Trabajo Fin de Máster en Robótica y Automática

Santiago Andres Solis Paiva

Escuela Técnica Superior de Ingenieros Industriales (UPM)

Master in Robotics and Automation

Learning system for compliant autonomous tasks in harsh environments

Author:
Santiago Andres Solis Paiva

Supervisors:
Manuel Ferre Perez
Mario Di Castro

July 2, 2018

Contents

Abstract

CERN's work-line is deeply related to harsh environments where radiation is its main concern towards the people working there. Therefore, a robotics team was founded to find solutions to situation where the environment is too dangerous for people to intervene. The purpose of this work is to achieve an autonomous system capable of performing repetitive tasks in structured environments that are considered harsh for humans to perform activities. Furthermore, the equipment in the surroundings are of high importance and cost, which requires the system to be complaint so that none of the equipment are damaged in case of unexpected circumstances. The integration of an online modifiable trajectory generation system known as Dynamic Movement Primitives (DMP) is used to perform specific tasks. Kinematic demonstrations along with locally weighted regression (LWR) are used to learn the trajectories and implemented in the system. A compliant behaviour is achieved through the modification of the DMPs canonical system and use of an impedance controller.

Acknowledgements

I would like to thank my UPM supervisor Manuel Ferre, for his help in making this opportunity happen. Also, words of appreciation to my CERN supervisor Mario Di Castro, for giving me also a chance to work in CERN, for his input and guidance into throughout my work.

Thanks to the robotics team of 628, who has been so helpful with any questions that I had. Giacomo Lunghi and Alessandro Mosca, for their help in my times of doubt. For Jorge Camarero, Carlos Veiga and David Blanco for being awesome. Thanks to Mengping Zheng for being a friend who I had not expected to have in this journey.

Finally to my parents, whom gave me the opportunity to start this master's degree and let me work on something that I love.

Chapter 1

Introduction

In this chapter, an introduction to the CERN's tunnels working environment is going to be presented, in particular the area where robot systems need to be used. Furthermore, based on a project undergoing at the organization, a description of the motivation behind this work is given. How the current situation is in its state and the work to be accomplished. Additionally, a small showcase of the current robot systems at CERN are introduced as well. Finally, the structure of this thesis is presented.

1.1 CERN and robot interventions

This work was developed in the European Organization for Nuclear Research most commonly known as CERN. One of the biggest international organization dedicated to the research of particle physics located in the north-west suburb on the Franco-Swiss border, where currently 22 member countries contribute to the high-energy physics research.

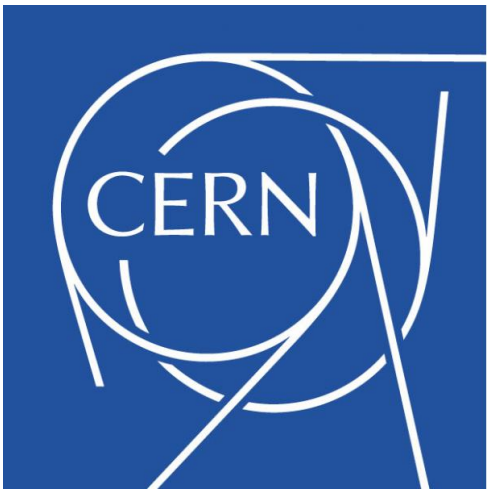

Figure 1.1: CERN's Logo

An important workplace at CERN is the tunnels that contain the different accelerators systems, such as LHC; one of the biggest accelerators known to man. These places are considered harsh environments, where radiation can reach high levels making human intervention very limited due to the negative effects it can produce in the workers' health.

For that reason, the robotics team is involved in projects were robots are used instead of humans to perform tasks at the tunnels. These are known as interventions and are usually

Figure 1.2: LHC's tunnel environment

performed in a tele-operated way with supervision of human operators. This proves to be a challenge since the tunnel's environment is not structured for robotic solutions and its possible to affect the high-performance devices that surround the area.

The different tasks that are performed by the remotely operated robots in the CERN's acceleration tunnels can be separated into three main lines.

- Remote visual inspection: By using visual feedback near the radioactive areas, the devices and tunnel are assessed by operators and to check if everything is not out of the ordinary.
- Remote inspection through sensors: By using specific robotic systems designed at CERN, the tunnels are inspected by measurements of radiation, temperature, and other signals which can give an overview of the environment in the area.
- Remote tools manipulation: Tasks that are in need of direct physical intervention and tools are used for manipulating devices or structures in the tunnels.

1.2 Thesis Motivation

In these harsh condition environments, robots are expected to assists operators in the best possible way to accomplish a defined task. Nevertheless, the robot and operator's performance can be affected due to unexpected structures in the area. Therefore it is highly desirable to have a versatile system which can adapt to these events and accomplish an specified objective.

In order to create an adaptable system, machine learning is an important component to take into account. Pre-programming a robot so it can learn different skills in an ever changing environment is impractical, since it is not easy to predict every possible scenario. Therefore, learning algorithms are tools that can be used to imitate skills and adapt to changing situations.

Furthermore, it is desirable to explore and incorporate an autonomous trajectory generation framework to perform specific tasks that can also be assimilated with different projects being developed at CERN; involving tele-robotics and computer vision. A system

to use with robots which can be robust, adaptable and capable of imitating paths through simple, yet effective, learning methods.

An approach to obtain information and learn new skills for an autonomous system can often be done by other software. Simulations are applicable in many different cases but as various conditions arise, the complexity becomes unmeasurable and it might not be the most adaptable approach. In our case, an operator familiar with the desired task is the best candidate to obtain information for learning. Therefore, an approach which involves an interaction with the operator is a suitable method for a robotic system to learn from.

Furthermore, due to the many conditions in which every task is subject to, it is not unforeseen that a robot task can fail in its execution. This can result in collisions with obstacles and devices or equipment which can create costly damages. Mistakes can happen due to unexpected events. Let it be an obstacle which was not considered in the task planning or a misjudgement of the structure near the robot's working area. Therefore, it is desirable to minimize the reaction if this were to happen.

From these points, two main elements are desired to be accomplished in this work. One is the incorporation of a learning framework that imitates desired trajectories for specific tasks. An element which will be addressed in this proposed learning framework by using kinesthetic training method through the defined hand-guidance mode. And to explore an adaptable path generation architecture that can perform automated tasks under harsh environments without damaging equipment that may come in contact with the robot. A feature achievable with the use of Dynamic Movement Primitives (DMP) framework in the proposed autonomous mode.

1.2.1 Remote Handling in Harsh Environments

In a simple and straightforward sense, the accelerator's purpose is to study the behaviour of particles when they collide into each other or into a stationary target. When this occurs, the reaction of such collision generates radiation in the environment. As a consequence, the surrounding objects are inflicted with these particles and they become radioactive. These areas cannot be approached without the proper gears and measurements for safety reasons.

High magnetic fields, oxygen deficiency and radiation are the main core hazards which create the harsh environment in the accelerator's tunnels. Nevertheless, in order to have a the best results of the measurements needed for the different types of research, the devices in which the CERN relies on must be maintained. This means that even though they are located in harsh environment, the devices need to be accessed in a safe way.

In the accelerators, an important device that needs to be used, are the collimators. Their design was meant to protect the integrity of the accelerators and their purpose is to constrain the high-energy particles (Burkart (2012)). They serve as a beam "cleaner", where all the stray particles are disposed for the accelerator's safety. This is due to the high energy that is involved in the experiments which requires high precision and control.

The maintenance of a collimator is not trivial as it is located in a very radioactive area. Furthermore, the full collimation system is extremely vast, as it consist of 84 two-sided movable collimators of different designs and materials with over 390 degrees-of-freedom. Some of the responsibility in maintaining these devices fall into the robotics team, which uses a tele-operated robot system near the radioactive sources.

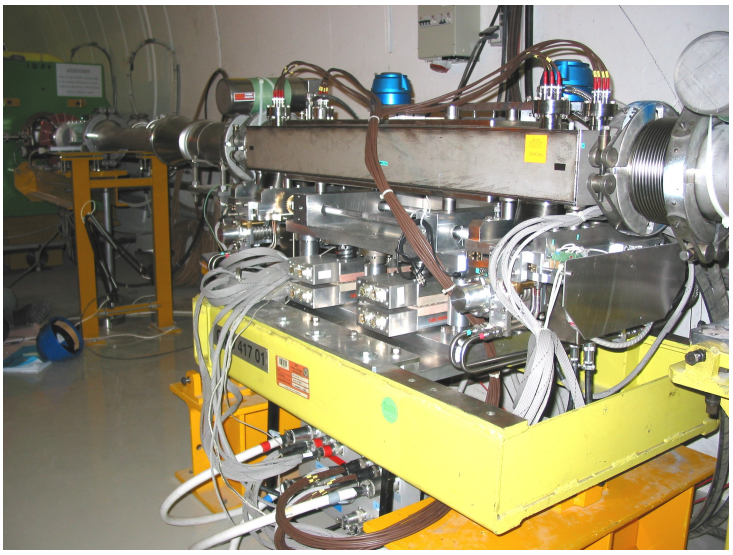

Figure 1.3: Installed collimator in the SPS accelerator

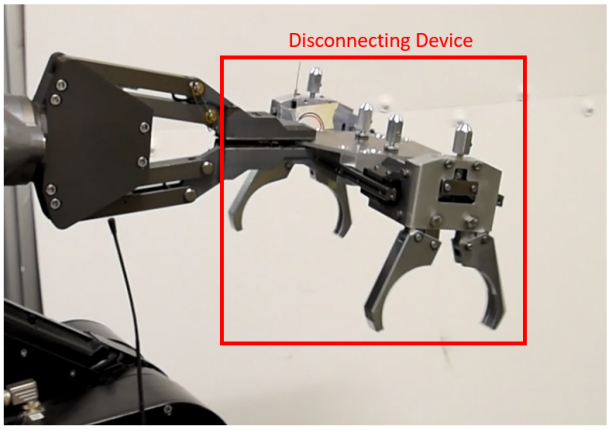

Figure 1.4: Disconnecting device

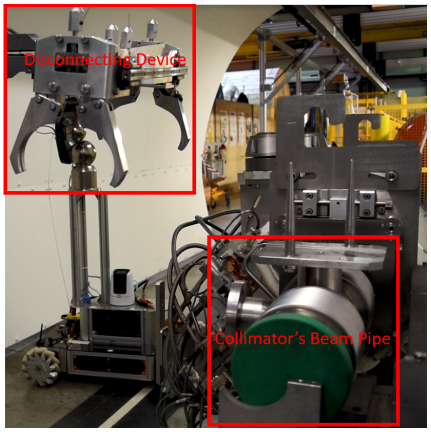

Figure 1.5: Beam pipe and disconnecting device

1.2.2 Collimator Disconnection Task

One specific activity for maintaining the collimators is the disconnection task of the beam pipe. As the name suggest its purpose is to disconnect part of the accelerators ring, also known as the beam pipe, in order to perform other works; whether on the collimator or on adjacent devices. This task will be a motivation in which this work is developed. The automation of this activity can reduce the stress and time consumption of the operators which in return will provide better efficiency in the long run due to its repetitive nature.

The following is a mock-up test done by the robotics' team in one of the laboratories to perform the before-mentioned task. A disconnection system device is shown in Figure 1.4. It is meant to be put on top of the beam pipe near the collimator and both pieces are shown in Figure 1.5. After following a procedure of screwing several parts of the disconnecting device, the beam pipe can be separated from the main beam line.

By screwing the locations shown in Figure 1.6, the disconnection device is properly set and the beam pipe can be removed. To do so, the Figure 1.7 shows how the robot should move for this task. Care to note that even though several parts just need to be screwed,

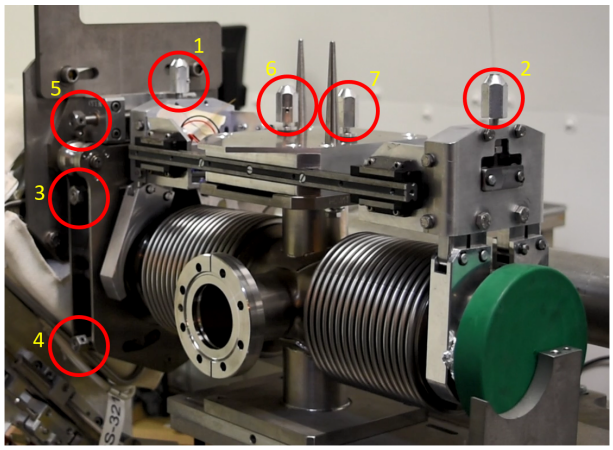

Figure 1.6: Location of each screw for disconnecting task and order of screwing

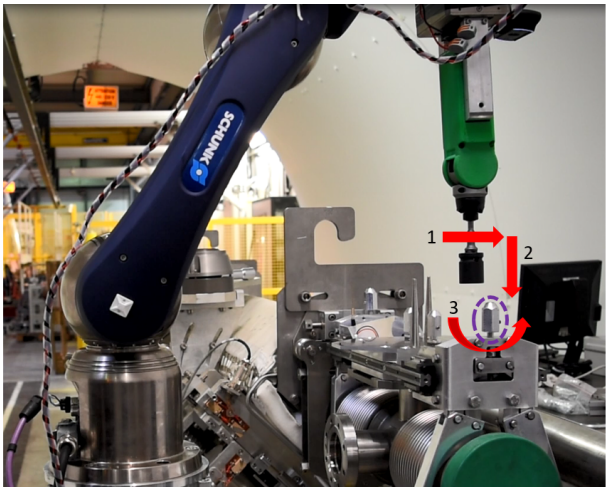

Figure 1.7: Robot screwing in the disconnection device

they must be done in a specific order.

Once the disconnected device has been set and the screws have been tighten, the beam pipe can be removed (Figure 1.7)and other maintenance tasks can be performed.

Nevertheless, the environment can not always be predicted precisely. Other operators do work around the area which might have left some undesirable obstacles, or a new equipment was installed and not taken into consideration during the mock-up simulations. Also, since the device are structurally highly complex it is possible that there might be some unforeseen parts in the vicinity. If this was the case, it is possible for the robotic system to cause damage in the workplace if any mistake were to be made. The proposed system is implemented to avoid this kind of outcome and to mitigate any damage that could happen.

1.2.3 Robots at CERN

CERN has a specialized unit for robotic interventions in the accelerator tunnels. As previously mentioned, the robots are used to substitute human actions near radioactive sources and allow a safer working environment.

The system framework for the robotic development is known as CERNTAURO, the

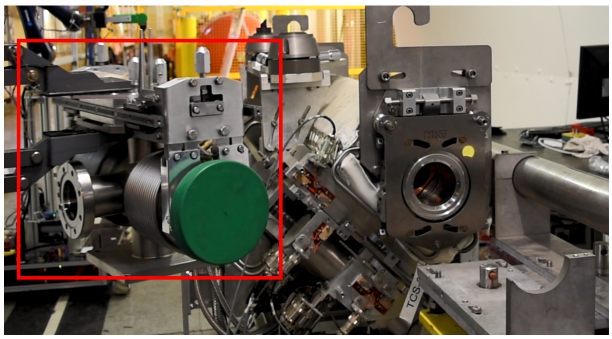

Figure 1.8: Disconnected beam pipe from the accelerator (mock-up)

CERN Telemanipulation semi-Autonomous Unit for Robot Operations. It is a software library for performing tele-operation tasks with semi-autonomous robots. It allows to work in hazardous environment by using different robot systems and devices, either developed at CERN or by other companies, e.g. KUKA or Schunk, and perform research in robotic solutions.

Figure 1.9: Kuka iiwa robot arm

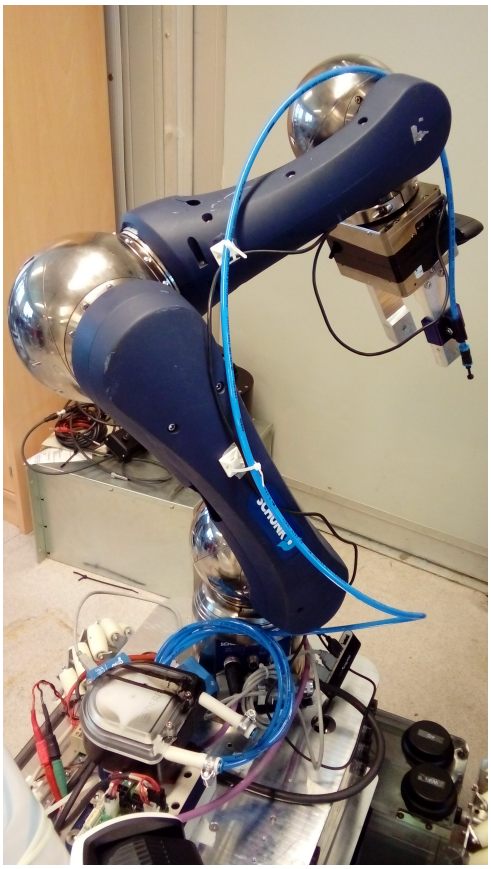

Figure 1.10: Schunk robot arm

In this framework, different modules are developed to perform a variety of functions. Wire-less communication between different robot arms, world mapping through sensor fusion, visual tracking and location identification with vision systems are just a few modules which comprise the framework as a whole. All of which are done to improve the performance of robotic interventions at CERN.

1.3 Master thesis organization

This thesis work is presented in the following structure. Chapter 2 is an overview of robot learning. Different algorithm and the challenges that arise from using this approach schemes.

Chapter 3 is the definition and explanation of dynamical movement primitives, a path planning framework for trajectory generation, and the locally weighted regression learning algorithm used for this work. The basic structure of the framework is presented and the proposed modifications are explained to understand the underlying work of the system.

Chapter 4 is the implementation structure of the system. A more detailed explanation of the logic in each part of the proposed solution. Here the learning and control phase are explained to understand the system's algorithms.

Chapter 5 are the results obtained through the proposed solution for this work. The explanation of different test are done, and the validation of this work's proposal.

Finally, Chapter 6 are the conclusions and future work of this thesis. A summary of the results are explained, future improvement and possible new implementations are pointed out.

Chapter 2

Robot Learning

In this chapter an introduction to robot learning is presented, and the challenges for this field when using this approach in modern robotic systems.

In recent years, developments have been on demand to learn from demonstration due to an easier way of programming robots. Algorithms to expand the capabilities of robot systems through interactive teaching has become an attractive method for future developers Chernova and Thomaz (2012). These methods are also known as imitation learning. The process for capturing skills to learn from, is done through different methods and various elements are used to learn a policy which is closely related to mimicking a behaviour.

2.1 Robot Learning Challenges

Robot learning has a wide range of disciplines which are involved. For this reason, many challenges appear when approaching this kind of solutions. In this section these are presented and briefly described.

Learner and Teacher Correspondence

A concerning issue when dealing in robot learning is the correspondence problem. Here it is important to know how can the several input of actions can be translated into the working space of the learning system. A difference between body structure of the teacher and robot can be problematic in the learning algorithms frameworks. The structures dissimilarity needs to be compensated during training. Many cases are that the learning robot acquires the shape of a desired trajectory and then through trial and error it achieves its goal by compensating the dissimilarities performed, similar to reinforcement learning algorithms.

When talking about correspondence, the user needs to account for the difference between kinematic structures in learner and teacher. That is one of the reasons why using kinesthetic learning method helps to bypass this problematic situation. The teacher can use the robots body to make it learn a skill and does not rely on other body structures or components for generating the demonstration for a specific behaviour.

Machine Learning and Scalability

To map the relationship between input and output is not an easy task. One needs to consider what is the behaviour that a system should imitate. What are the actions

needed to encapsulate the desired pattern to be imitated. Solving this problem, often it is used machine learning techniques. An indispensable element in pattern replication by demonstration.

Machine learning is a field that explores the link in patterns and its relationship between an input and output state. It is meant to construct models that can predict a specific behaviour from a set of information, an important element when trying to learn a specific behaviour.

Even though machine learning has been widely used in the current literature of robot learning, there is always two great disadvantages which are intertwined; scalability and computational load. When learning methods are used in a robotics application, there is always problems in the robots that contain high degrees of freedoms, because the solutions can be intractable in a finite time (Kober and Peters (2010)). In real-time applications, this is a very limiting condition since the algorithm is constrained by the system's hardware; as it is also the memory and processing power. However, in our system this restriction is not highly restrictive, because real-time learning processing is not needed and the hardware specifications of the robot allows heavy computational loads in the system. Furthermore, this load is highly dependent on the specific type of learning method that is used. In this work, the pattern learning is done off-line and uses a fast-learning approach which does not require very high resources.

2.2 Robotic trajectory-planning

Trajectory-planning is a problem that considers commands needed to perform motions through the passage of time. In the field of robotics, a trajectory-planning algorithm starts with a problem formulation and defines the path needed to arrive at the solution, considering any restrictions there might by on the system. It will then formulate a series of desired points in time for the system to accomplish.

