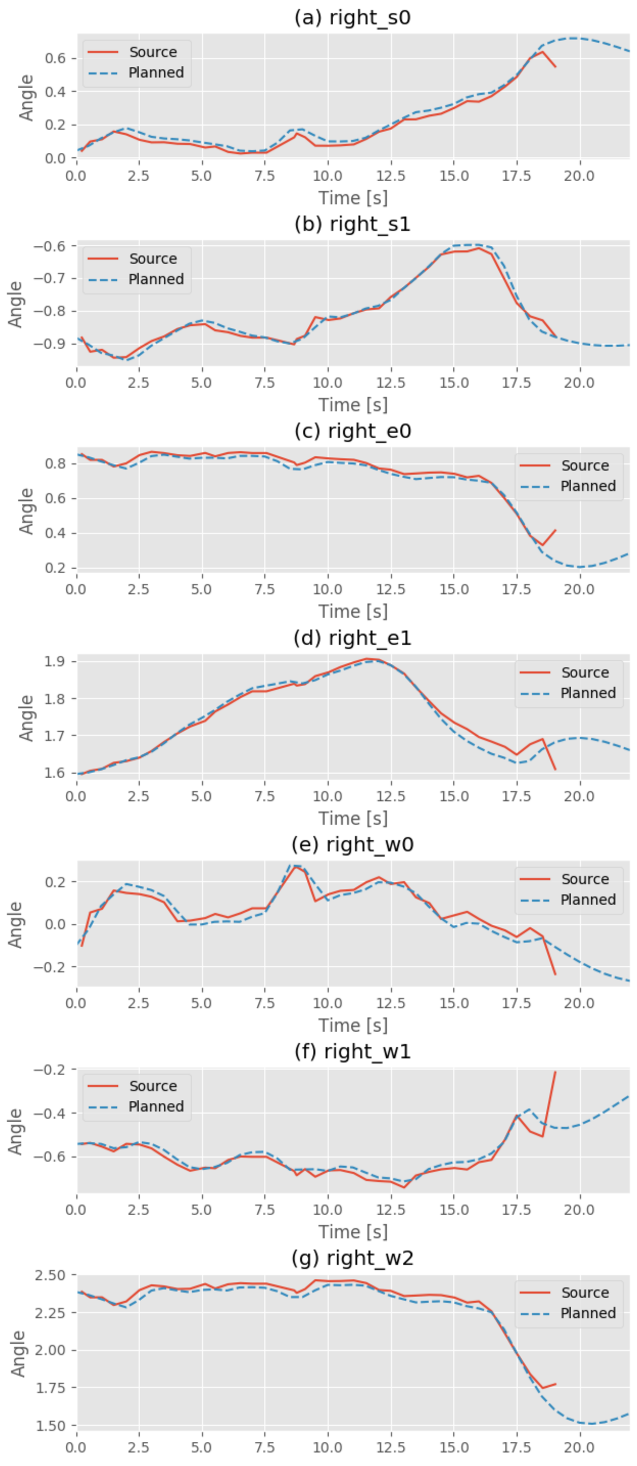


Figure 5.14: Comparison of source and planned trajectories.