In literature, there are many different kinds of trajectory planning methods to reach a solution to the problem at hand. Otte (2015) presents an overview of the different trajectory-planning algorithms that can be used in the robotics field. For example, there is the potential field methods, depth-first search, Dijkstra's algorithm, random trees and many more. Nevertheless, all these methods rely in an important element which is how the world is modelled. For very structured and repetitive situations in which the environment will not change, these algorithms are very efficient and different can reach to optimal solutions. In a situation when the environment is constantly changing and the circumstances are not always the same, a reliance in the model can become very time-consuming and even impractical for applications.

It is desirable to use a learning model that does not rely heavily on the definition of environmental constrains or on the world's representation to find a solution. The system that can use the knowledge of an expert, in this case an operator, which can decide what actions should be taken. A trajectory planning algorithm that generates a stream of desired data-points for a specific path, that can react to deviations and adapt instantly with the use of global parameters. All these elements are incorporated in the proposed framework based on dynamical movement primitives and various modifications which will be described more in depth later in this work.

2.3 Learning Motion's Patterns

Learning motions patterns is the equivalent to performing a specific action that the pattern can relate to. This process begins with an acquisition of a demonstration. To obtain data is the essence of capturing a desired skill to be learned, but depending on the system, it is prone to having noise and sensor error. Noisy signals can result in unexpected and undesirable behaviour even if the proposed model is properly trained (Argall et al. (2009)). Nevertheless, it is possible to obtain good data measurements through the use of good sensors, and the application of algorithms can further reduced error in the training data if required.

Learning from demonstration also brings to consider the needed method that is required to accomplish this action. The training information can come from different sources, which can be separated by into two branches; demonstrations or experience. The former gets data from other systems, yet the latter receives training from its own system. The most important difference between these two is that demonstration can provide, but not necessarily is, an optimal action for a given situation, while the experience comes from different trial and error scenarios in which the system finds a solution by its own. What this means is that using demonstrations is more closely related to a supervised learning method. On the other hand, information taken by experience is more suited for reinforcement learning methods where there is a reward function that the system would like to maximize, or minimize depending on the function's definition. In this work, using supervised learning methods is useful since demonstrations will come from an expert on the task at hand. The pattern can be extracted from this information, so the robot can then replicate the learned dynamics.

In literature, there are many kinds of learning algorithms to can be used and has been extensively researched. The most common methods to accomplish this, in supervised learning, are classification, regression and neural networks.

Classification methods are usually related to high level abstraction in decision making Chernova and Thomaz (2012). Even though they can be useful in this work, the purpose is not to use a classify data obtained from a robot but to generate a stream of predictions that can replicate a continuous signal from a dataset. The more suitable approach is to use a supervised learning algorithm such as a regression method.

Artificial neural networks are also considered candidates as a learning algorithm to imitate a signal, but they demand a high amount of training-sets and time Hussein et al. (2017). This makes them unsuitable in a framework where fast learning is considered an advantage and desired path are un-predictable and constantly changing.

With regression methods, the input and output relationship is established by a function which takes an input and links it directly to an output. This relationship is not necessarily linear and it is continuous space. Therefore, regression algorithms are ideal candidates to receive information from a learned behaviour and use it to feed the control system in this work.

2.3.1 Regression

In the literature, regression methods have been extensibility researched in many different fields over the years. Furthermore, learning actions in a continuous space are very suitable for low level robot control which makes it advantageous in replicating paths. This is because the input is linked directly to the output, and an action can be represented

as a path in continuous values, as opposed to classification methods which uses classes and labelling. Regression algorithms are suitable in path learning since it can directly relate inputs and outputs for controlling a robot system.

Works done by Mülling et al. (2013) use Linear Bayesian Regression to teach a robot play table tennis. This method uses a statistical analysis related to Bayesian inference. Gaussian processes make use of a likelihood function to capture the existing relationship between input and outputs in a Bayesian setting. Even though this method can be very precise and robust, it falls deeply in the curse of dimensionality. Thereof a higher degrees of freedom a robot will need a more computational load which might become infeasible (Meier et al. (2014)).

On the other hand, there is a common method known as the locally weighted regression (LWR). This algorithm captures a pattern suitable for trajectories where motions are continuous signal .Furthermore, LWR uses several local models to capture the essence of a pattern, resulting in a computational efficient algorithm. However, to obtain accurate and precise results, several tuning parameters must be set. Works like Schaal et al. (2003) and Ijspeert et al. (2013) uses locally weighted regression methods to encode various trajectories into dynamic movement primitives. Lauretti et al. (2017), uses this algorithm to learn rehabilitation motions for a robotic arm. LWR is widely used in many different applications and have proven good results in other works, therefore this method will be chosen for implementation.

2.4 Policy Representation

A policy is a mapping function that describes the state of a system. It is an element that the robot uses to execute different actions. These functions can be learned with various methods. It maps the continuous state of vector x in the control system and the environment, in a time dependent way, to a continuous output vector u (Equation 2.1).

$$u = \pi(x, \Theta, t) \tag{2.1}$$

A parameter Θ is a vector closely related to a specific situation. The model will be the source of behaviour that a system will take given the different inputs in a given situation.

Using non-linear dynamics as a policy is an idea coming from motor pattern generators (MPG) in neurobiology (Schaal (2003)). This idea got used as a base for the formulation known as Dynamic Movement Primitives which encodes a desired kinematic state of acceleration, velocities and position. For each movement there is a primitive that represents it, and therefore the motor primitives concept is introduced.

2.4.1 Motor Primitives in Robot systems

The concept of motor primitives was presented by Ijspeert et al. (2002) as a form of representing a control policy for basic movements in humanoid robots. Complex motions can be composed of sets of primitive actions and can be executed either in series or parallel. It provides several benefits of stability for singular, or rhythmical, movements and presents features of re-scalability. The representation for this policy is lineally dependent on its parameters which can make the learning very fast in robotic applications.

Other works involve task-parametrized movement learnings which uses automatic adaptations when new situations are encountered. Task parameters take landmark points

that the robot should pass and encodes a task-adaptive motion into the system Calinon (2016). Nevertheless, the use of this method is presented on pattern representation through Gaussian mixture models. Algorithms that rely on the prediction by Bayesian inference are very computationally expensive which makes them time-consuming to use.

The framework in which this work is going to be developed is using the dynamic movement primitives. A framework which relies on its own set of dynamics, and brings several benefits in its use.

An adaptability to bring in new situations and the linearity of the parameters Θ that can define a policy during specific situations are benefits for using this framework. Additionally, its functionality extension through coupling terms, which will be explained later in this work, make it a very powerful policy representation. Furthermore, fast learning algorithms are applicable to define specific patterns which is very useful during implementation.

The dynamic movement primitives framework allows coordination of many several degrees of freedom through the coupling of many non-linear equations. All these elements are taken into account for the further implementation of such framework.

Chapter 3

Dynamic Movement Primitives

It has become common practice in science to model natural phenomena with systems of coupled non-linear differential equations Ijspeert et al. (2013). This approach allows using systems that can provide rich and complex behaviours to obtain known results without the need to specify details of the generation of such behaviours. Dynamical systems offer an alternative to the standard generation of trajectories since motion is produced through differential equations.

The dynamic movement primitives (DMP) is a generic model system that uses multidimensional systems of weakly non-linear equations to generate patterns according to the parameters that are defined in its core equations. It's base functionality is based on the creation of a landscape using the commonly known point attractor spring-damper dynamic equations, which can be used for accomplishing a task in a more complex system. In other words, it produces a flow field from a dynamical equation which has an encoded desired trajectory which can represent kinematic motor behaviour which can also be further extended to multiple degrees of freedom. This by no means implies the needs to have an increasingly high complexity in order to create complex behaviour. On the contrary, due to its simple form, it is feasible to create very complex systems from a simple dynamic.

There are various benefits in using this generic system known as DMP and it can be summarized in the following points.

- Learn smooth and complex trajectories that can represent a task or at least the motion that involves performing it.
- Has no direct dependence of the time variable.
- Provides stability in the coordination of multidimensional systems. The differential equations converge to a point in space and automatically adapts to perturbations of the system's state.
- Learning open parameters can be done through non-complicated algorithms.
- Functionality can be extended with the use of new coupling terms.
- The trajectory's scale can be automatically adapted without the need to re-learn parameters.

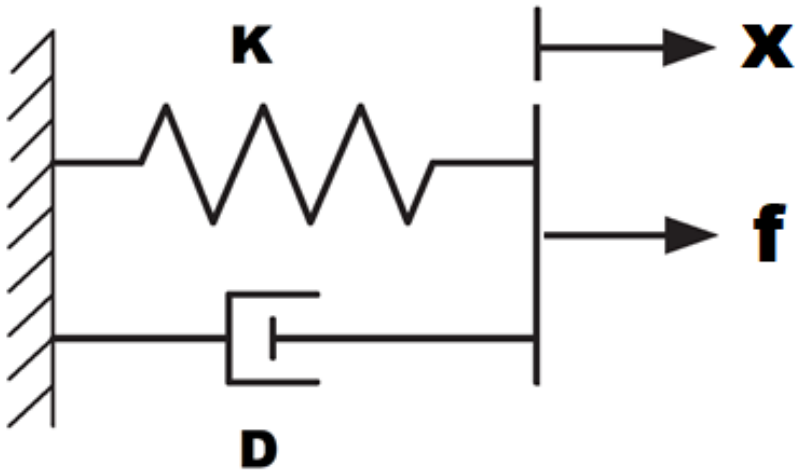

Figure 3.1: Spring-damper model representation

3.1 Transformation System

Dynamic movement primitive is originally based on differential equation which is very well known and has several properties that are advantageous. This is defined as the transformation system of the DMP, and it is a fundamental base where all motion will be generated from. The simplest attractor system that can be used is based on the dynamic formulation of the damped-spring model as shown in figure 3.1.

The differential equation that represents a one dimensional spring-damper system disturbed by an external force is the following equation as indicated by Glatz (2012):

$$\tau \frac{d^2x}{dt^2} = K(g - x) - D\frac{dx}{dt} + (g - x_0)f(t) \tag{3.1}$$

$$\tau \frac{dx}{dt} = v \tag{3.2}$$

The system's state variable is given by x, which is usually defined as a position and its respective derivatives. The parameters that can modify the behaviour are K the stiffness constant, D the damping parameter, g the destination (goal), x_0 the starting point and τ the temporal scaling factor. Finally, there is the $f(t)$ term, which is the non-linear forcing function that determines system's path to reach the goal position.

Notice there exists a factor term next to the non-linear forcing term defined as $(g - x_0)$. This value is a modulating parameter which eliminates the forcing signal once the goal has been reached by the system and it becomes an useful element to gain stability when reaching a goal position. It also brings scaling properties to the system which will be useful when adapting to new conditions.

The spring-damper dynamic's response is presented in Figure 3.2. One of the main issues of this dynamics is the high accelerations that are present in the motion's beginning. This will be addressed in the modified transformation system which will be explained later on.

In this work, the forcing term is defined in such a manner that it is active only during a finite time frame until the system reaches its goal. However, if it is needed, it can be chosen as a periodic function. The latter will work in a similar way as initially stated, but with repeating motions.

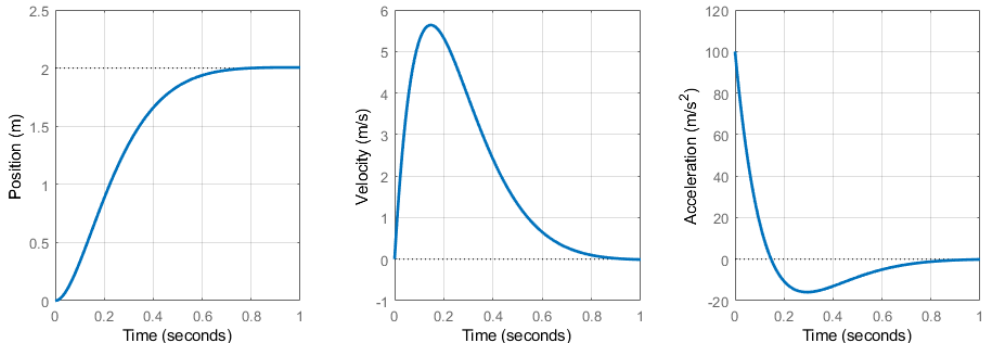

Figure 3.2: Spring-Damper graphical response in time without any external force. ($f(t) = 0$), $K = 50$, $D = 12.5$, $\tau = 1.0$, $x_0 = 0$, $g = 2.0$

Using the point attractor dynamics guarantees reaching the goal position due to the spring-damper behaviour. The parameters of K and D are chosen so the system is critically damped and converges monotonically towards the goal without oscillating Gams et al. (2014). Otherwise, the system will make a bouncing motion when reaching the goal position which it is not desirable.

To apply this kind of dynamical differential equations (3.1) in the definition of motor primitives] will require information about the position, velocity, and acceleration. These state variables will be provided by the demonstration done by an operator. And further integration methods will be used to obtain all the state variables, this will be explained further in the implementation chapter of this work. For now, it is assumed to get the signals as they are from a demonstration data.

3.2 Non-Linear Forcing Term and Canonical System

In the transformation system, it is of high importance to define the forcing term (f) found in the differential equation of the spring-damper model. This function will provide the property of changing the point attractor landscape, a desirable feature for performing complex paths. Therefore, it will define the specified pattern in the system.

A point attractor landscape is referred to the field flow that will outline how the system's state will operate throughout its duration. In this landscape, it is possible to encode complex movements that represent the desired movement behaviour.

The initial dynamics is to have a state variable that begins in a starting point that continuously move towards another final point in a linear path. When the state variable reaches the goal position, the system does not move from there. Adding the feature of "landscape" results in the state variable to move in a non-linear path. The changes of the landscape will change therefore the pathway and allow a more complex pattern to be reflected in the dynamical system.

Notice in Figure 3.3, the arrows represent a forcing value that change the shape of a non-linear function. The red line identifies the linear direction of the system if there were no forcing term involved. By specific moments the value will change and create variation in the differential equations which will in return change the function's shape, resulting in an different shape as shown.

In order to achieve this feature, the base of forcing term function will be chosen as a

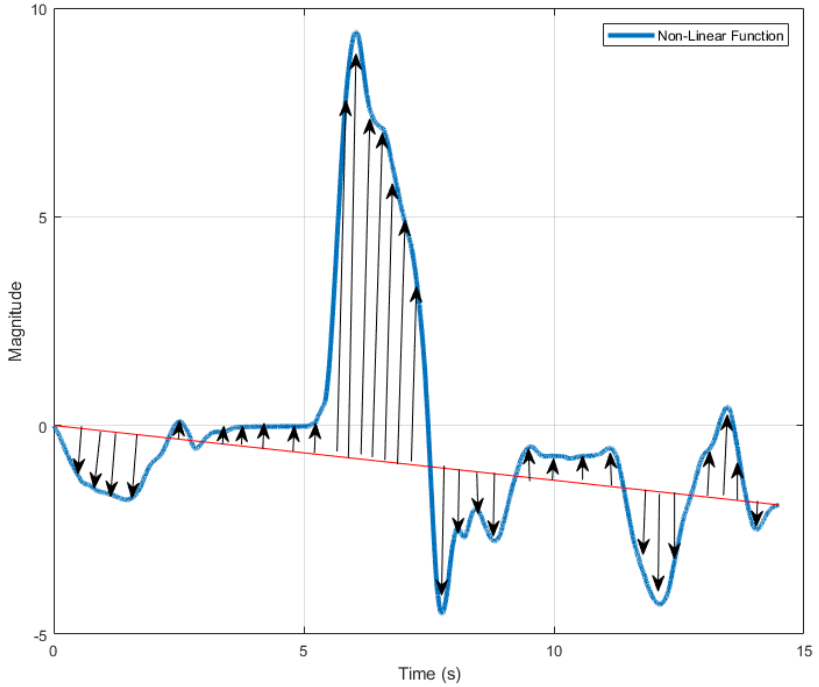

Figure 3.3: Landscape field flow representation for a non-linear trajectory

non-linear defined as the following equation Ijspeert et al. (2013).

$$f(t) = \frac{\sum_{i=1}^{N} \psi(t) w_i}{\sum_{i=1}^{N} \psi(t)} \tag{3.3}$$

The forcing function is based on the normalization of linear combinations of a fixed basis function ψ, also known as the kernel function. The w_i are adjustable weights which can be learned through different methods of machine learning and N is the number of ψ functions defined previously.

In this work, a simplified squared exponential function is used as a fixed basis function. Which has proven to give good results like in the works done by Gams et al. (2014), Samant et al. (2016) and Hoffmann et al. (2009).

$$\psi(t) = e^{-h_i(t-c_i)^2} \tag{3.4}$$

The kernel function parameters are t which is the current time of the system, h a parameter which defines the bandwidth and c the centre location. Its graphical representation is presented in Figure 3.4. Both the combination of Equations 3.4 and 3.3 create a non-linear forcing signal that is dependent on the activation kernel function. This means that there must be a distribution alongside the sampling signal in order to obtain the desired non-linearity, and depending on the number of kernel functions, the system will be able to represent more complex behaviours.

The forcing term, defined as Equation 3.4, is a good candidate to encode a pattern into the dynamical equations. Nevertheless, there exist a direct time dependency of the time variable which will not allow a straightforward coupling with other dynamical systems Ijspeert et al. (2013). This is because, by definition, every system is not coupled

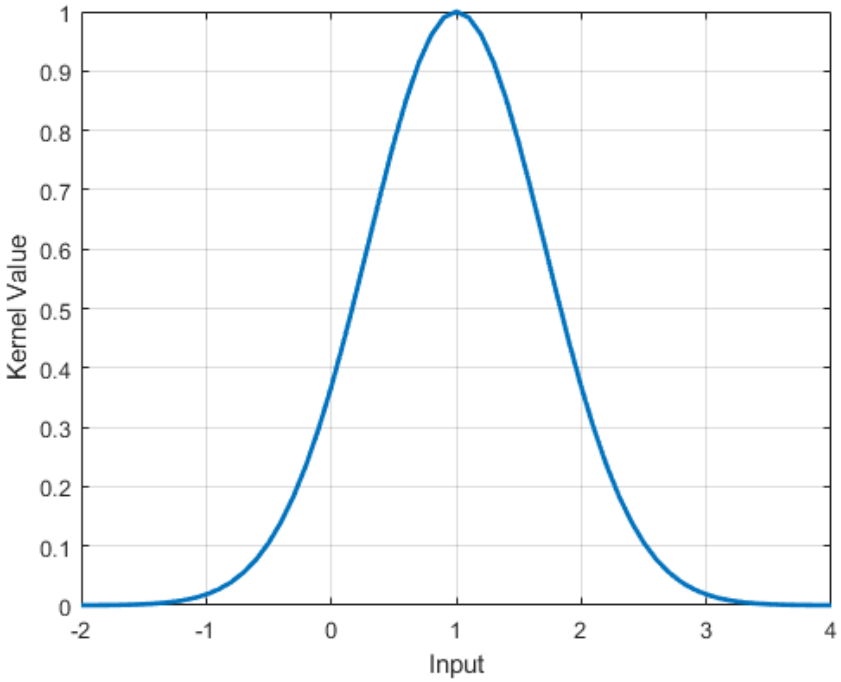

Figure 3.4: Squared exponential kernel function ($h = 1.0$, $c = 1.0$)

with each other and therefore its time duration might be different between each other. Furthermore, any variation in the system will be reflected in different timestamps which can create undesirable behaviours. To solve this problem, the DMP framework presents a new differential equation known as the canonical system.

$$\tau \frac{ds}{dt} = -\alpha_s s \tag{3.5}$$

In this new system, the state variable is defined as s and the parameters are α_s a decaying constant and τ a temporal scaling parameter. The purpose of using this formulation is to change the time dependency of the DMP, in particular, of the forcing term f. By replacing the time (t) with the new state of the canonical system (Equation 3.6).

$$f : t \rightarrow \ s \tag{3.6}$$

Now that the new state variable is implemented, it is important to take into consideration that now, the state of s converges monotonically to zero and the point where $s = 0$ is a stable fixed point.

As a result, we reformulate our forcing term so the new parameter s is taken into the equations and therefore will change accordingly and avoid the time dependency of the time parameter in our forcing equation term 3.7.

$$f(s) = \frac{\sum_{i=1}^{N} \psi(s) w_i s}{\sum_{i=1}^{N} \psi(s)} \tag{3.7}$$

Moreover, due to the time dependency change, the kernel function needs represent the same modification shown in Equation 3.8. Therefore, the kernel function is changed to adapt to the new changes.

$$\psi(s) = e^{-h_i(s-c_i)^2} \tag{3.8}$$

From the canonical dynamics point of view, the system can start at some arbitrary value (e.g. start at 1) and will always converge monotonically towards zero. Consequently, it will, like the forcing term, be active only during a finite time. By removing the time dependency through this formulation, the system will be capable of coupling with other dynamical systems while also featuring an online modulation which will be mentioned later in this chapter.

The canonical system brings the functionality of synchronization with other dynamical systems. Normally, the transformation system is used to represent a single degree of freedom and is limited by the state in which it is defined. Meaning that to depict a more complex system, several different transformation systems have to be used altogether.

Different systems running separately can cause unexpected behaviour since it is the group of every state that is desirable to learn. Therefore the system must be bounded by a singular entity that can control the flow of different states simultaneously. The canonical system's equation can work like an anchor that feeds each transformation system and keeps the different degrees of freedom in synchronization between each other (Figure 3.5).

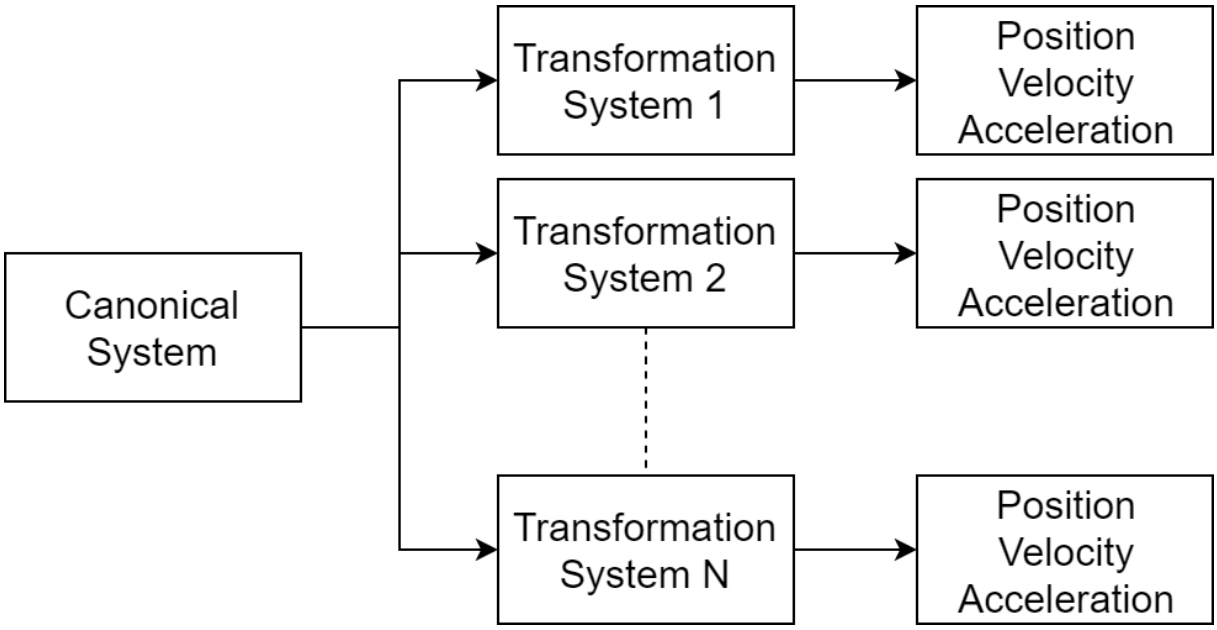

Figure 3.5: Diagram of canonical system use for synchronization in N multiple degrees of freedom

3.3 Modified Transformation System

One of the main important elements in Dynamic Movement Primitives is when defining the transformation system differential equations. Initially, it presents good behaviours but under certain conditions, the system can behave erratically which can be very undesirable. This is especially the case when handling high sensitive tasks and use high-cost types of equipment.

The transformation system expressed in Equation 3.1 has several drawbacks that have been pinpointed by Pastor et al. (2009).

1. When the starting point (x_0) is the same as the goal (g), it will generate no movement because the acceleration for this case will be zero.

2. If the goal is close to the starting point then a small change of the variable will create high accelerations that can go beyond the limits of the physical system.

3. There is a high dependence of the system's reference axis. If the goal position is changed across the zero point of the reference axis, then the whole movement can invert its shape.

These before-mentioned drawbacks can generate undesirable behaviours. Therefore, to create movements without singularities and high accelerations there is a need to choose a differential equation that will be invariant under invertible affine transformations. A new formulation, inspired by findings in neurophysiology, of the differential equations was presented by Hoffmann et al. (2009) with the following structure.

$$\tau \frac{d^2x}{dt^2} = K(g - x) - D\frac{dx}{dt} - K(g - x_0) + Kf \tag{3.9}$$

The parameters are the same as in Equation 3.1. Under this new formulation, the movement is improved and the system is not highly dependent on the coordinate system where it was initially defined in.

An independent equation of the coordinate system is very important since the robot's position can be changed depending on the task needs and space constrictions. If the learned movement changes orientation during the task, it might become a source of problems during operation.

3.4 Learning Dynamic Movement Primitives

Dynamic movement primitives rely on properly learning of parameters that define the desired trajectory. In order to learn a custom path, the algorithm can be divided in two parts. First, the global parameters must be stored and defined from the demonstration data, such as initial position (x_0), final position or goal (g), decay values (α_s), stiffness and damping coefficients (K, D), and learning hyper-parameters of kernel number and bandwidth (h). Second, to accomplish a mapping of the data, a supervised learning algorithm is used to compute the weight parameters (w) of the non-linear forcing term (f).

Initially, there is a reference trajectory that will represent the desired values of position, velocity and acceleration respectively ($[x, \dot{x}, \ddot{x}] \in$ Training Sample) given by any kind of demonstration to the system. From these signals, the data will be encoded by computing the non-linear forcing function target. The encoding Equation 3.10 is used which results of manipulating the Equation 3.9.

$$f_{target} = \frac{\tau}{K}\frac{d^2x}{dt^2} + \frac{D}{K}\frac{dx}{dt} - (g - x) + (g - x_0)s \tag{3.10}$$

Depending on the system's sensors, the values for velocity or acceleration may need to be computed if needed. Normally, it may appear troublesome to use numerical methods for the differentiation of the position signal since the calculation will create a lot of noise in the resulting signal. It could cause problems which can generate sudden unwanted high accelerations in the system, however, if a low-pass filter is used then the noise will

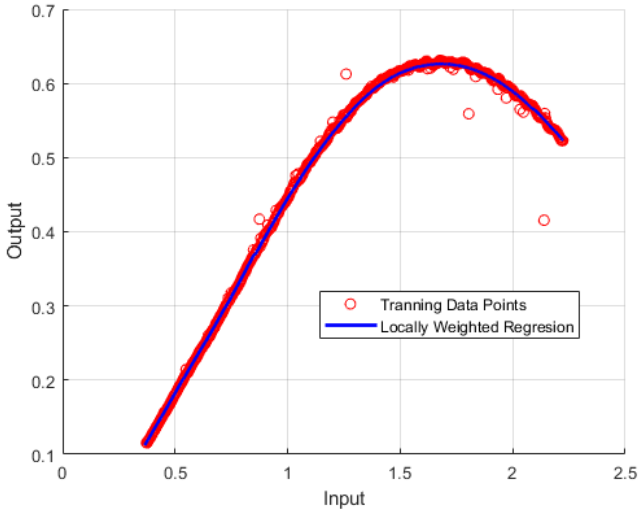

Figure 3.6: Locally Weighted Regression example

be mitigated to a certain degree and the resulting signal will be smoother. Furthermore, by using differential equations, the system will have a smooth response. In a sense, the transformation system will work as a filter as well.

3.4.1 Locally Weighted Regression

To map the reference of a provided dataset, a piece-wise linear model of the kernel function is going to be used. This will be known as the non-linear forcing term, and the different weights that define this function needs to be computed.

In this work, the Locally Weighted Regression will be used (Figure 3.6). Nevertheless, the learning algorithm can be changed according to the needs of the system and other supervised learning methods are also applicable (e.g. Gaussian Mixture Regression).

There are several benefits to using LWR in this work. The algorithm is not overly complicated, its implementation is very straightforward and the fitting of the curve has presented in other works good results. Moreover, its computational speed is faster in comparison with other algorithms like the Gaussian Mixture Model Regression which requires the dataset to be accessible at all times in order to generate predictions through the Bayesian formulation. This is a greatly desirable treat when looking to create various dynamical movement primitives as it reduces the computational load and time consumption as a one-shot motion learning algorithm.

Locally weighted regression is a non-parametric statistic algorithm useful for learning complex behaviours in autonomous robotics systems. It uses several local models and finds the line of best fit. It creates the best function that can represent a relationship between the input and output variables.

In this work, the non-linear function will be based on the equation 3.7, where there will be several kernel functions that will be activated only between a defined range like shown in figure 3.7.

The formulation will then be able to represent non-linearities through the activation of several kernel functions ψ. For each one of them, the value of w_i is calculated by minimizing the locally weighted quadratic error of Equation 3.11.

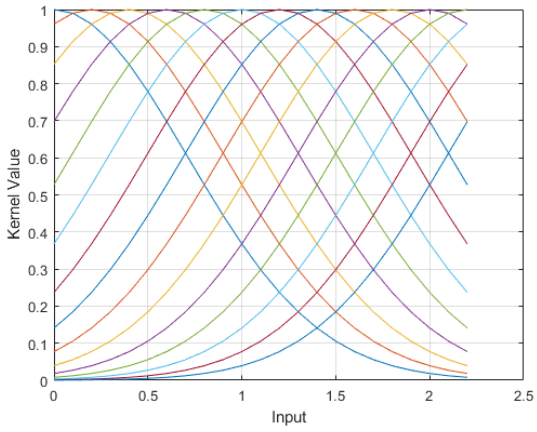

Figure 3.7: Kernel activation functions example

$$J_i = \sum_{t=1}^{P} \psi(t)(f(t) - w_i \xi(t))^2 \tag{3.11}$$

The parameter of $\xi(t)$ is defined as the equivalence of time into the canonical system's values. In other words, it is the result of fitting the trajectory's time variable into the canonical system equation's response. A visualization of this is shown in Figure 3.8.

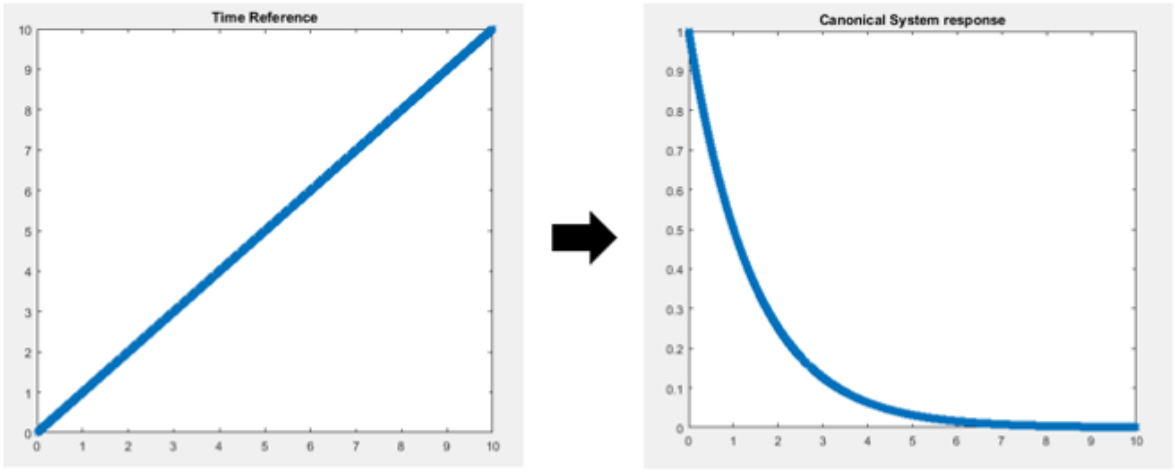

Figure 3.8: Time reference and canonical system response

The algorithm as in any other supervised learning method is essentially to minimize the error that exists between the original data and the predictor's output. To accomplish this, there are two main steps that are normally employed.

1. Choose a base value for prediction. This means to select the value of the weights (w_i).

2. Calculate cost (J_i) and update value to reduce its value to a minimum.

This working principle is used mostly in any kind of gradient descent method to find the best solution. Nevertheless, these methods need a selection of the hyper-parameters

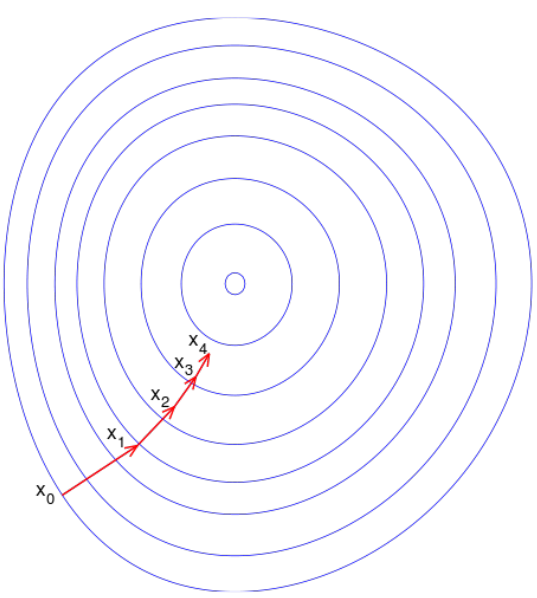

Figure 3.9: Gradient Descent graph solution representation Wikipedia contributors (2018)

such as the learning rate and iteration number. It is not an autonomous method unless some cross-validation is developed in order to select the best hyper-parameters and might become computationally expensive to perform; e.g. using a high number of iteration in order to get the best minimization of the cost function or changing the learning rate iteratively to optimize the weight parameters. With large datasets, this can become time-consuming as the algorithm uses many of the system's resources.

To solve the minimization of the locally weighted quadratic error as expressed by Equation 3.11, besides using a gradient descent method, it is possible to use other techniques. A fast algorithm that can obtain the same results is the normal method employed in the linear regression methods. In this case, there is no need to perform several iterations to obtain the most optimal solution.

A downside for using this method is that the number of data points is correlated to the solution's accuracy and the computational cost. Due to the nature of inverting matrices, the solution might not be found or depending on the computation it might become numerically unstable. Nevertheless, this can be mitigated when using the locally weighted regression method since the dataset is split into many different local models. The number of data is reduced by the selected kernel function's bandwidth (h_i) and therefore the normal method can result in an numerically accurate solution independently of having a dataset with a high number of data points.

The normal equation used in this work and is based on the paper presented by Ijspeert et al. (2013) shown in Equation 3.12.

$$w_i = \frac{S^T \Gamma_i f_{target}}{S^T \Gamma_i S} \tag{3.12}$$

The parameters of S, Γ_i, and f_{target} are all matrices that represent the function's input, kernel function values and force target signal respectively; i refers to the selected kernel function. They are defined in the following way, where P denotes the maximum number of data points in each local model.

$$S = \begin{bmatrix} \xi(1) \\ \xi(2) \\ \vdots \\ \xi(P) \end{bmatrix} \Gamma_i = \begin{bmatrix} \psi_i(1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \psi_i(P) \end{bmatrix} f_{target} = \begin{bmatrix} f_{target}(1) \\ f_{target}(2) \\ \vdots \\ f_{target}(P) \end{bmatrix} \tag{3.13}$$

This is known as batch regression and as its name suggest, the solution is calculated from each batch which will result in the corresponding weight value for the predictor function.

3.5 Canonical Coupling Term

One of the greatest benefits of using differential equations is the capability of functionality expansion through coupling terms. It directly relates to the system's maintainability as it is of ease to expand the basic performance of the defined primitive. This means it is possible to incorporate new features in the trajectory generation algorithm to improve the system's performance in a simple manner and achieve more complex behaviours.

The capacity of these coupling terms is not fully taken advantage of this work, yet is important to know that the system can be further improved in future works. An example of coupling terms use in other projects is the integration of a collision avoidance signal into the differential equations done by Ijspeert et al. (2013) and Stavridis et al. (2017). Also, synchronization between robot arms done by enhancing the second order equations as accomplished by Umlauft et al. (2014), or realization of bi-manual tasks through an existing dynamical movement primitive framework extension with the use of force feedback signals by Gams et al. (2014).

In this work, the coupling term is used to accomplish the task of stopping the system trajectory generation if there is a high deviation between the generated movement and the actual robot's position. The resulting behaviour will allow for the system to halt safely when there might be some unexpected obstacle that was not taken into account when training the system, minimizing any damage to the robot and environment. This method allows for the system to monitor every degree of freedom with little to no effort, and without the need of using other external sensors to detect the collision. This method is based on the work done by Umlauft et al. (2014), where the canonical system's velocity is enhanced by the inclusion of an error as it is shown in Equation 3.14.

$$\tau \frac{ds}{dt} = \frac{\alpha_s s}{1 + \gamma_{error}(\tilde{x} - x)^2} \tag{3.14}$$

The parameters of a new enhanced canonical system are γ_{error} the error constant, $\tilde{x}$ actual position of the manipulator, the x value of the transformation system and τ time constant. Care to note, that the manipulator's position and the transformation system will be points in a task space denoted by the end-effector's coordinates in the world. Hence an error value obtained with this formulation will be the quadratic euclidean distance from one point to another.

The modification done in the canonical system will generate a decrease of the decaying rate which will depend on the parameter γ_{error}. In consequence, a low decay rate will affect all the transformation systems that are linked and generate points at a slower rate, creating a small transition to stop the robot. When an error increases towards infinity,

the system will come to a halt since the system's decay rate will tend to 0 as shown in Equation 3.15. When halting the system, it automatically saves the last position which can be used then as a reference to restart the system.

$$\lim_{\tilde{x}-x\to\infty} \frac{\alpha_s s}{1+\gamma_{error}(\tilde{x}-x)^2} = 0 \tag{3.15}$$

On the other hand, if the system has no errors (Equation 3.16) then there is no change in the canonical system and all the transformation systems will work normally.

$$\lim_{\tilde{x}-x\to 0} \frac{\alpha_s s}{1+\gamma_{error}(\tilde{x}-x)^2} = \alpha_s s \tag{3.16}$$

Nevertheless, these equations cause the system to slow down but never to stop. As the value only reaches 0 by having an infinity error. Therefore, a new equation is proposed based on the logarithmic function and is presented in Equation 3.17.

$$\alpha_s = -Alog(B(\tilde{x}-x)^2 + C) \tag{3.17}$$

By modifying values of the decay rate directly, we can change the speed generation of data. It is even possible to have a negative speed and, instead of computing forward in time, it can compute backwards. This is especially useful in the detection of an obstacle in the system's path. Because when the error is detected, the system can retake the already performed path.

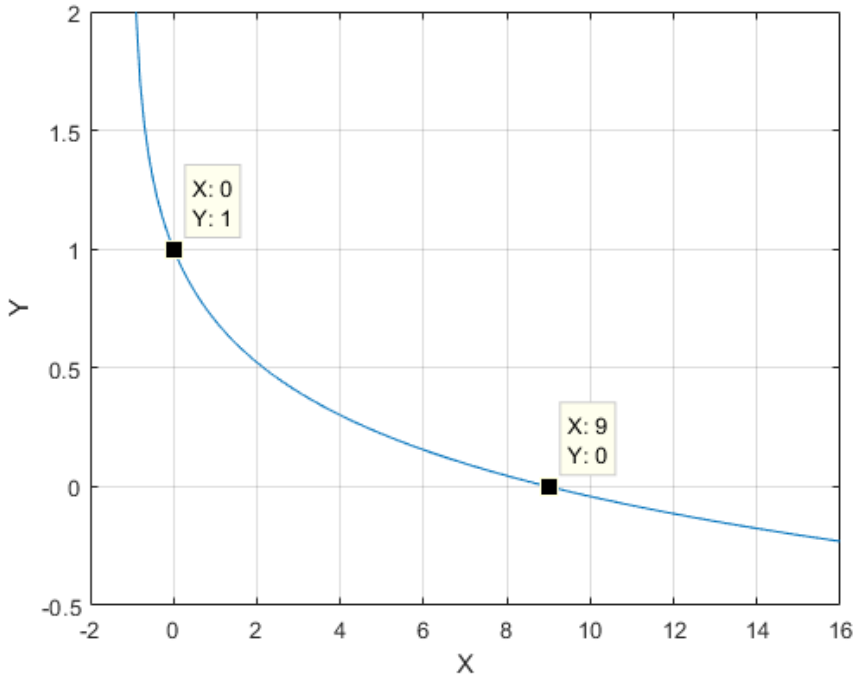

Figure 3.10: Logarithmic decay of function 3.17

The values of A, B and C are chosen following the rules presented by Equation 3.18

$$A <= 1.0; B > 0; C \approx 0.1 \tag{3.18}$$

Using these values presents the behaviour demonstrated by Figure 3.10. The maximum decay rate that the system can have is 1.0. This also means, that the error will decrease

the value at a maximum of -1.0, with certain point where the value become negative. The decreasing speed is linked to the value of B and it can be chosen depending on the needs, and it will define the location for the sign change of the logarithmic function. In Figure 3.10, B was chosen 0.1 and the value for reaching 0 is 9, which makes it a good candidate for the tests further in this work. Nevertheless, these values were chosen empirically and they can be tuned as needed.