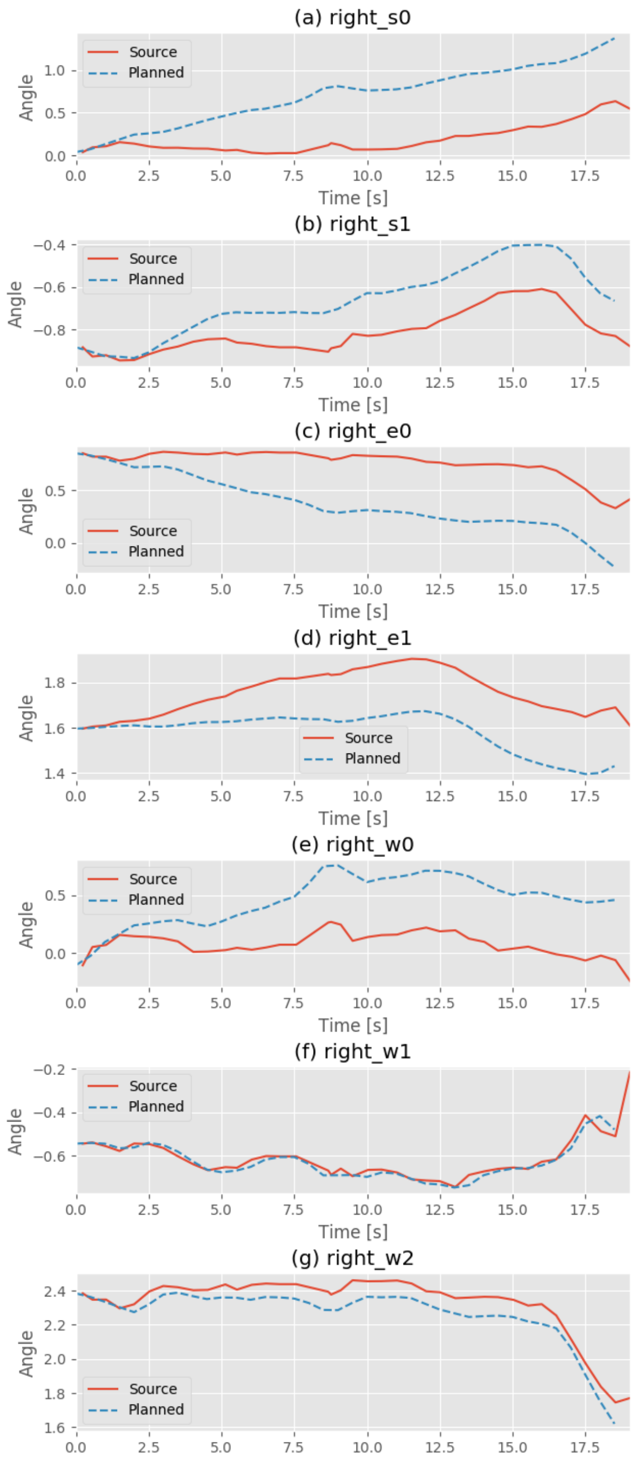


Figure 5.15: Planned trajectory modifying the end position.