3.6 Point to Point Modulation

Another feature that lies in using dynamic systems in motion planning is the possibility to modify the learned trajectory without re-learning the path. This can be done either during the system's initialization by just changing some of the parameters or done during the trajectory's execution (online).

Changing the parameters will result in an automatic adaptation to the new conditions and the transformation system will generate a new path without the need to deeply modify the system's structure. What's more, the computational load for modifying to the new adaptation will be low and therefore is acceptable an online trajectory change that does not disrupt the initial executed motion.

An example of such modulation is to change the goal position, either at the trajectory's starting point or during execution of the motion. Online trajectory changes can result in a sudden differential of positions which is an undesirable element in the current dynamical system since it can generate very high velocities and accelerations. Therefore, to create a smooth transition between goal positions a dynamic change based on another differential equation is introduced (Equation 3.19). This will result in a smooth change from one value to another and it will avoid having high accelerations due to the sudden change of the goal position.

$$\tau \frac{dg}{dt} = \alpha_g g(g_0 - g) \tag{3.19}$$

The state of the system will be the goal values g. The parameters represent the time scale τ, the constant rate of change α_g and the initial goal position g_0.

3.7 Primitive Motions Library

Performing repetitive tasks is of great value in any robotics application and its organization is an important aspect when dealing with several movement primitives. The concept of using motion primitives library is very simple and has been presented by other researcher such as Hoffmann et al. (2009) and Pastor et al. (2009). Every DMP can represent a basic movement, from these it is possible to build more complex motions to accomplish new and different tasks. Like creating building blocks and then used together in different combinations to achieve a new goal.

The library construction would begin with the teacher demonstrating a set of motions. These are then labelled and stored in the library. Each movement can be named depending on the task or motion, e.g. reach-object can move the robot arm from the start to goal position in an ellipsoidal trajectory.

From the labelling, it is now possible to use the library to perform a task that was not initially defined with each independent DMP. Yet to know which motion block to use,

it is needed a high-level interpretation of what they do. The use of this library is still dependent on the task which needs to be done, and the block usage will depend on every different task that needs to be performed.

Every motion block represents a basic movement, therefore primitive, and will require from the teacher a high consumption of time in order to label each block appropriately. Nevertheless, with high-level abstraction, it is possible to use the blocks to perform new and complex movements without the need to re-learn new motions.

Chapter 4

Implementation of Learning Framework

To make use of the Dynamic Movement Primitive framework in a robotic system, it is important to define a structure on which the system is built upon. It must be able to learn demonstrated trajectories and make use of them. The system will make use of an impedance controller, a dynamic movement primitive library, a locally weighted regression algorithm and an enhanced canonical system which all of them will be explained further in this chapter. To start, the framework will be implemented in two phases.

The work-flow overview shown in Figure 4.1 presents the before-mentioned phases named as learning and control respectively. In the learning phase, the system will learn the DMP parameters through a demonstration done by a teacher in Hand-Guidance Mode; the latter will be explained later in this chapter. These learnt DMPs will be stored in a primitive motion library. Afterwards, the control phase comes into play. The system, supervised by the user, will select the DMP from the library according to the desired task. Thereupon, an autonomous mode will use the chosen primitives and the robot control system altogether with the path generator is executed. On the side, there will also be presented a robot-guidance mode which as the name suggest it guides the user through primitive motions.

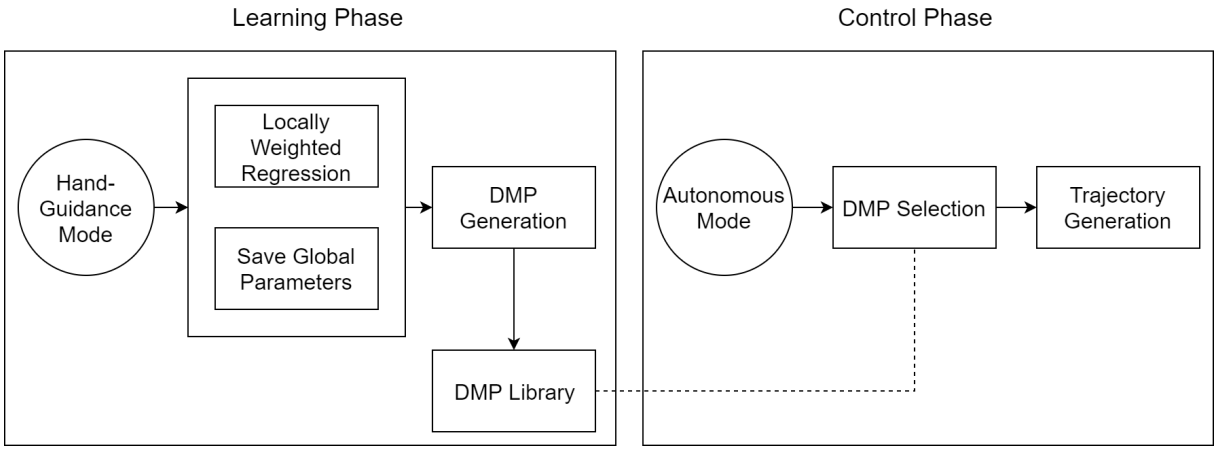

Figure 4.1: System schematic work-flow

4.1 Learning Phase

To make use of the operators working knowledge in a task, trajectories needs to be taught to the robot system. To do so, the learning phase is presented. In this phase, the system needs to be given a demonstration done by a teacher. This information will be stored and properly processed so the new primitive can be generated, after which it will be stored in a library for future use.

The chosen method to perform the demonstration will be through kinematic guidance, known as Hand-Guidance Mode. A user friendly mode where the robot can be guided physically through any trajectory, and with the use of its sensors to record relevant data.

4.1.1 Hand-Guidance Mode

For the current learning system, it is important to receive the teacher's input in order to learn the trajectory to accomplish specific tasks. Therefore, is important to have a method in which it is possible to obtain the robots sensors data that properly represents the teacher's intention; since it is directly related to the good performance of the system. For that reason, in this work, the Hand-Guidance Mode is presented. A human-robot interaction mode which will be a starting point in the learning system. Resulting in a capacity to receive specific data from the teacher and store it for further processing in the learning algorithm.

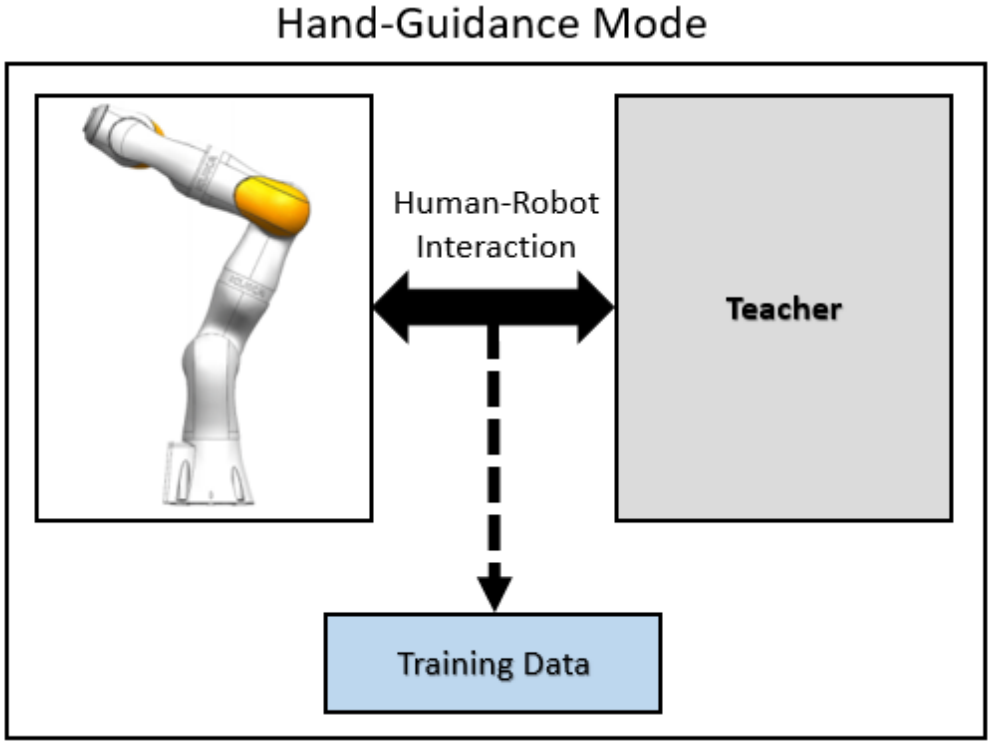

Figure 4.2: Hand-Guidance Mode representation

The hand guidance mode's basic principle is to use an impedance control to move the robot through different motions using the teachers guidance. Normally, but not

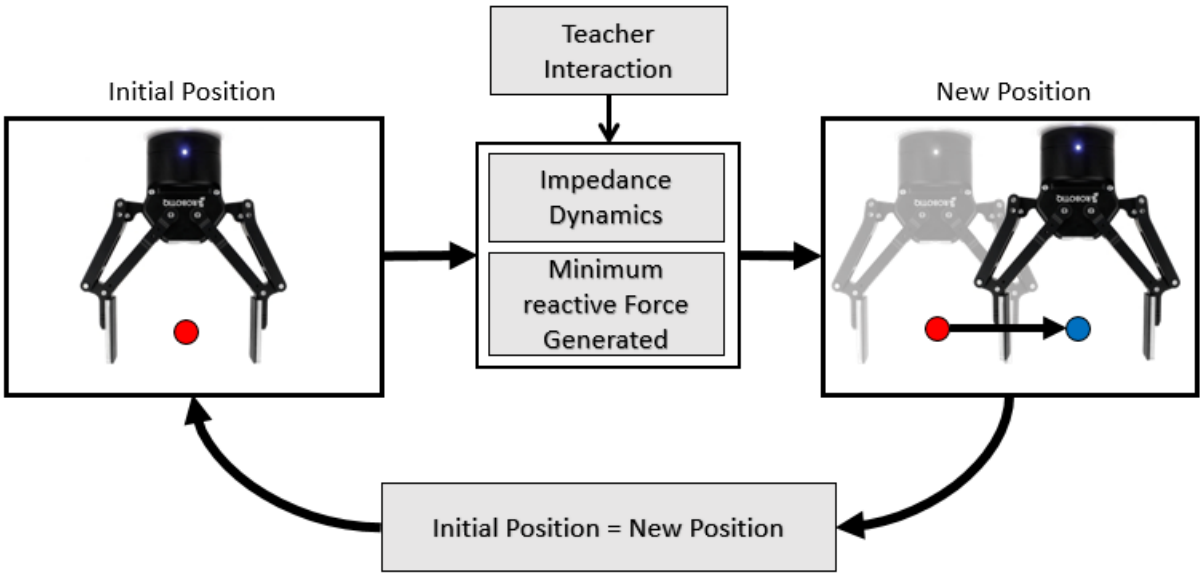

Figure 4.3: Robot end-effector position change with impedance control loop and human interaction

exclusively, the teacher's hand will be used to generate robot motion into its body and therefore the name of hand-guidance mode.

This approach allows a dynamic interaction between the environment and robot structure. The controller will generate a dynamic relationship between force and position through a spring-damper behaviour of the teacher's input force and the robot structure. These will be regulated through the parameters of stiffness and damping which play an important role in the hand-guidance mode.

The teacher will provide the motion generation input for the robotic system which will create a difference between the robots current and a desired position. The purpose of using the impedance control in the hand-guidance mode is to have a low resistance for the movements applied, so the teacher does not feel a strong repelling force coming from the robots mass and inertia while teaching a specific movement. Thus creating a minimum force interaction between the teacher and robot (Figure 4.3).

The impedance control scheme is used in joint space and it will allow to use the robot in any position configuration that is possible. First, the system is initialized and the impedance dynamics are applied to every joint of the robot arm in the initial pose. Stiffness and damping parameters are set at a minimum possible which will result in a very compliant robot arm. The teacher will move the arm as desired with very little resistance and at every instance of the control loop, the desired position is constantly updated according to the current position of the arm. This way, the robot pose will be changed according to the teachers desired motion.

For the whole process, the system will save all the data necessary to use in the learning algorithm (Figure 4.4). It is of interest to use the Cartesians position of the tool center point (TCP) with its orientation in Euler form $(x, y, z, \alpha, \beta, \gamma)$, and the joint configuration of the robot arm $(q_1, \ldots, q_n)$. All the data will be processed later on in order to learn the systems position in joint and task (Cartesian) space.

The impedance control is a pathway to new positions that are provided by the teacher. Nonetheless, due to the nature of a very compliant robot arm, it is of ease to reach the robot arm joints limits. This becomes a problem that can damage the robot arm and

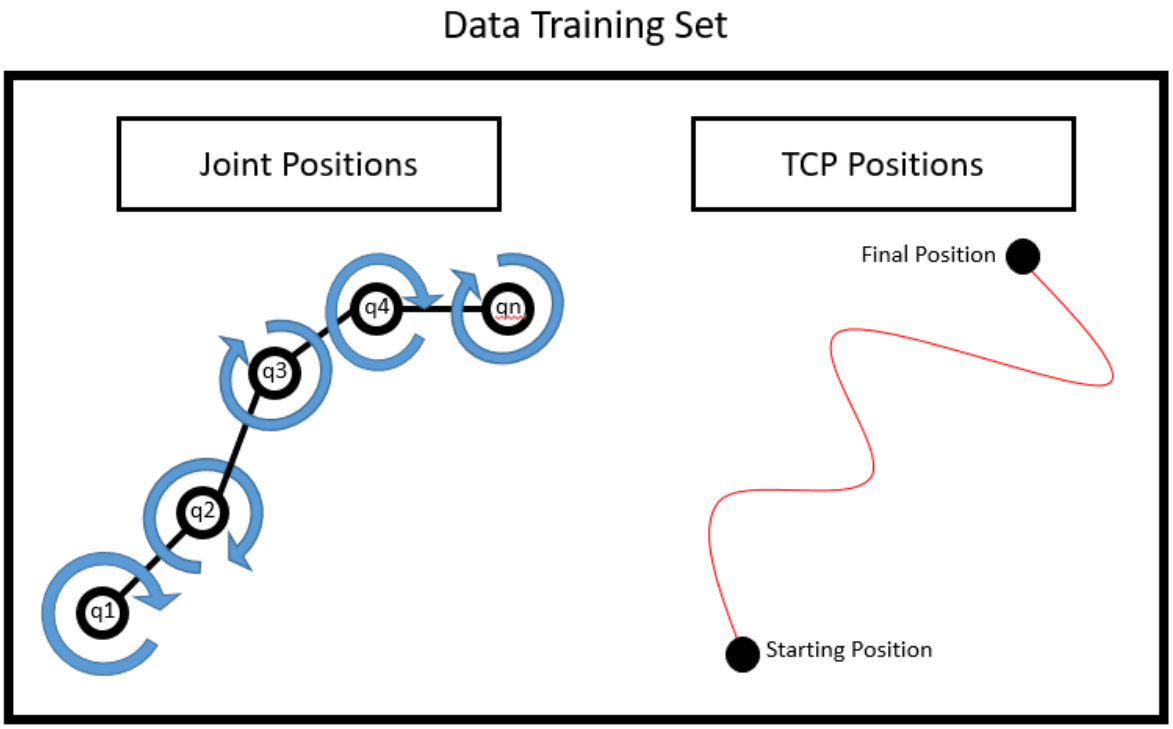

Figure 4.4: Data training set taken from robot motion

block its movement completely. Thus, this will be addressed by manipulating the teachers motion with the impedance control during the learning phase.

Through a slight modification of the impedance control's parameters it is possible to obtain different levels of compliant systems. In the specific range of values that are close to the limits of each joint, there will be an increase of the stiffness parameter for the impedance dynamics. The robot will no longer update the values of the desired position and, in return, it will generate a less compliant behaviour that creates a repellent force near the robot's. When the teacher puts the robot in an unfavourable pose for one of the joints, it is possible to feel a force which will not allow the system to go further and therefore avoid reaching the system's.

There are 3 different ranges that are defined to avoid going over the limits of the joints in the robot arm as shown in Figure 4.5.

- The free movement zone: Is where the teacher has complete freedom in moving the robot arm. Here the impedance parameters are set at a possible minimum so the teacher can move the robot in any way possible with little force feedback.

- The restricted movement zone: It is 2%-10% the robot limit's value which depends on the axis. In this mode, the stiffness parameter is increased according to how far the joint is with respect of the end-effector. This is done so the joints which are closer to the base have a higher impedance compared to ones near the end effector. Most of the times, a force is applied at the end effector for guidance, therefore, the robot joints will receive a higher torque when they are closer to the base.

- Blocking zone: Is the values less than 2% the limit's value. Here, the hand-guidance mode will stop all movement of the robot arm. This last feature is included as

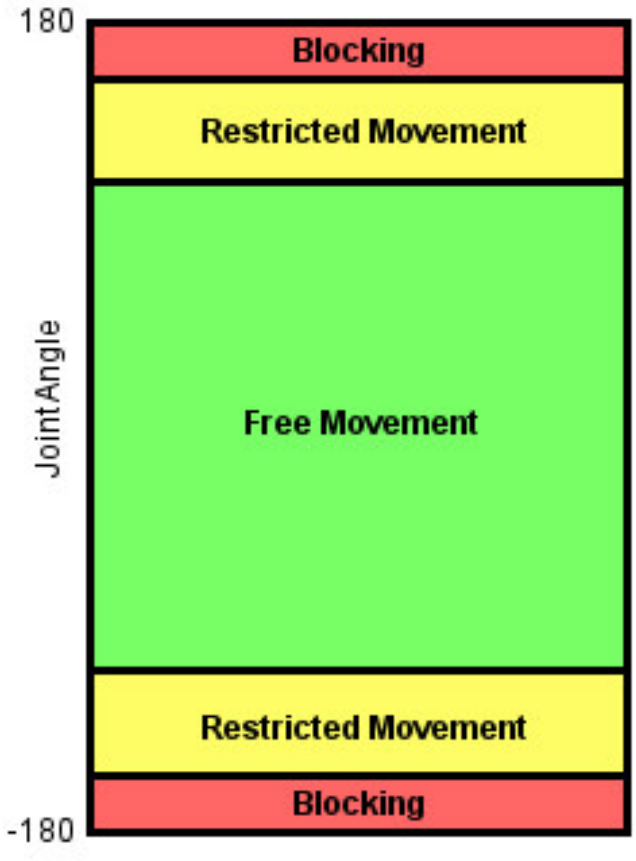

Figure 4.5: Joint's zones in hand-guidance mode

a safety measure of the robot so it doesn't reach under any circumstance the set limits.

As explained before, most of the workspace for use to the teacher is in the free movement zone as it has a minimum force interaction. On the other hand, inside the restricted movement zone, the teacher will feel a repelling force in the joint whenever it is getting close to its limits and can act accordingly to avoid blocking the robot's movement completely.

Using this mode can bring comfort to the user when trying to teach a specific trajectory and it has a safety measure to detect a bad configuration of the robot. It is a method for accomplishing kinesthetic learning and mitigates any discomfort that the user may have when trying to move a robot which can be heavy.

4.1.2 Data Processing

When working with learning algorithms, initially the signals need to be manipulated in different ways to obtain good representation of the patterns that are embedded in the data samples.

This section will be focused on explaining the steps taken to process the hand-guidance training data in a structure to learn trajectories through dynamic movement primitives.

In this sub-section the Figure 4.6 gives an overview of the flow-chart to do all the computations and achieve a learning system of dynamic movement primitives.

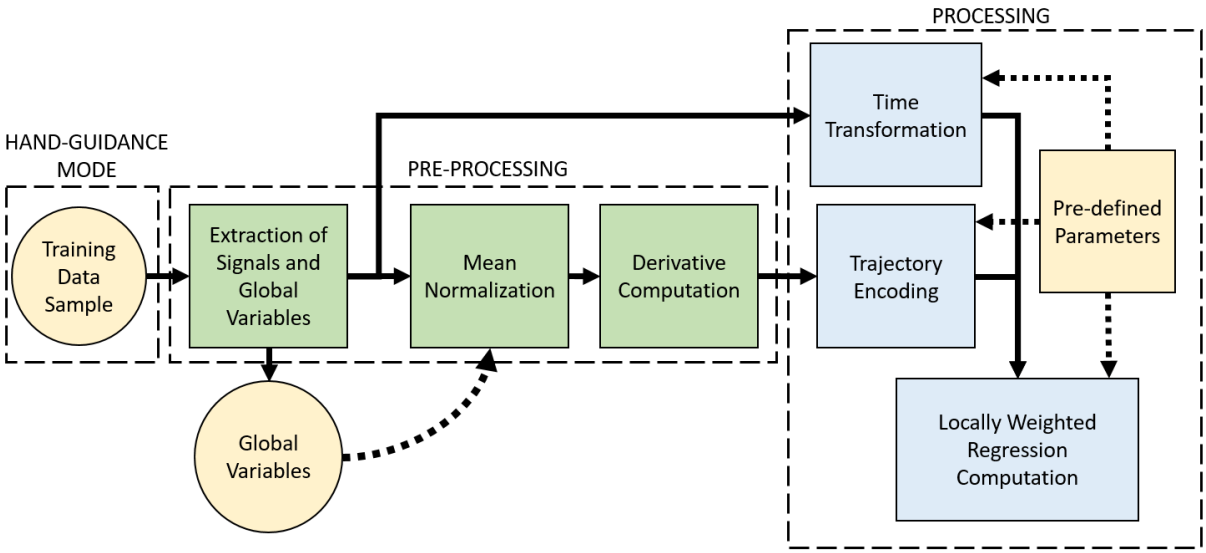

Figure 4.6: Data Processing Flow-Chart

4.1.2.1 Pre-Processing

Initially an interface is developed to create a preprocessing module. The raw data is initially saved with a structure of a matrix, where every column represents one degree of freedom and every row is a data value at each step. This does not mean it needs to be such but normally its interpretation is done like this throughout the work for simplicity. Once every column is identified as a degree of freedom, the global parameters are extracted for each one of them. These include, final and initial position, time duration, step size in between data-points and maximum-minimum values.

Once the global variables are extracted, the data needs to be adapted for the supervised learning algorithm. In pre-processing, a mean normalization is applied by using the Equation 4.1. This, for the most part, is a precondition to decrease the condition number of the matrix.

In numerical analysis, the condition number represents the output's sensitivity to changes in the input. It is of interest the computation of a normal equation (3.12) with a high number of data-points to obtain the weights that can best represent the trajectory. Nevertheless, due to the non-triviality of the computational process for solving the normal equation, the numerical stability needs to be taken into consideration in order to have a good numerical precision.

Computing big matrix solutions inside a real system can have some unwanted results if not treated correctly. Numbers are represented by the size of its memory allocation in the system and therefore errors are introduced. This is known as the foaling point numerical error. If a matrix is well conditioned then the precision will be good due to the condition number being low. A good precondition to get a well conditioned matrix is by using a normalizing factor into the values of the matrix (Hartley (1997)). This is because the numbers' representation can be more precise when removing pre-existing conditions that are already known. Take for example the offset in a dataset, this value will be constant throughout the signal which normally will occupy memory in its representation. If the

value can be removed then the computation of matrices will increase in its numerical precision.

$$\hat{X} = \frac{X - \bar{X}}{x_{max} - x_{min}} \tag{4.1}$$

Here the value of $\hat{X}$ represents the normalized signal, X the raw values, $\bar{X}$ the mean, X_{max} maximum and X_{min} minimum values. The data signals are then normalized to a zero mean and scaled to the unit value. Afterwards, the actual derivatives need to be obtained through a finite-difference approximation so it is possible to encode the trajectory into a primitive.

The forward and backwards formulations are common ways of computing derivatives in discrete signals. These are known to be a first order accurate approximation which means that the error's magnitude is roughly proportional to the step size. On the other hand, there is a centred approximation formula which is known to be a second order accurate. This states that the error is proportional to the step sizes' square value(LeVeque (2007)). Therefore, using a centred formulation will reduce the error from a computational approximation to a less degree in comparison with a first order accuracy equations.

The first derivative is calculated with the known symmetric difference quotient which is shown in Equation 4.2.

$$X' = \frac{X(t+1) - X(t-1)}{2h} \tag{4.2}$$

Where X' is the derivative, t is the time input and h is the step size. Similarly to compute a second derivative, the Equation 4.3 is used.

$$X'' = \frac{X(t+1) - 2X(t) + X(t-1)}{h^2} \tag{4.3}$$

4.1.2.2 Processing

Once all the signals' derivatives are computed, the implemented supervised learning algorithm can be used. First, each the degrees of freedoms are separated and the equation 3.10 is used in each one of them to encode the trajectory into the dynamic primitive through equations 3.12 and 3.13. In order to obtain the encoded trajectory some parameters must be pre-defined initially such as the stiffness (K) and damping (D) values. These can be chosen so the system can be critically damped, as done in many different works such as Samant et al. (2016), Schaal et al. (2003) and Karlsson (2017). In this work, due to the use of a modified transformation system, the stiffness and damping values were selected through empirical observations of the resulting learning algorithm.

Furthermore, the time signal needs to be transformed into the exponential canonical system's state variable. This is done through mapping of the linear time signal into a response of an exponential decaying function as shown at Figure 3.8. In this step, the parameters such as starting point (normally chosen as 1) and decaying parameter needs to be defined so the time signal is properly adapted into the canonical system.

Once the signal is encoded and the canonical system's input state variable is mapped, the locally weighted regression can be employed to compute the weights through the normal equation. Depending on the chosen amount of desired kernel functions, the weights are computed iteratively throughout of the encoded trajectory for the non-linear forcing function.

As a result, the system at this point is capable of generating the weights which will define the trajectory to be performed and the non-linear forcing function will be able to represent a specific movement.

4.1.2.3 TCP's Orientation Signal Continuity Algorithm

Initially, the values in Euler form will be taken from the robot system. This obtained by the forward kinematics from the robot structure. As a consequence there TCP's orientation values create a problem due to the representation needed for Cartesian space. The data values are taken initially in Euler form and they are constrained in the range of -180 to 180 degrees. Resulting in an undesired singularity near the plus/minus 180 degree position.

The value will abruptly change near the singularity, creating instantaneous jumps in the signal. As a consequence, the learning algorithm will fail to represent the data since it is imperative that the signal be continuous in time. To avoid this complication, the orientation must be modified to have a uninterrupted waveform.

The transformation will be done through the quaternion representation, the reason lies in its capacity of representing a continuous orientation signal through simple mathematical manipulation. The Equation 4.4 defines the transformation between Euler angles to quaternions.

$$\begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} cos(\frac{\alpha}{2})cos(\frac{\beta}{2})cos(\frac{\gamma}{2}) + sin(\frac{\alpha}{2})sin(\frac{\beta}{2})sin(\frac{\gamma}{2}) \\ cos(\frac{\alpha}{2})sin(\frac{\beta}{2})cos(\frac{\gamma}{2}) - sin(\frac{\alpha}{2})cos(\frac{\beta}{2})sin(\frac{\gamma}{2}) \\ cos(\frac{\alpha}{2})cos(\frac{\beta}{2})sin(\frac{\gamma}{2}) + sin(\frac{\alpha}{2})sin(\frac{\beta}{2})cos(\frac{\gamma}{2}) \\ sin(\frac{\alpha}{2})cos(\frac{\beta}{2})cos(\frac{\gamma}{2}) - cos(\frac{\alpha}{2})sin(\frac{\beta}{2})sin(\frac{\gamma}{2}) \end{bmatrix} \tag{4.4}$$

The vector $q = [q_w, q_x, q_y, q_z]^T$ represents the quaternion and parameters of α, β, γ define the robot's TCP orientation. Furthermore, in order to obtain once again Euler angles from quaternion the Equation 4.5 needs to be used.

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} atan2(2(q_w q_z + q_x q_y), 1 - 2(q_y^2 + q_z^2)) \\ asin(2(q_w q_y - q_z q_x)) \\ atan2(2(q_w q_x + q_y q_z), 1 - 2(q_x^2 + q_y^2)) \end{bmatrix} \tag{4.5}$$

One advantage taken from the quaternion definition, which this work makes use of, is that the orientation of a quaternion is the same if multiplied by a scalar as long as the unit length constrain is maintained. Thus, if the quaternion is multiplied by -1 the orientation does not change. This becomes an important element that will be used to create a continuous signal that can represent the TCP's direction.

When all the Euler orientation signal is transformed by the quaternion expressions shown in Equation 4.4, the resulting signal will also have unwanted jumps. Therefore, it is important to find when does the jump in signal occurs. This is done by observing the time location when the sign changes.

When the signal changes sign in Euler representation, the values in quaternion will change sign as well and it will be shown as well when transforming the orientation space. Nevertheless, in the quaternion space, all its parameters (q_w, q_x, q_y, q_z) change by its negative counterpart. Knowing this, the quaternion can be multiplied by -1 in these specific moments where there is a sign change and a continuous signal can be obtained.

The process to find the unwanted sign change can be identified by multiplying two adjacent values in time of the Euler orientation signal. If the value is negative and its

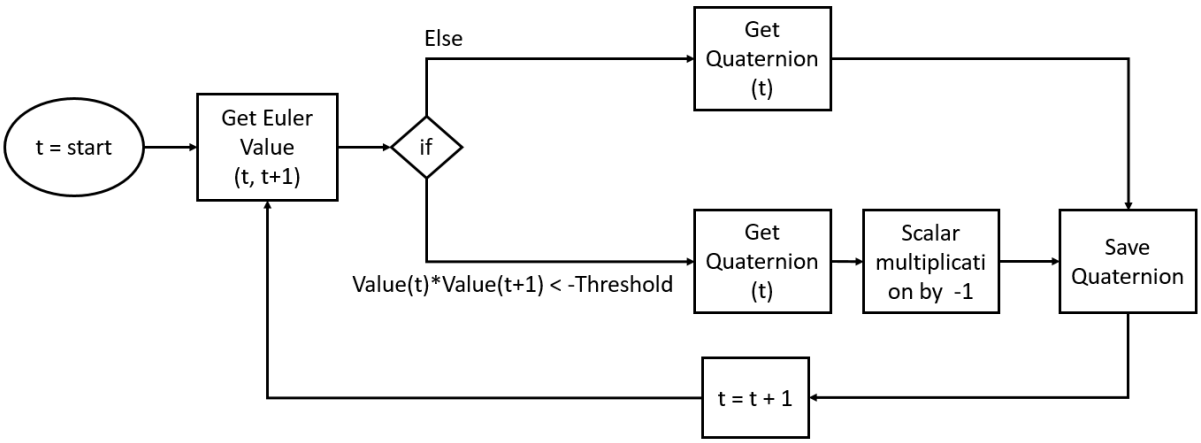

Figure 4.7: Euler angles transformation into quaternions processing algorithm

magnitude is high then there is an unwanted signal jump. The magnitude is important to consider since there can be changes in orientation near the 0 value which are by any means desired to keep. Figure 4.7 shows the flow-chart of how the logic of this process is done to get the quaternion signal.

4.2 Control Phase

The system, as mentioned before, initially learns a trajectory. It goes through the learning phase until all the parameters needed to run the primitives are computed and stored in a library. Once this happens, the operator can use the control phase to make use of the dynamic movement primitives.

In this section, an overview of the implementation architecture is explored. The different control modes of the current framework are presented and explained. These modes are selectable by the operator depending on its needs; named as robot-guidance and autonomous mode.

When working on robot system, there is a need to consider the architecture needed for the path planning. In the current robotic system, the control laws are already designed so delving too much will not be part of this work. Nevertheless, an overview will be provided to understand how the robotic system uses dynamic primitives.

The dynamic movement primitives are normally used as a trajectory generator in the control loop (Figure 4.8). Continuously creating desired points of positions, velocities or accelerations throughout the task's duration to accomplish a goal. The signal of desired positions will be fed to the different modes and then into the robot system. Furthermore the error is taken into account and fed to the path generation block to accomplish a stopping compliant behaviour. Furthermore, other inputs can be added into the DMP generator which can be discussed in future works.

Initially, the operator needs to select a control mode for the operation of the robot. These are defined as either the experimental robot-guidance or the autonomous mode. Afterwards, a selection of primitives must be done from the library. These can involve just one or several different ones, depending on the user's needs. Once this is done and all the parameters are set, the system can be started in its control loop and the trajectory generator will start to feed the robot system of desired values.

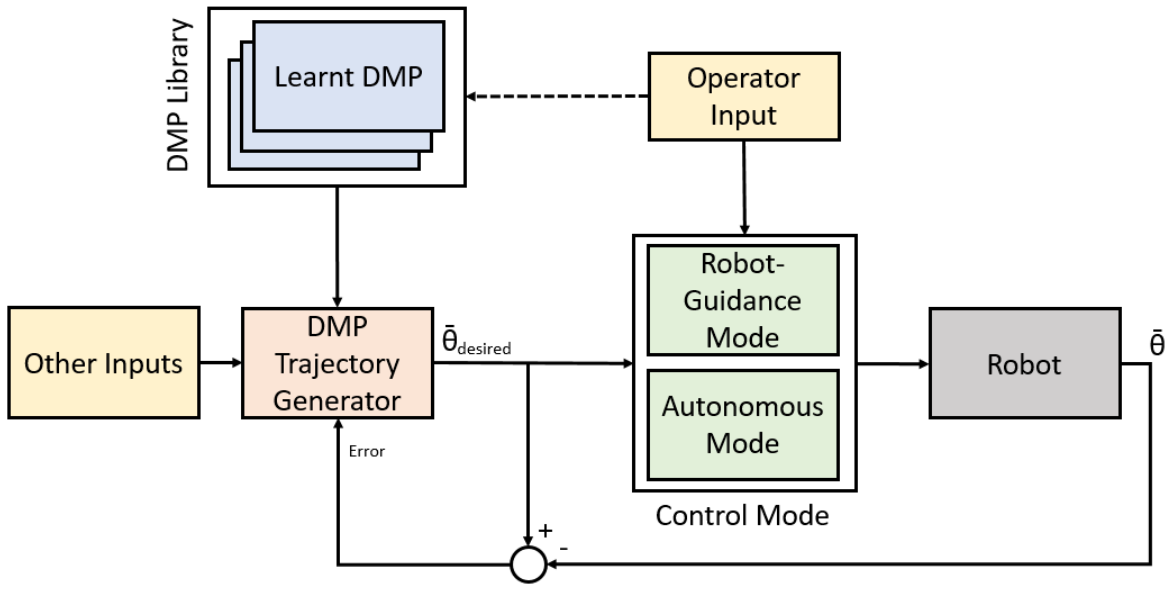

Figure 4.8: General structure of the control phase

4.2.1 DMP trajectory Generator

The trajectory generator is defined as the system which, from an input and parameters, will create the desired points for the robot control laws. Learnt dynamic movement primitives play an important role and its generation will be explained in more detail.

A block diagram of the DMP generator is shown in figure 4.9. In the beginning, some parameters have to be set. These can be the starting position, goal, decaying rates or weights. They can be chosen and changed depending on the task's needs or taken automatically from the learnt primitives. If they are changed, the system will adapt the outcome to match the new parameters, otherwise it will repeat the same dynamic as it was initially learnt.

The system then starts with the canonical values. They are fed to the transformation system's non-linear forcing term which will generate the desired kinematics for the robot.

The dynamical movement primitives depend on various differential equations known as the transformation and canonical system. Therefore, a numerical method is used to compute the values at each instance of time. For this work, an Euler method was chosen as it uses a simple finite difference approximation to compute the different values in time. The forward Euler method uses an initial value, already known, to find the next value in time (Equation 4.6).

$$x_{n+1} \approx x_n + hx'(t_n, x_n) \tag{4.6}$$

The function of $x'(t_n, x_n)$ is the derivative of x at the time n and h is a time step. The benefit of using this method is the fast computation and simple operations. Furthermore, it can be used for higher-order ODE by introducing extra variables. For example, in our case, the second-order equation of the transformation system (Equation 3.9) can be rewritten as two first-order equations. Hence, it is possible to redefine the state variables to Equation 4.7.

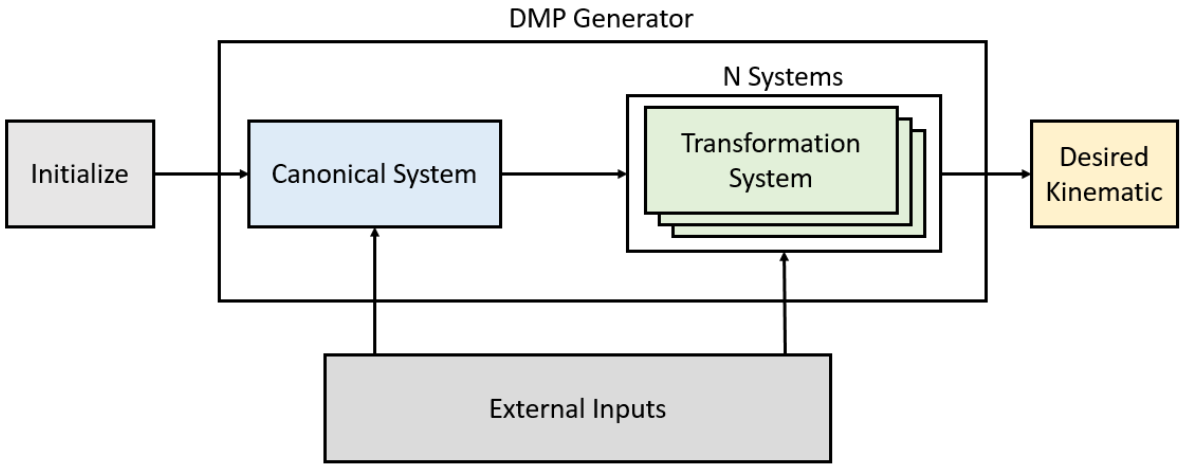

Figure 4.9: Block diagram for the DMP generator

$$\begin{aligned} \frac{d^2x}{dt^2} &= v' \\ \frac{dx}{dt} &= v \end{aligned} \tag{4.7}$$

Once again, we rewrite our transformation system with the new variable definition and make use of the forward Euler method for numerical integration in Equations 4.8.

$$\begin{aligned} v'_n &= K(g-x) - Dv - K(g-x_o)s + Kf \\ v_{n+1} &= v_n + v'_n h \\ x_{n+1} &= x_n + v_n h \end{aligned} \tag{4.8}$$

The h will normally be chosen of low magnitude to allow a good numerical computation. If it were to be chosen of high value then there will be a big divergence from the actual dynamics and give wrong results. This value will also be closely related to the time dependency of the system, since it is from this computation that the time is also defined.

Additionally, as shown in the Figure 4.9, there are external inputs to take into consideration. These are the dynamic changes in the parameters of the goal position, provided by Equation 3.19, and the error between actual robot response to desired trajectory values. The latter is taken to the α_s parameter to change the canonical system's dynamics through the Equation 3.17 and provide an enhanced compliance. However, there will be slight difference on its use depending on the selected operation mode.

4.2.2 Robot-Guidance Mode

This mode was inspired by works done in assistive and rehabilitation robotics such as done by Lauretti et al. (2017) and Chen et al. (2017). In this mode, the system becomes compliant with the human operator's input and is capable of reproducing learnt trajectories through human-robot interaction only.