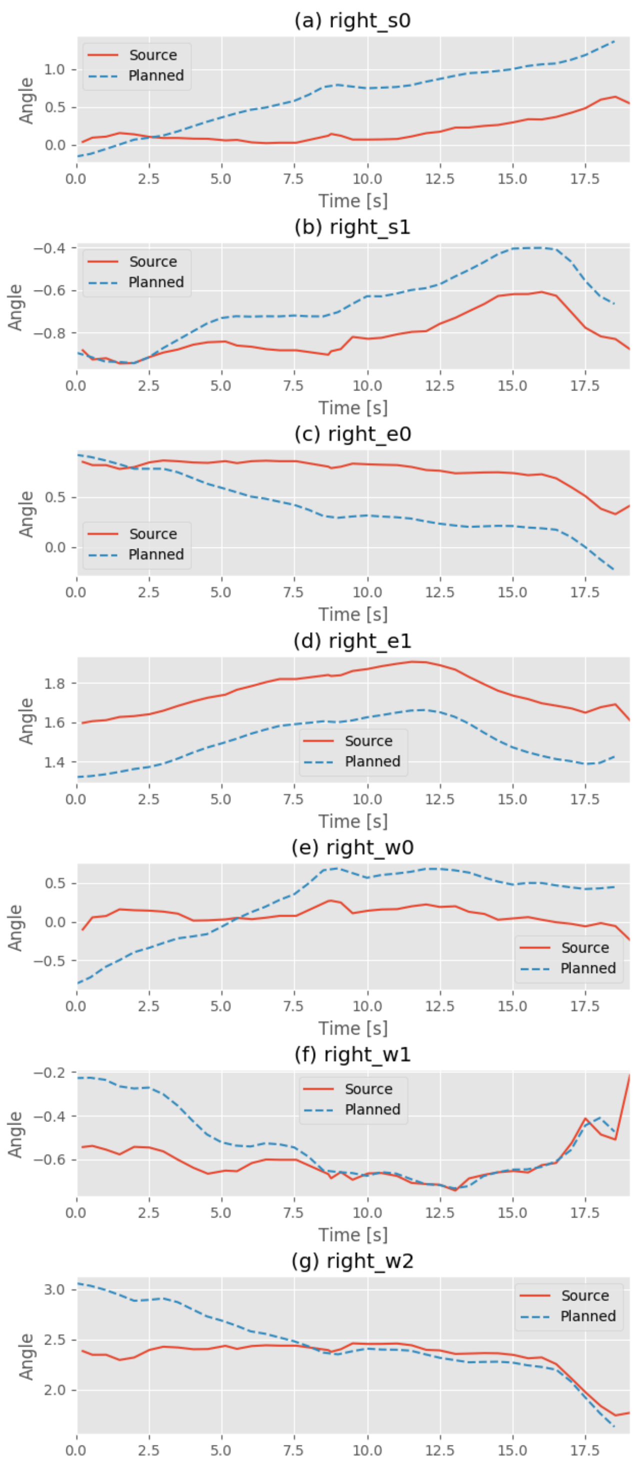


Figure 5.16: Planned trajectory modifying the start and end position.

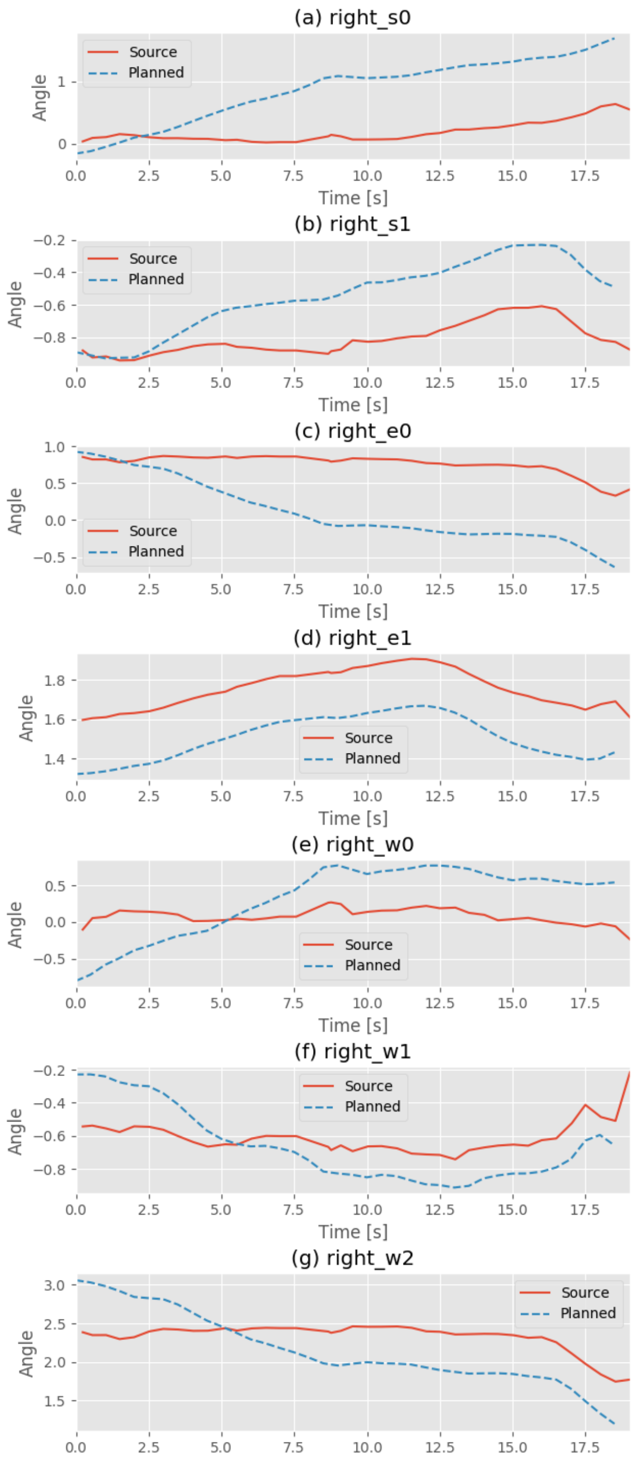


Figure 5.17: Planned trajectory modifying start position and end position and orientation.