The general purpose of this mode is to test the dynamic movement primitives through human-robot cooperation and familiarize the operator with learnt trajectories to perform specific tasks with the robot arm. For that reason, it is possible to test the developed path planning and the several different implementations done for this work (e.g. dynamic stopping mechanism) in a constantly running control loop. Furthermore,

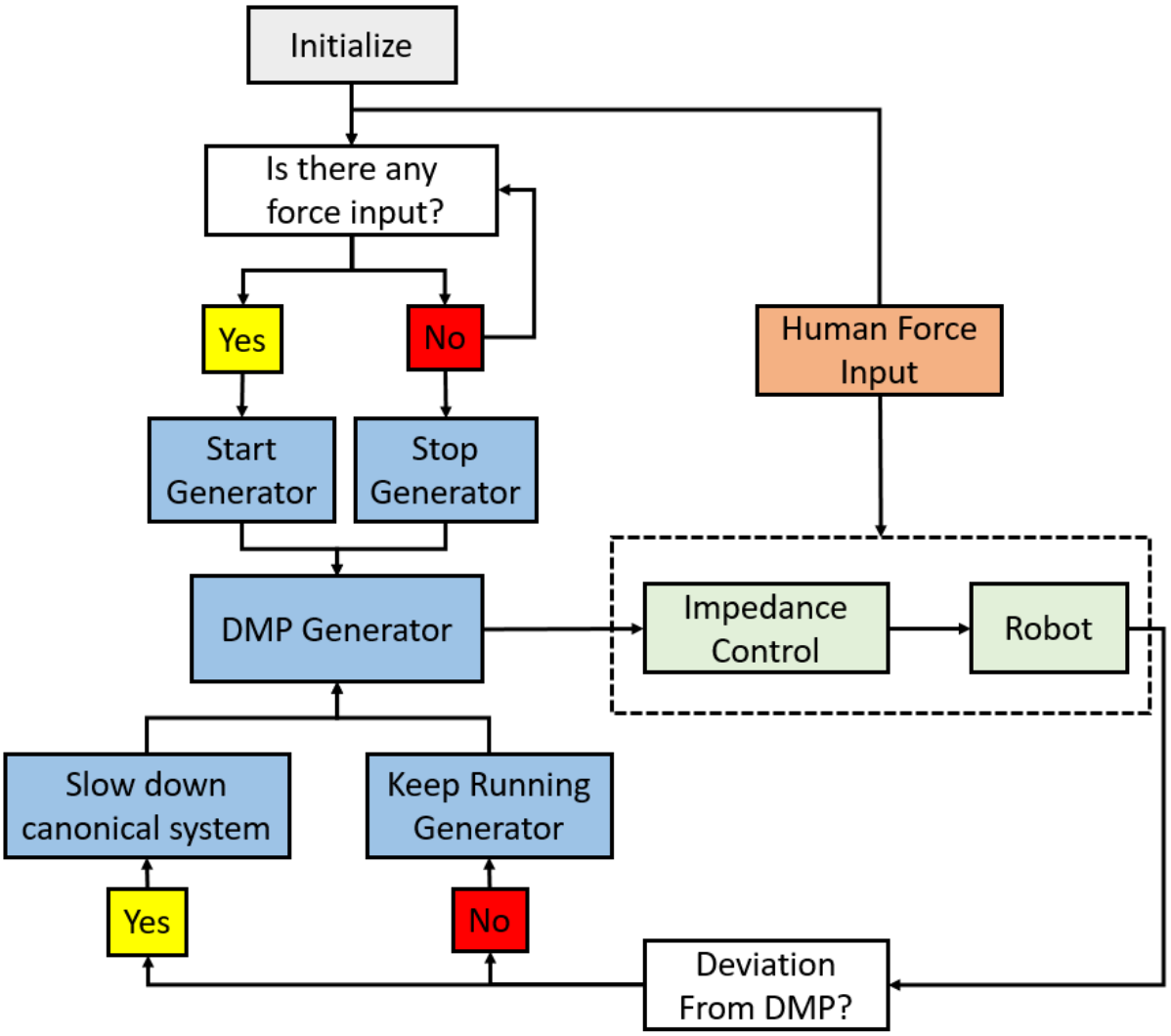

Figure 4.10: Logic flow-chart of robot-guidance mode

this mode represents the implementation of known primitives framework into other kinds of applications to test its adaptability into new projects.

Hand-Guidance and Robot-Guidance Mode

Robot-guidance mode is developed based on the hand-guidance mode presented earlier in subsection 4.1.1 which was used as a teaching method. The latter is done through the use of a joint impedance control architecture which allows the user to perform smooth motions with the robot's body. In the control phase, the idea will be used as a base structure but with some modifications to adapt for the purpose of robot-guidance mode.

Initially, in the hand-guidance mode, the input of the impedance control relies only on the interaction between the operator and robot. Basically, it just follows the operators motion. On the other hand, the robot-guidance mode takes into consideration the dynamic movement primitive generator (DMP generator) in the control architecture to create a more complex behaviour.

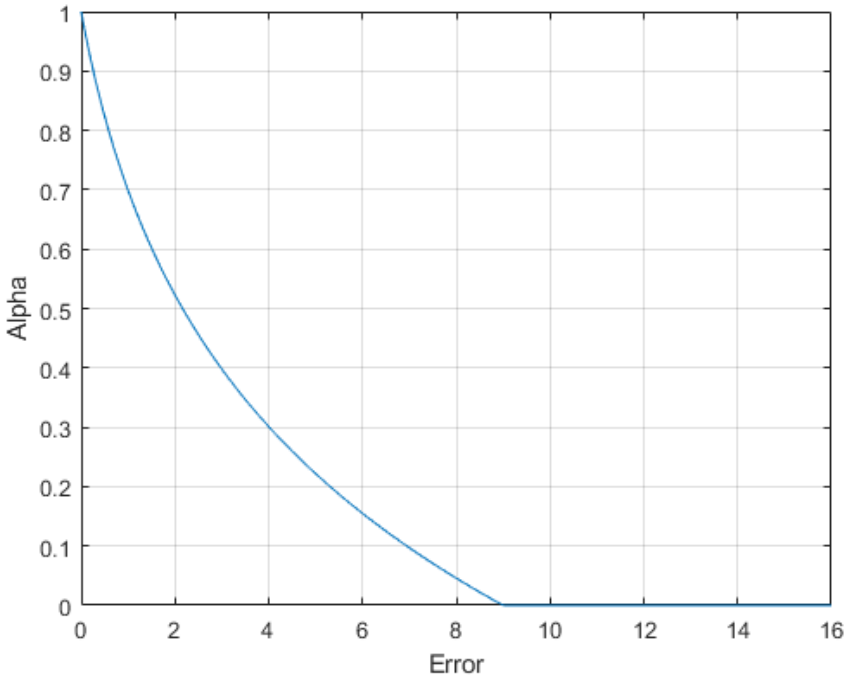

Figure 4.11: Dynamic change of α_s with saturation at 0

Mode Architecture

The working logic of this mode is implemented as shown in Figure 4.10. There are two elements which allows the creation of points from the DMP generator. One is the detection of an input force. In this case, the force will be detected in the robot's end-effector. Second, there is no deviation between robot and the last command sent from the system. As a result, the robot can only move if the user applies a force and there is no high deviation values.

The use of impedance control can provide a very compliant behaviour depending on the parameters of stiffness and damping that are used. Nonetheless, it also allows for the user to move the robot in unwanted directions. It is desired that the robot follows a learnt trajectory, therefore an error that relates the robot's actual position and the path's points is considered .

To compute an error value that comes from the robot's movement and the DMP generator, a coupling term will be implemented. This value will be fed to the canonical system through the Equation 3.17 and will decrease the generation of points by modifying the decaying rate of the canonical function. Additionally a limit of the value 0 is set so the decaying rate does not become negative 4.11. As a result, the generation of desired points will be linked by the current robot actual position and user interaction with the system.

This mode's outcome is a behaviour that will move the robot depending on its learnt trajectory and the operators input force. The desired positions will come as the user interacts with the robot, nevertheless if the operator deviates from the original form of the primitive, a decaying parameter (canonical) will tend to zero and the generation of points will quickly slow-down until it halts. The transformation equation's values will be set to no longer update new points and the robot will try to force itself into its last known position. However, due to the impedance control and the compliance it brings to

the touch of environmental forces (the operator's touch), the motion to move back into the original path does not feel harsh and it can be very intuitive.

If the operator moves with the robot, it will follow the initial learnt trajectory, otherwise it will not move and stay put in the last known position.The operator will then be able to familiarize with the learnt primitive through its motions and also will be capable of assessing if the learnt trajectory is optimal for the task at hand.

4.2.3 Autonomous Mode

This mode was developed to accomplish tasks by setting parameters and using learnt behaviours. The system will run selected dynamic movement primitives in a continuous loop until it achieves its goal. Furthermore, it also uses the capabilities of the implemented framework such as goal dynamic change and compliant behaviour based on the impedance controller and the enhanced canonical system.

Task Initialization

For accomplishing a task in the current proposed system, first a task must be set and defined. This means, that several primitives must be developed first and then used accordingly to achieve a goal. To do so, the learning phase is used as necessary to get the needed primitives to perform the desired activity.

To begin using the autonomous mode, there is a initialization phase that must take place. Here the task is defined and the trajectories are selected. It is important to know that the task can be a result of using several different sub-tasks that need to be achieved in certain order, e.g. moving the TCP up and then down.

Using the DMP library allows that every primitive can be repeated, or even used in different conditions. Thus, changing global parameters, such as goal or starting position will result in an adaptation of the primitive. This adaptability, as it was already explained, comes from the transformation system. Additionally, the pattern will be similar due to the modified properties of the differential equations.

From the DMP library the parameters should be identified and modified accordingly to achieve a desired behaviour. This process is demonstrated at Figure 4.12 in general terms.

In the example figure, the desired pattern will be formed of three sub-tasks defined by DMP A and C. The initial global parameters for the first sub-task (DMP A) are found ok with the initial desired path. However, C's learnt parameters do not match the current criteria in the wanted sub-task. Therefore, C's goal position are modified to adapt for the new circumstances (DMP C*). Furthermore, the third sub-task is just a repetition of the first one (DMP A), but the goal position has been changed to achieve the new desired path (DMP A*).

Task Control Algorithm

Once the task structure has been defined, the program can take this data and start the robot's control algorithm . The system has been initialized, now the DMP generator will be started from the parameters taken in the task structure and commands are created from the transformation system. These are then fed into the control law of the robot system which will work with an impedance control. A block diagram description is presented in Figure 4.13.

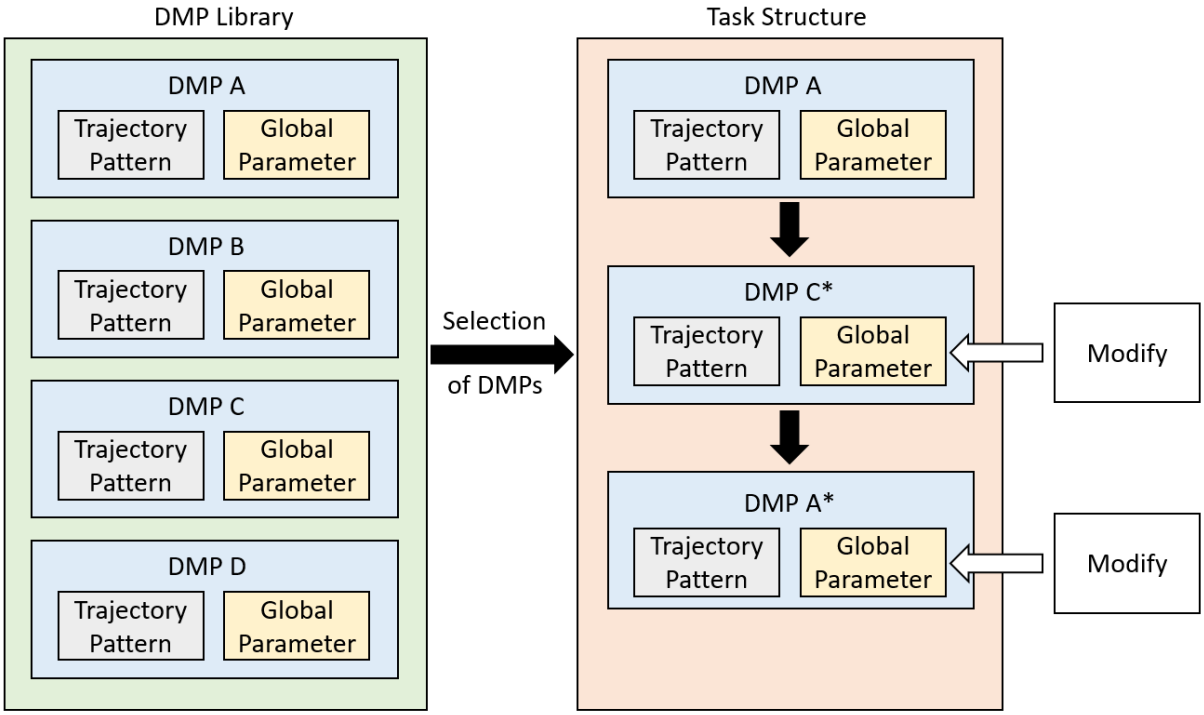

Figure 4.12: Example diagram for task structure definition using a DMP library

The system will repeatedly perform a the DMP control loop to generate the trajectory and constantly use a function that outputs the success of a sub-task. In this case, the euclidean distance between current position in every degree of freedom (θ_n) and its corresponding goal value (Equation 4.9) is used to find if the sub-task has been achieved or not . When all have reached a distance less than a threshold, the task or sub-task can be considered a success (4.10).

The function that defines the accomplishment of a sub-task, or task, doesn't necessarily have to be a distance function. It can also use different parameters such as feeling a force with the TCP force sensor or when an object has been detected by a camera.

$$dist_n = \sqrt{(\theta_n - goal_n)^2} \tag{4.9}$$

$$\text{if } (dist_n < \text{Threshold}) \rightarrow \text{Goal Achieved} \tag{4.10}$$

Once the sub-task goal has been reached, the algorithm will move on to the next sub-task in the task loop. It uses the last robot's position to update the starting point of the next DMP parameters. Once this is done, the DMP control loop will start its cycle again. This is repeatedly done until all the selected primitives are performed.

The control system is done through an impedance controller to add a compliant behaviour when there is some sort of disturbance coming from the environment. It allows for an error to be introduced to the DMP generator and it minimize any kind of damage it might come from obstacle contact by using the canonical coupling term.

Based on the equation 3.17, variations of α_s will be dependent on the error. This will be usually taken by the difference between the position of the robot and primitive. However, due to the use of an impedance controller, an error will constantly reduce the transformation system's velocity. To avoid this behaviour, we take into consideration a force threshold that the system must feel before activating the decaying function variation.

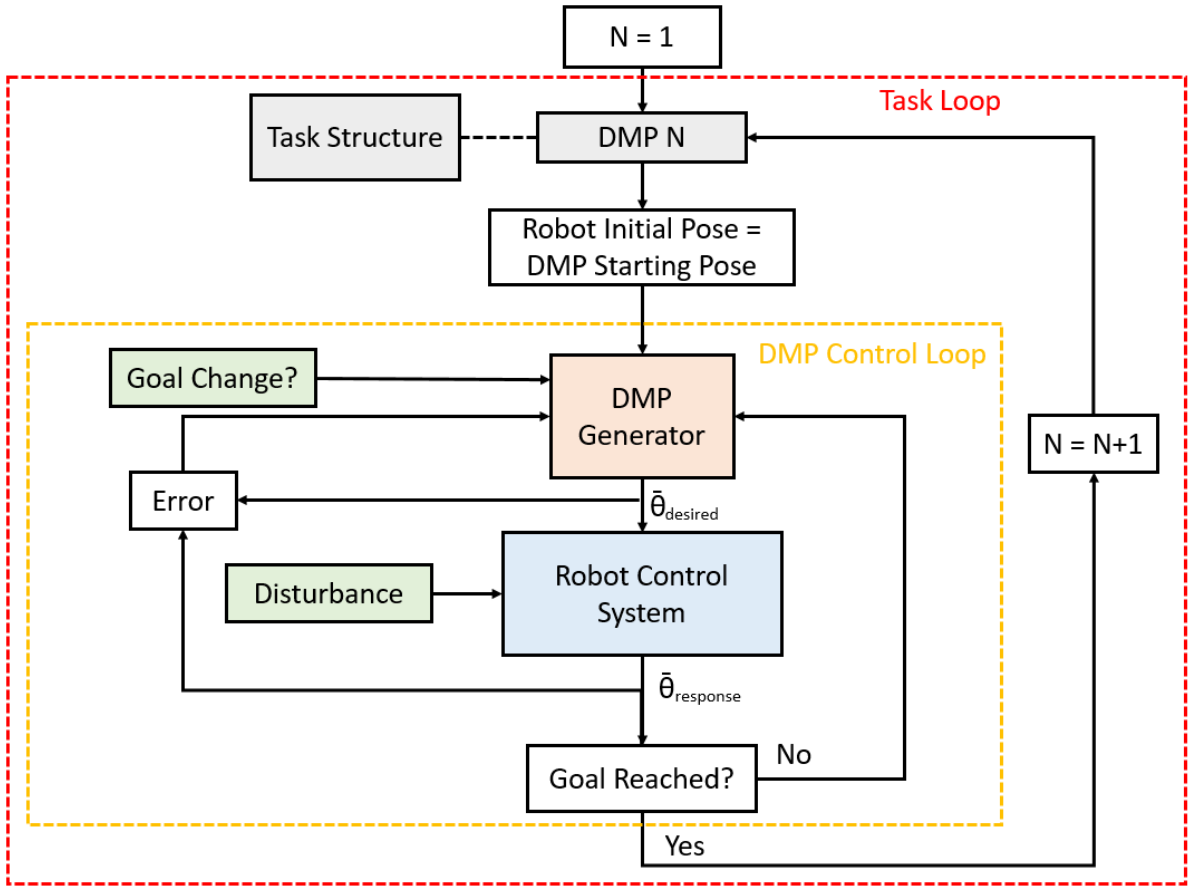

Figure 4.13: Task control loop and DMP control loop block diagram

This will consider the forces on the system and therefore, if nothing is felt, even if there is an error between primitive and robot, the system does not try to slow down.

Figure 4.14 describes the logic flow-chart when a force is detected by the sensor. If it is identified a force and deviation, the parameter will be changed according to the defined dynamics. In the beginning it will slow down the system. When an error manages to cross the logarithmic function's 0 point a collision has been detected by the system.

During collision, the α_s will be defined as negative and will be used to start the robot's return movement. This means, a retracing of the motions already performed by the robot. Once a certain time has passed, the system will stop entirely, and the operator can intervene in the situation. Either by moving the robot into a different place, actuating on the obstacle or even changing the primitive to use for this particular task.

Additionally, for the DMP generator there's an input that allows the change of goal position for each running primitives. The DMP generator will use the dynamics explained in Equation 3.19 and create a smooth transition between goal values. This will avoid any high differential changes in the transformation system equations and will create a transient without high accelerations or spikes.

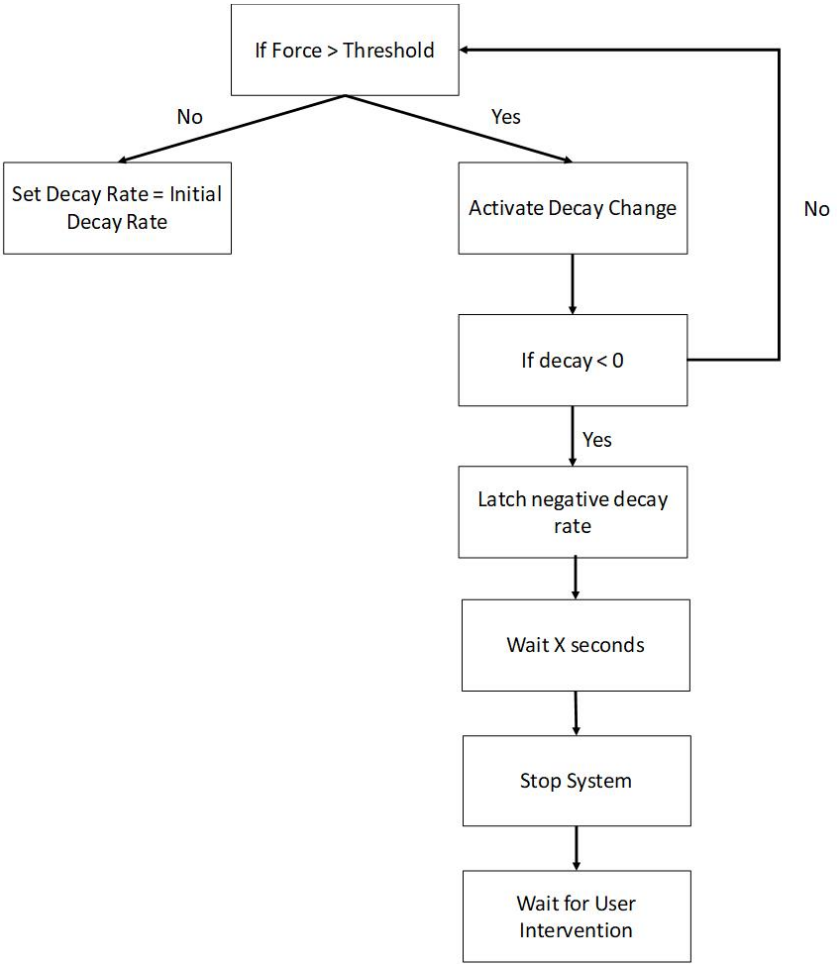

Figure 4.14: Autonomous mode decay rate variation logic flow-chart

Chapter 5

Experimental Results

In following chapter the proposed learning framework's validity is tested. First an overview of the working environment will be described. Test with simulations are also done to view the relationship between learned trajectories and executed ones. Following, the framework is used in a real robotic system. the following are the tests that are presented with no relevant order.

- Force interaction in Hand-Guidance's mode.
- Changing goal parameter through the DMP Generator's.
- Human-robot interaction in robot-guidance mode.
- Single primitives tests in autonomous mode.
- Obstacle collision response in autonomous mode.
- Unscrewing a connector through high level task selection in autonomous mode.

5.1 Workspace environment

In order to achieve the objective of this work, the proposed learning framework will be implemented through the use of a robotic arm that can fully perform the desired tasks. In this section an overview of the robotic system will be presented with its characteristics and capabilities.

5.1.1 Kuka LWR Iiwa 14 R820

For this work, the Kuka LWR Iiwa 14 R820 robotic arm is presented (Figure 5.1). The system is classified as a lightweight robot by KUKA Roboter GmbH (2016a) with 7 different axis as shown in Figure 5.2 and its structure is arranged as an in-line wrist, join modules and a base frame.

Its design focus is to obtain a safe Human-Robot collaboration robotic system that is most commonly used in an industrial setting. Signals of positioning, torque and temperature with many different locking mechanism to assure safety in its operation.

Kuka offers its own system software with a pre-built library which has several different functions ready to use (KUKA Roboter GmbH (2016b)). Its custom made operating system is known as Sunrise and is designed for fast application development in an

Figure 5.1: Kuka LWR Iiwa 14 R820

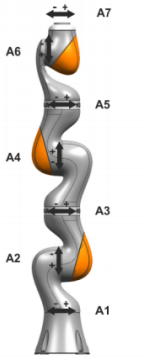

Figure 5.2: Joints location

industrial setting. Additionally, the interface's language is based on Java Runtime Environment.

Joint Limits

As shown in Figure 5.2, 7 joints are used in this robot arm. They are identified by the letter A following the joint number. These work in a limited range of angles which is defined by the Table 5.1, where the highest rotational range is 340 degrees, and the lowest is 240 degrees.

Joint No.	Limits (°)
A1	±170
A2	±120
A3	±170
A4	±120
A5	±170
A6	±120
A7	±175

Table 5.1: Joint Limits For Kuka Iiwa r820

This is an important information to consider, since beyond these ranges the robot internal software will force a stop to avoid damaging the joints.

Robot's task space and joint impedance control

The impedance control for this system is used in joint space, where it takes the joints' position to create a spring-damper dynamic interaction with the environment. It is used to obtain compliant system and make it safer to detect obstacles in the robot's path that are can come in direct contact with the robot.

It is vital to use joint space for this control scheme since it will allow more flexibility in the robot arm. Resulting in a safer motion due to a compliant behaviour in each of the joint and not only in the end-effector as it is done for the impedance control using task space.

The task space describes the end-effector's position and orientation in world coordinates ($[x, y, z, \alpha, \beta, \gamma]$). And the DMP generator can make use of these coordinates to learn more intuitive trajectories that can be modified. Knowing a world coordinate position can be

useful when coupling other systems and makes it easier to understand the data. Therefore, in order to use both joint impedance control and task space, the inverse kinematics (IK) of the robot needs to be computed.

The solution of a high degree robot arm's IK is complex and not trivial since there are infinite solutions to a specific robot arm pose. In this work, the implementation of an IK algorithm is not needed since the robot's software already has a pre-built library which uses its own algorithm. The already used IK is based on the redundancy angle of the robot (A3 axis) and the wrist position (interjection of joint's axis A6 and A7).

5.1.2 TCP's orientation to quaternion

To obtain good results in the learning algorithm, the signals must be always continuous. Otherwise, the relationship between input and output will be lost. In order to avoid this, the orientation of the TCP which is in Euler form, must go through the algorithm demonstrated at Figure 4.7 in chapter 4. The representation with quaternions will help in generating a smooth signal for the controller.

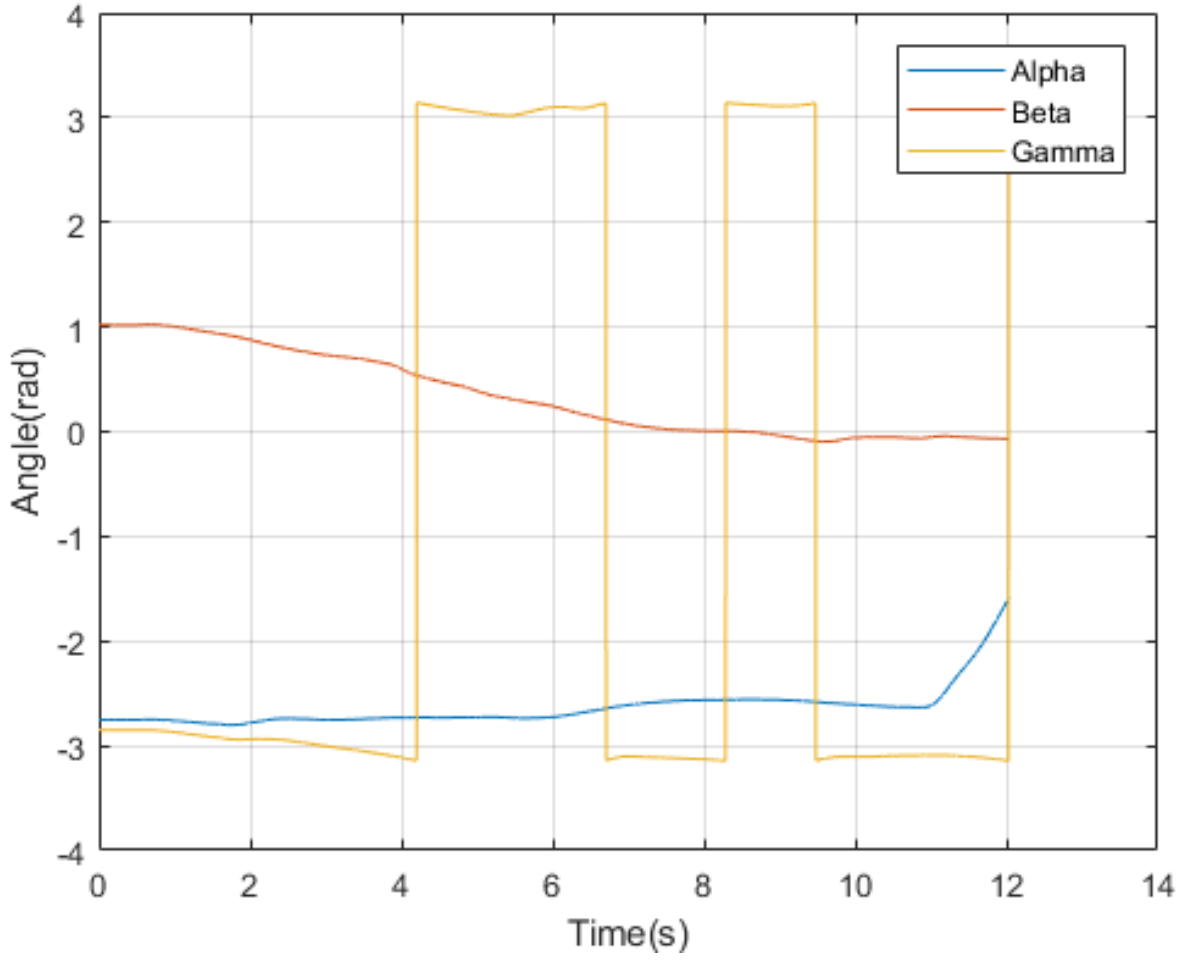

Figure 5.3: Robot's TCP Euler's Orientation trajectory

It is possible to visualize the jumps in the gamma (γ) orientation between π and $-\pi$ radians in Figure 5.3. When the values are transformed in quaternions, these jumps are also apparent in the new orientation space as shown in Figure 5.4, and as expected all the parameters (q_w, q_x, q_y, q_z) jump as well.

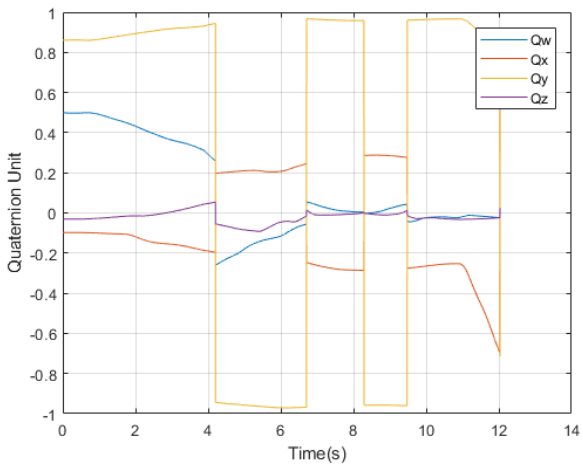

Figure 5.4: Quaternion's Orientation trajectory transformation before applying the continuity algorithm

Now the algorithm in sub-section 4.1.2 is used for these demonstrations, and the resulting signal becomes the one shown in Figure 5.5.

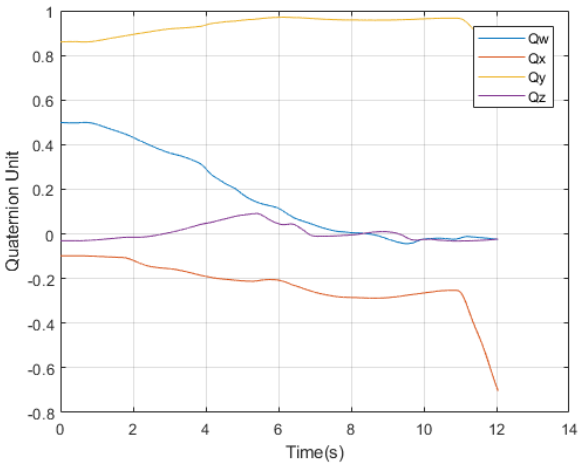

Figure 5.5: Quaternion Transformation with signal continuity algorithm

In the new orientation space, the signals are all continuous as expected. The data is now usable in the learning algorithm which will allow the use of task space in the robot arm.

5.2 Hand-Guidance Mode low force interaction

When using Hand-Guidance Mode, due to the use of a very low stiffness and damping coefficients in the impedance control it is possible to have a low force interaction with the robot. The parameters were selected very low and they are presented in the Table 5.2. Axis A1 to A4 present a higher stiffness because the force that can be exerted in these joints can be higher. This because the operator usually moves the TCP, which is the farthest part of the robot arm and will create a higher torque on the joints close to the base. Therefore, to avoid involuntary movements from the operator into a these axis,

the impedance values of these are slightly higher than the others. As a side note, the parameter for the damping is related to the Lehr's damping ratio. In these tests an operator uses the presented mode to move the robot in different directions. The position values of joints were recorded and the force in the TCP, which is the point of contact between the robot and operator, as well.

Joint #	Stiffness ($Nm/radian$)	Damping(unit-less)
A1	2	0.9
A2	2	0.9
A3	2	0.9
A4	2	0.9
A5	1	0.9
A6	0.5	0.9
A7	0.5	0.9

Table 5.2: Hand-Guidance Impedance Control Parameters

A trajectory is presented in Figure 5.6. Altogether, the force magnitude was recorded as well and is shown in Figure 5.7. A force applied to the TCP as it is the location where an operator can constantly apply the motion.

The robot-operator interaction in this test has a duration of 20.85 seconds and the mean force reflected on the TCP is of 3.8921 Newtons.

Several other tests were executed and are presented as the Table 5.3. As an average, there is a force of $5.2219N$ to move the robot by using its TCP. This mode results in an easier way for the operator to teach a specific motion through its body. It is important to note that the operator could also move the robot on its joints directly, which can make the TCP force lower than expected.

Most movements were done with no difficulty in the presented hand-guidance mode, however, there were small exceptions when trying to move from specific poses. These were near singularities where the robot could not move a joint to achieve a desired motion's direction. In these cases, the operator had to do a different kind of movement to leave the singularity. It was noticeable the most in demonstration # 7, where the operator was presented with a robot pose near a singularity. Resulting in a higher force applied to the TCP.

Demonstration (#)	Mean TCP Force (N)	Duration (s)
1	2.8512	12.10
2	4.6584	5.45
3	3.6004	4.84
4	7.6220	3.12
5	5.1200	10.56
6	2.1340	15.77
7	10.2253	16.64
8	4.2546	8.21
9	3.4568	9.01
10	8.2983	4.98

Table 5.3: Mean TCP Forces in various demonstrations

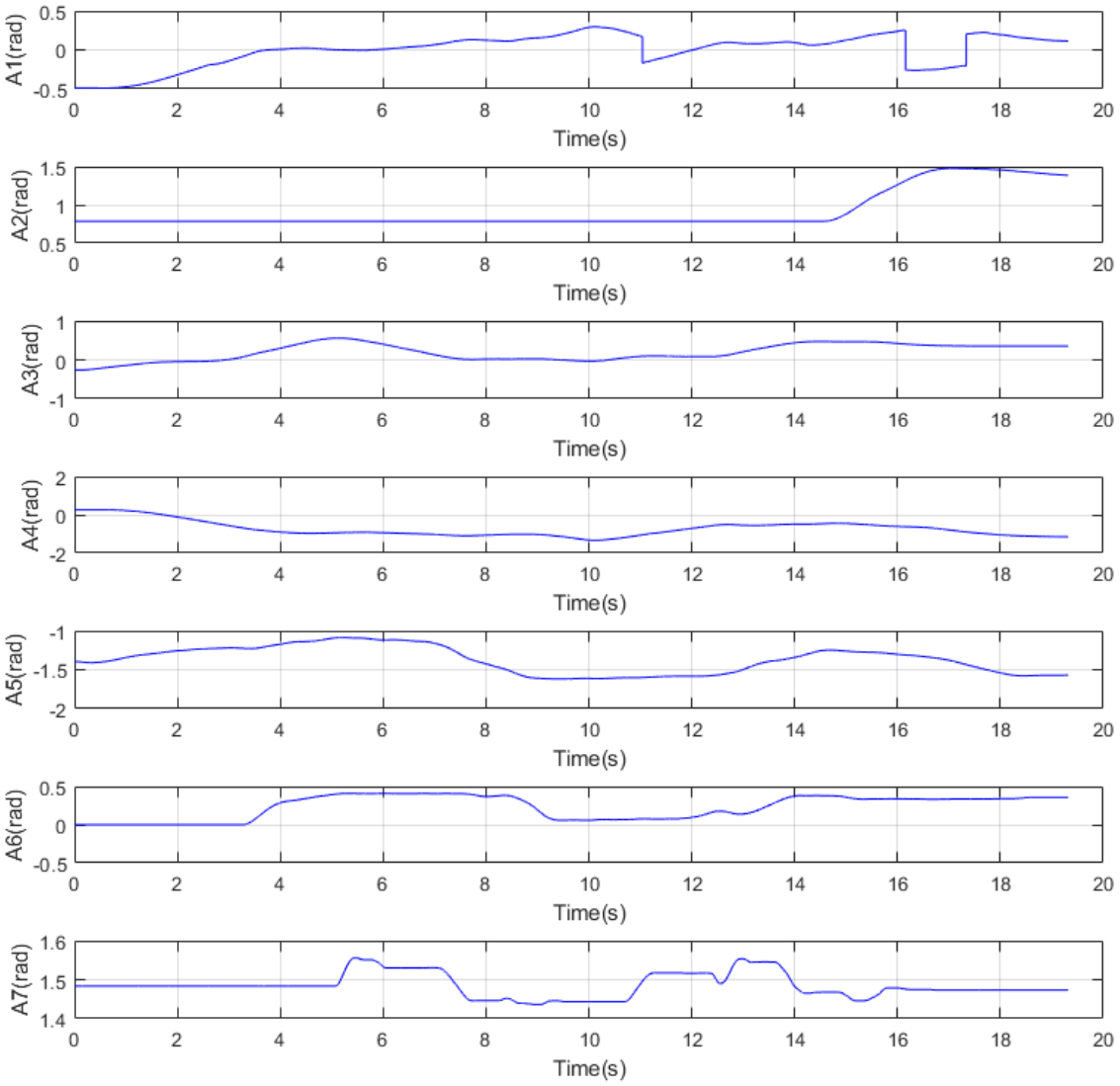

Figure 5.6: Joint's Motion in Hand-Guidance mode. In decreasing order, each graph is a respective axis of the robot arm (A1-A7)

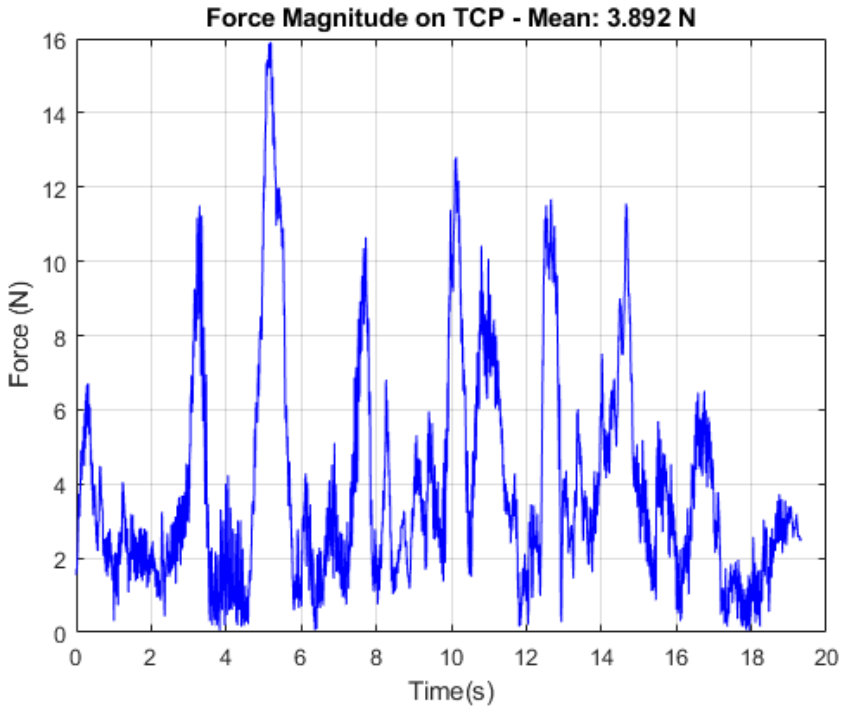

Figure 5.7: Hand-Guidance mode force magnitude on TCP during learning phase's kinesthetic teaching

5.3 Learning Dynamic Movement Primitives from demonstration

In this section, the hand-guidance mode will be used to learn a desired trajectory. A primitive will be created from the demonstrated data and its results will be presented.

Stiffness (K)	Damping (D)	No.Kernel	Kernel Bandwidth (h)
50	80	50	1.0

Table 5.4: Parameters for learning DMP

The following are results obtained from creating a primitive using 13 degrees of freedom with a duration of 6.75 seconds. The variables correspond to the position in world coordinates of the TCP and robot's arm joint configuration. The parameters used to learn this primitive are shown in Table 5.4. These values were chosen empirically through observation of data and behaviour of the system. They are selected due to the good results that were obtained, given by having no oscillations in the response and low error.

The signals errors were computed with the root mean squared error (RMSE), which can give an overview of how different is the signal between each other. The data comes from a robot demonstration as previously mentioned and the results are obtained by the dynamic movement primitives transformation systems output.

In Figure 5.8, the RMSE mean is 0.0351 between all the joint parameters. Nevertheless, there are some cases where the demonstration had very small movements which was not properly learnt (A3). Because of the system's smoothing property through both the kernel functions and differential equations, these small movements are filtered out which results

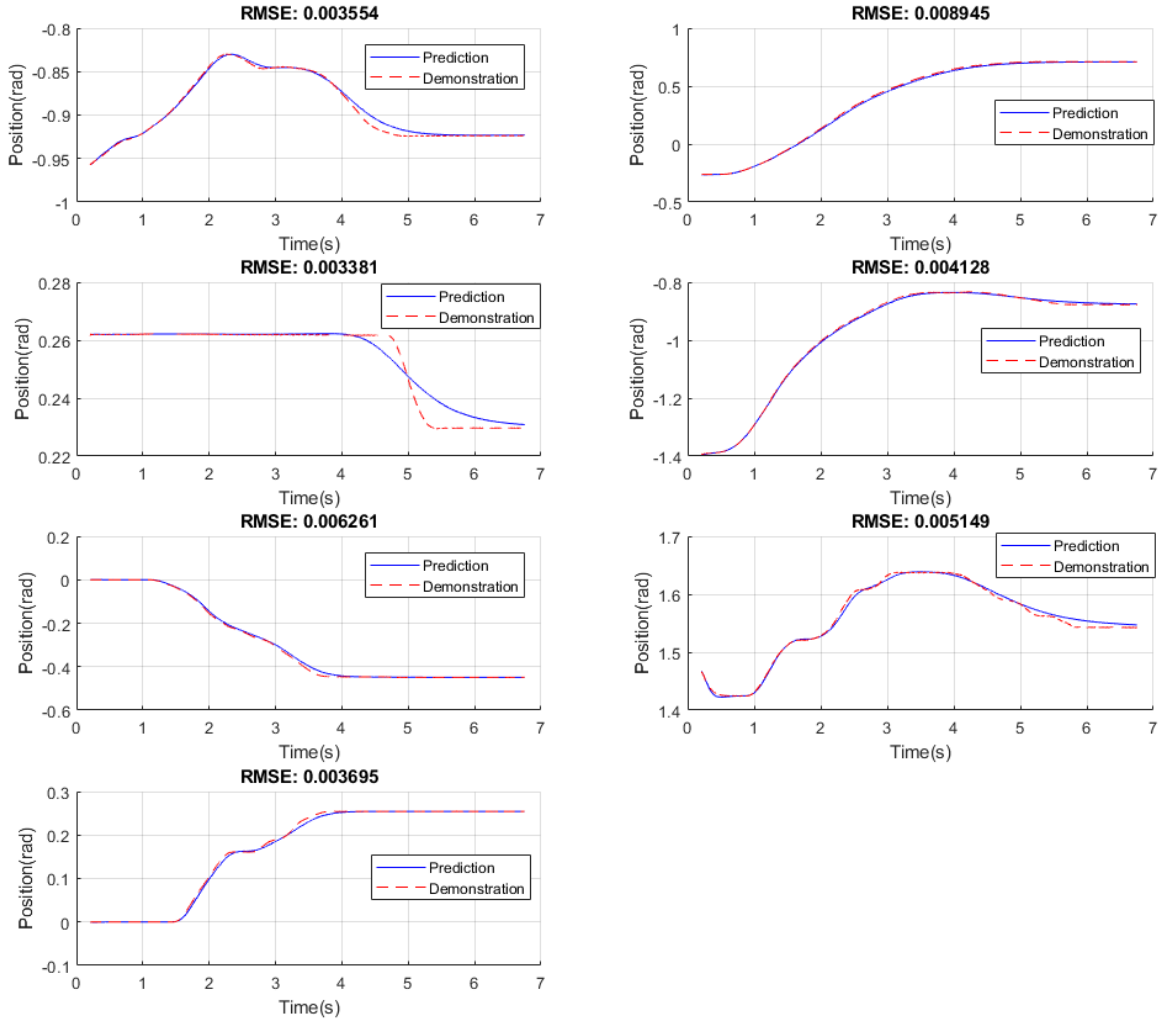

Figure 5.8: Joint Demonstration and predicted signals. From top to bottom, the left side are odd number of the robot's axis (A1, A3, A5, A7), and the right side are the even number (A2, A4, A6)

in the observer behaviour. On the other hand, the other joint predictions have small errors and they also smoothed the demonstrated trajectories.