Conclusions

We have been able to create a system that detects the joints of a human instructor, and, we have also been capable of making a change of reference systems with the origin centered in the robot's base applying simple algebraic operations. Besides, we have observed the importance of having a good sensor that minimizes the errors produced, and, in particular, the relevance of processing the signals to treat those errors, which are often unavoidable.

Additionally, we have been able to calculate the orientation of the hand and to format that information to feed the robot's Inverse Kinematics system, so as to position the robot's endpoint. We can see how the robot takes profit of feeding the IK system with the seed of the previous joint's configuration, leading to smooth trajectories, and we must emphasize the importance of choosing a good starting seed to start the process. We can appreciate how the system created is capable of mimicking the demonstrator's pose of the hand at every moment.

It is important to note that even though we can incorporate the experience of non-technical users into the system, we must always have a technician familiarized with the robot mechanics deciding a good joint's seed to ease the generalization process. However, this is totally agnostic to the human demonstrator.

As for the general system, including the DMPs, we have seen how it is able to generalize the movement of the endpoint to new unseen starting and ending positions for both of the non-trivial tested scenarios in most of the cases. We can state that Learning from Demonstration proposals are a good approach to incorporate human's knowledge into a robotic system giving the additional possibility to generalizing to unseen scenarios.

We have seen the system has some problems generalizing to new orientations in complex trajectories, but we think that decomposing these trajectories into more simple DMP blocks to later concatenate them may help to solve this problem.

Future Work

During this work we have developed only a small part of what an assistive robot needs to help a human. There are different lines of work that can be followed in order to improve the robot's functionalities.

First, we think that it is important to create a system that analyzes and feeds the DMP system with the initial and final positions. To this end, for the initial position Computer Vision algorithms could be used to detect the location of objects requested by the user in the Cartesian space. The final position could be taken from a list of predetermined positions that the user could query.

Additionally, it would be compelling to study other sensing systems that allow more precise joints detection to avoid the undesired peaks we had on the curves.

It would be very interesting as well to develop an implementation of the DMPs including the obstacle avoidance feature, as in [22], which could make use of Computer Vision tools to detect those obstacles.

Also, it is very important to create a sufficient number of demonstrations to build a library of DMP movements, complex enough to recreate any movement. It is even more important to make a study of the needs of a person with mobility problems to create that library and to make the proper aggregation of basic movements into more complex ones. For the system to be effective, it would need some kind of interface, for example a voice interface, that could translate the needs of the user into final actions of the robot. This could be developed combining Natural Language Processing solutions with Supervised Learning techniques to classify those needs into specific movements.

Finally, it would be worth to study the combination of LfD approaches with RL ones to increase the generalization skills of the systems, as done in [24]. In this paper they introduce PI^2 to their scheme, and we think that this addition could improve the generalization skills of our proposal considerably.

References

[1] Jens Kober, J. Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32:1238–1274, 09 2013. doi: 10.1177/0278364913495721.

[2] Stefan Schaal. *Dynamic Movement Primitives -A Framework for Motor Control in Humans and Humanoid Robotics*, pages 261–280. Springer Tokyo, Tokyo, 2006. ISBN 978-4-431-31381-6. doi: 10.1007/4-431-31381-8_23. URL https://doi.org/10.1007/4-431-31381-8_23.

[3] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, Feb 2013. ISSN 0899-7667. doi: 10.1162/NECO_a_00393.

[4] Rethink Robotics. Baxter research robot wiki. https://github.com/ogroth/tf-gqn, 2015. Online; Acessed 14/03/2020.

[5] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis. Optimal and autonomous control using reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 29(6):2042–2062, June 2018. ISSN 2162-2388. doi: 10.1109/TNNLS.2017.2773458.

[6] B. Kiumarsi, H. Modares, and F.L. Lewis. Chapter five - optimal tracking control of uncertain systems: On-policy and off-policy reinforcement learning approaches. In Kyriakos G. Vamvoudakis and Sarangapani Jagannathan, editors, *Control of Complex Systems*, pages 165 – 186. Butterworth-Heinemann, 2016. ISBN 978-0-12-805246-4. doi: https://doi.org/10.1016/B978-0-12-805246-4.00005-7. URL http://www.sciencedirect.com/science/article/pii/B9780128052464000057.

[7] Tao Bian, Yu Jiang, and Zhong-Ping Jiang. Adaptive dynamic programming and optimal control of nonlinear nonaffine systems. *Automatica*, 50(10):2624 – 2632, 2014. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2014.08.023. URL http://www.sciencedirect.com/science/article/pii/S0005109814003392.

[8] Bahare Kiumarsi, Wei Kang, and Frank Lewis. H∞ control of nonaffine aerial systems using off-policy reinforcement learning. *Unmanned Systems*, 04:1–10, 02 2016. doi: 10.1142/S2301385016400069.

[9] Zuyuan Zhu and Huosheng Hu. Robot learning from demonstration in robotic assembly: A survey. *Robotics*, 7:17, 2018.

[10] C. Li, C. Yang, J. Wan, A. Annamalai, and A. Cangelosi. Neural learning and kalman filtering enhanced teaching by demonstration for a baxter robot. In *2017 23rd International Conference on Automation and Computing (ICAC)*, pages 1–6, Sep. 2017. doi: 10.23919/IConAC.2017.8081985.

[11] C. Chen, C. Yang, C. Zeng, N. Wang, and Z. Li. Robot learning from multiple demonstrations with dynamic movement primitive. In *2017 2nd International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 523–528, Aug 2017. doi: 10.1109/ICARM.2017.8273217.

[12] Ning Wang, Chuize Chen, and Chenguang Yang. A robot learning framework based on adaptive admittance control and generalizable motion modeling with neural network controller. *Neurocomputing*, 2019. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2019.04.100. URL http://www.sciencedirect.com/science/article/pii/S0925231219314432.

[13] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469 – 483, 2009. ISSN 0921-8890.

doi: https://doi.org/10.1016/j.robot.2008.10.024. URL http://www.sciencedirect.com/science/article/pii/S0921889008001772.

[14] K. Dautenhahn and C. L. Nehaniv. *The Agent-Based Perspective on Imitation*, pages 1–40. MIT Press, 2002.

[15] Microsoft. Kinect for windows site. https://developer.microsoft.com/en-us/windows/kinect/, 2012. Online; Acessed 14/03/2020.

[16] Cao Wenming, Jianqi Zhong, Guitao Cao, and Zhiquan He. Physiological function assessment based on kinect v2. *IEEE Access*, PP:1–1, 07 2019. doi: 10.1109/ACCESS.2019.2932101.

[17] Microsoft. Kinect for windows sdk 2.0. https://www.microsoft.com/en-us/download/details.aspx?id=44561, 2014. Online; Acessed 15/03/2020.

[18] Lisa Jamhoury. Understanding kinect v2 joints and coordinate system. https://medium.com/@lisajamhoury/understanding-kinect-v2-joints-and-coordinate-system-4f4b90b9df16, 2018. Online; Acessed 15/03/2020.

[19] Dong Han, Hong Nie, Jinbao Chen, and Meng Chen. Optimal randomized path planning for redundant manipulators based on memory-goal-biasing. *International Journal of Advanced Robotic Systems*, 15: 172988141878704, 07 2018. doi: 10.1177/1729881418787049.

[20] Antonio Barrientos. *Fundamentos de Robótica*. McGraw-Hill, 01 2007.

[21] Andrea Alaimo, V. Artale, C. Milazzo, and Angela Ricciardello. Comparison between euler and quaternion parametrization in uav dynamics. *AIP Conference Proceedings*, 1558:1228–1231, 10 2013. doi: 10.1063/1.4825732.

[22] 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, May 2009. doi: 10.1109/ROBOT.2009.5152385.

[23] Scott Niekum. DMPs for ROS. http://wiki.ros.org/dmp, 2015. Online; Acessed 20/03/2020.

[24] Y. Yuan, Z. Li, T. Zhao, and D. Gan. Dmp-based motion generation for a walking exoskeleton robot using reinforcement learning. *IEEE Transactions on Industrial Electronics*, 67(5):3830–3839, 2020.

Parameters

Parameter	**Value**
K	100
D	$\sqrt{}(100)$
Basis Number	7
Width	logn(1000)
Alpha	10

Table A.1: Parameters for demonstration 1.

Parameter	**Value**
K	100
D	$\sqrt{}(100)$
Basis Number	13
Width	logn(1000)
Alpha	10

Table A.2: Parameters for demonstration 2.