The Figures 5.9,5.10 and 5.11 show the TCP's position world coordinates (X, Y and Z). And Figures 5.12, 5.13 and 5.14 are orientations. The starting points in world coordinates are $[266.3mm, -184.8mm, 823.3mm]$ and the goals are $[435.6mm, -520mm, 557.1mm]$.

The mean RMSE altogether is 3.3690, which makes it a precise motion, in Cartesian space, of the primitive. It can also be seen that the demonstrated signals have small oscillating motion un some parts. These are due to the operator's teaching movements and is an expected element when performing any kind of kinesthetic learning with a very low stiffness robot. Nevertheless, the predicted signals are all smoothed out due to the dynamic movement primitives' properties (kernel functions and differential equations), and these oscillations are not noticeable in the predictions. Furthermore, the canonical system links all degrees of freedom and each signals reaches their destination.

The Kuka Iiwa robot can store data for up to 1 ms, and depending on the systems computational load, the value can increase up to 40 ms. This needs to be taken into consideration when choosing the number of kernel functions and the bandwidth. If the

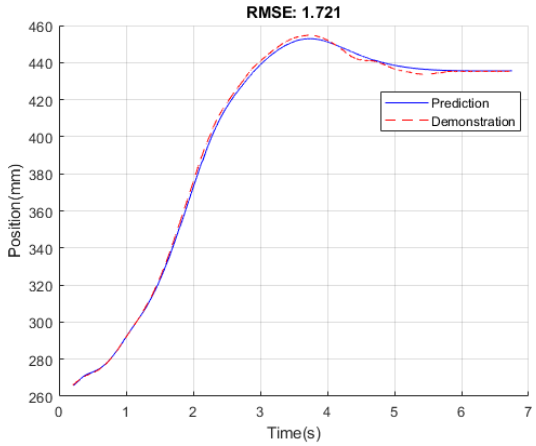

Figure 5.9: TCP's X position prediction

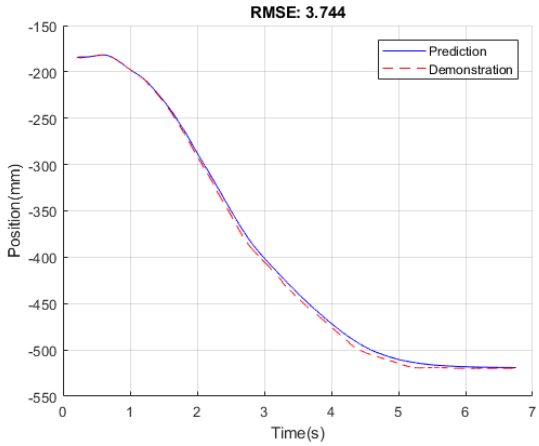

Figure 5.10: TCP's Y position prediction

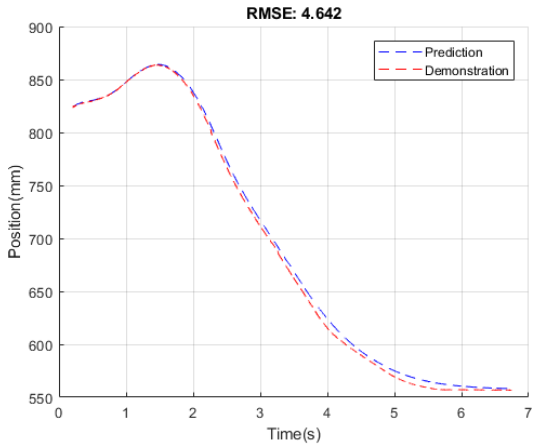

Figure 5.11: TCP's Z position prediction

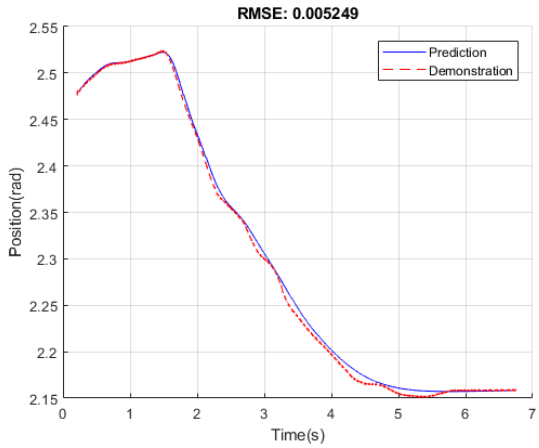

Figure 5.12: TCP's Alpha orientation prediction

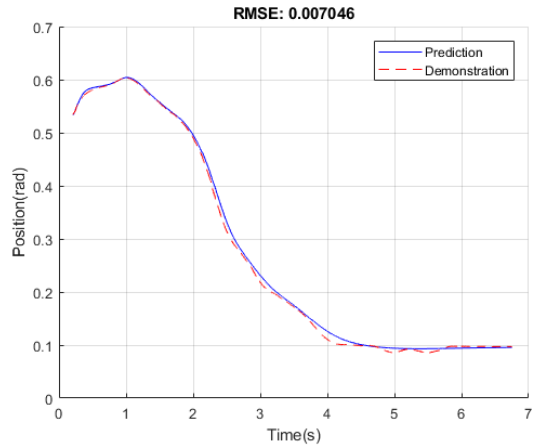

Figure 5.13: TCP's Beta orientation prediction

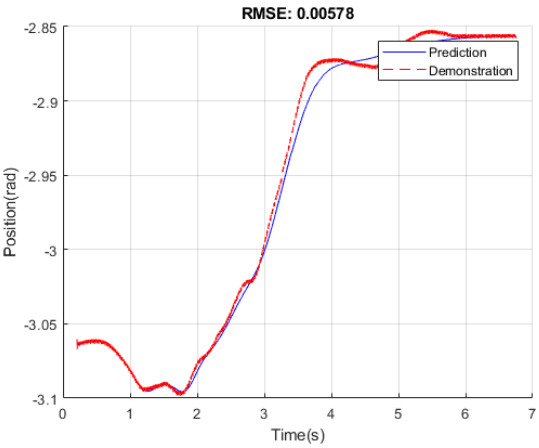

Figure 5.14: TCP's Gamma orientation prediction

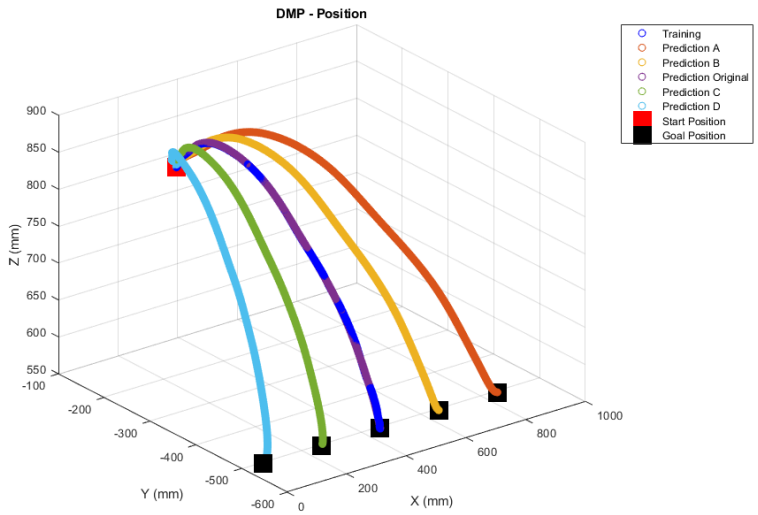

Figure 5.15: Variation of goal position with single primitive in 3D coordinates system

system's data logging frequency is high and precise then a high amount of kernel functions can be used with a low bandwidth. This way, the pattern can be reflected by the kernel's weights and the predictive signal will be smooth as well.

5.4 Dynamic goal change in DMP Generator

Using primitives allows to change dynamically the goal values of the system. This test will be using already known DMPs and apply two different changes. First the global parameter of the goal (g) will be changed to different values. Second, the goal change will be done while the system is executing the DMP generator.

For the tests, the DMP will use 3 degrees of freedom that represent $[X, Y, Z]$. This way it can be observe with more ease the behaviour of the DMP generator when running. Furthermore, the goal change will be done along the X axis with steps of 200 mm each.

For the first test shown in Figure 5.15, the original goal value is defined in $[435.4mm, -519.7mm, 556.8mm]$. The goal values for each prediction is defined by the black points and the system starts at the same position, denoted by a red point. The affected axis is shown in Figure 5.16. Here is more visible the dynamic adaptation to the new goal conditions and the error it implies.

The Table 5.5 shows the values set for the goal and the resulting prediction from the primitive at time of 6.75 seconds. It shows that the further away from the original primitive, the slower it takes the system to reach. Eventually, the DMP generator will generate the set value due to the transformation system's convergence property, but further in time. The maximum error was of 9.66 mm from a 400 mm change of the original goal's trajectory.

In Figure 5.17, a single primitive is executed in the DMP generator. Each path starts on the same position [-31.76 mm, 67.53 mm, 829.38 mm], denoted by a red square, and its original goal is the position [707.07 mm, 327.29 mm, 389.11 mm]. The system, in this case, will present a sudden change of the goal position while performing the DMP, and it will be indicated with a black arrow in the figure. The variation of the goal position is done along the X-axis in steps of 200 mm.

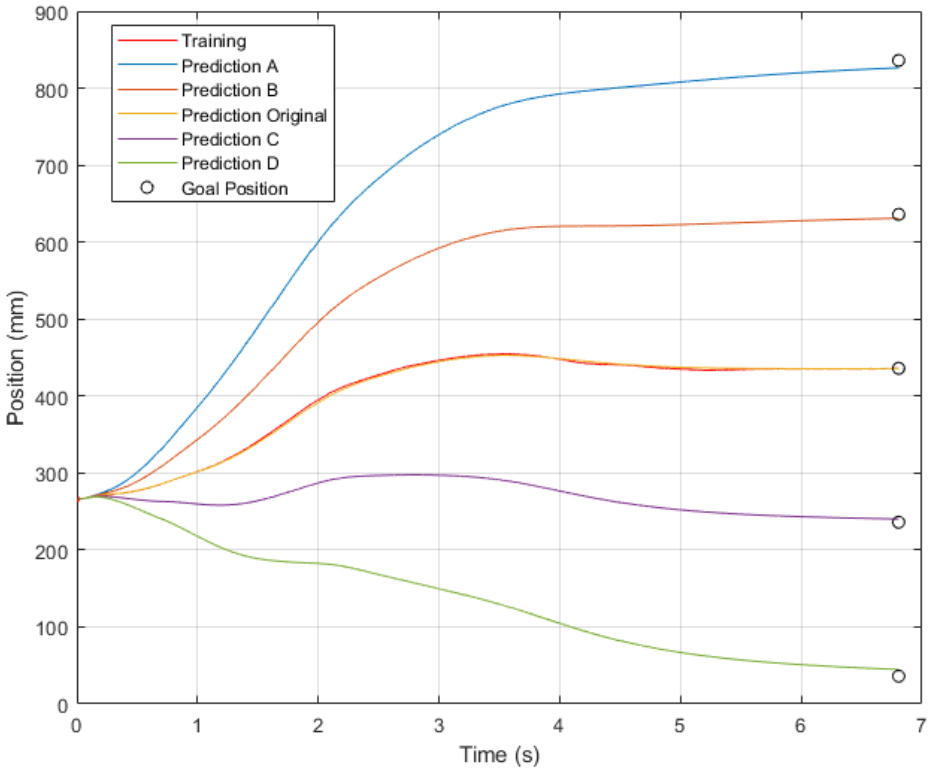

Figure 5.16: Variation of goal position with single primitive in X-axis

Prediction	Set Goal (mm)	Reached Value (mm)	Error (mm)
A	836.06	826.4	9.66
B	636.06	631	5.06
$Original$	436.06	435.6	0.46
C	236.06	240.2	4.14
D	36.06	44.82	8.76

Table 5.5: Goal changes values and their errors at the expected finishing time

It is observable how each sudden transition is done for each prediction and the adaptation of new conditions is not erratic or unstable. By using the dynamic change of goal presented in Equation 3.19, the trajectory's generation is smooth and it adapts to the initially learned shape which has some sort of "s" shape. Nevertheless, as mentioned before, there are two elements which must be taken into account with the new adaptations. When adapting to new conditions using a high-degree of freedom robot like the Kuka Iiwa r820, it is possible to compute an unreachable position. For example it can be a singularity location or a position out of reach by the robot's structure. Therefore, the real applicable adaptability of the primitives is reduced by the robot's kinematics in this current framework. To avoid this problem, several dynamic movement primitives must be trained to take into account these specific locations.

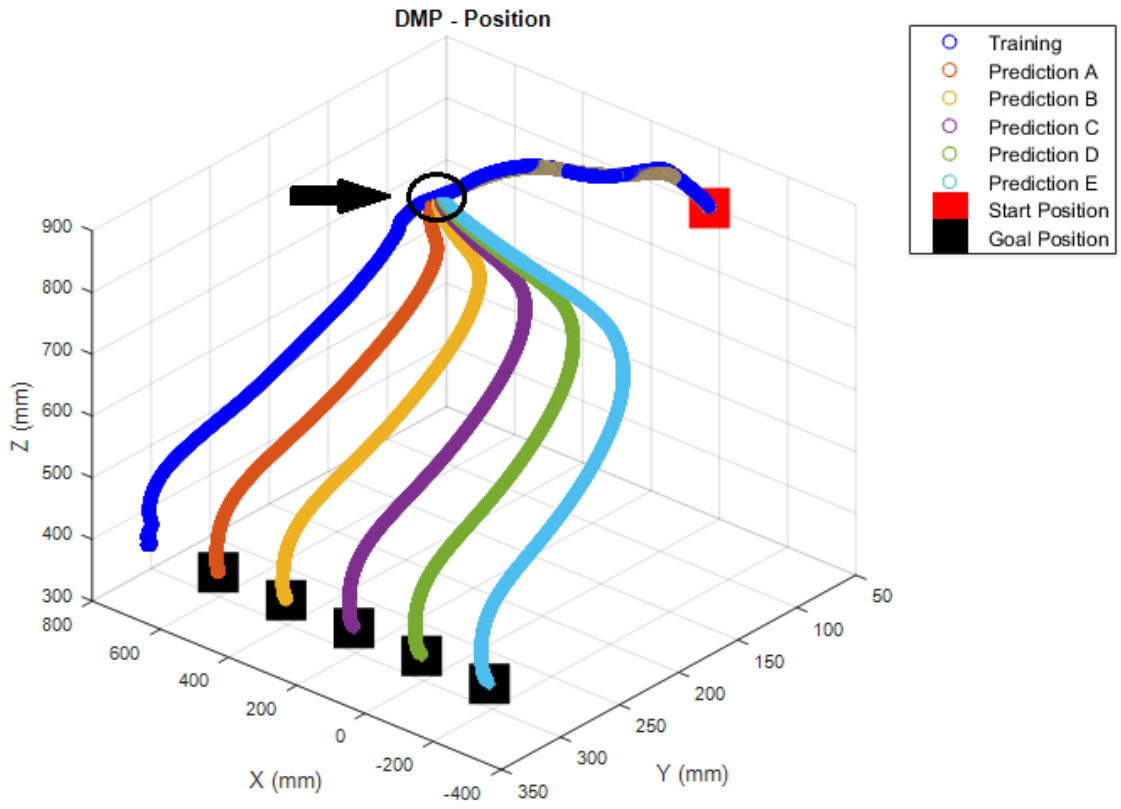

Figure 5.17: Goal position change online modulation

5.5 Robot-Guidance Mode

The experimental compliant mode was developed to further utilize the capabilities of the proposed framework and to test the replication of dynamic primitives through the human-robot interaction. Mostly performed as an experimental mode for testing new dynamics and behaviours.

This test will consist of using the before mentioned guidance mode. The system is initialized with a learned dynamic movement primitive. The beginning of the guidance begins when an user touches the robot's TCP. It will indicate that the guidance can begin and therefore, the DMP generator will start producing command set-points to the controller.

Figure 5.18 is the force signals in the duration of the test. First the user will apply a force and try to feel where it should go. The robots impedance force altogether with the DMP generator interacts with the user and it "pulls" towards the used primitive's direction. In the before mentioned figure, it is indicated by the normal guidance force.

Then the user will start to move away from the expected primitive direction and force it away. The system will detect this deviation and stop the point generation halting the movement of the robot. Nevertheless, due to the impedance control, the user can move the robot around but will feel a force that pulls towards the desired direction. In other words, the robot stops and tries to force the user to go back to the original direction. In Figure 5.18 this moment can be seen as the highest peaks of force in the signal.

After pulling the user will not move the robot and it will stay in place as long as the user doesn't interact with it. Afterwards, there will be another pull away from the direction

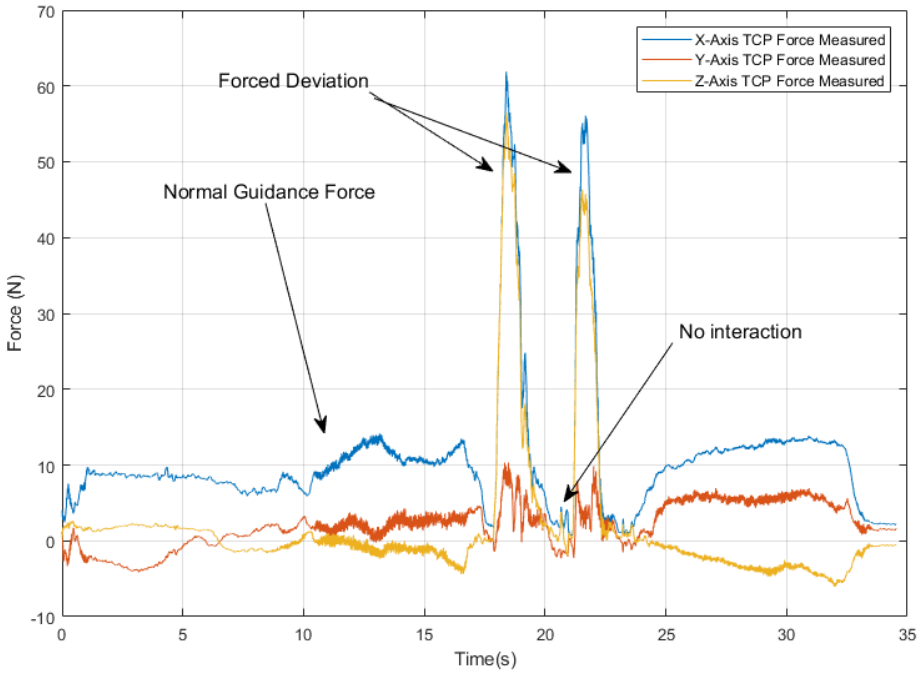

Figure 5.18: TCP Force during robot-guidance mode

of the primitive and the robot will not generate points away from its position. Following, the user will apply a slight force which the system will detect and start generating the next points to follow.

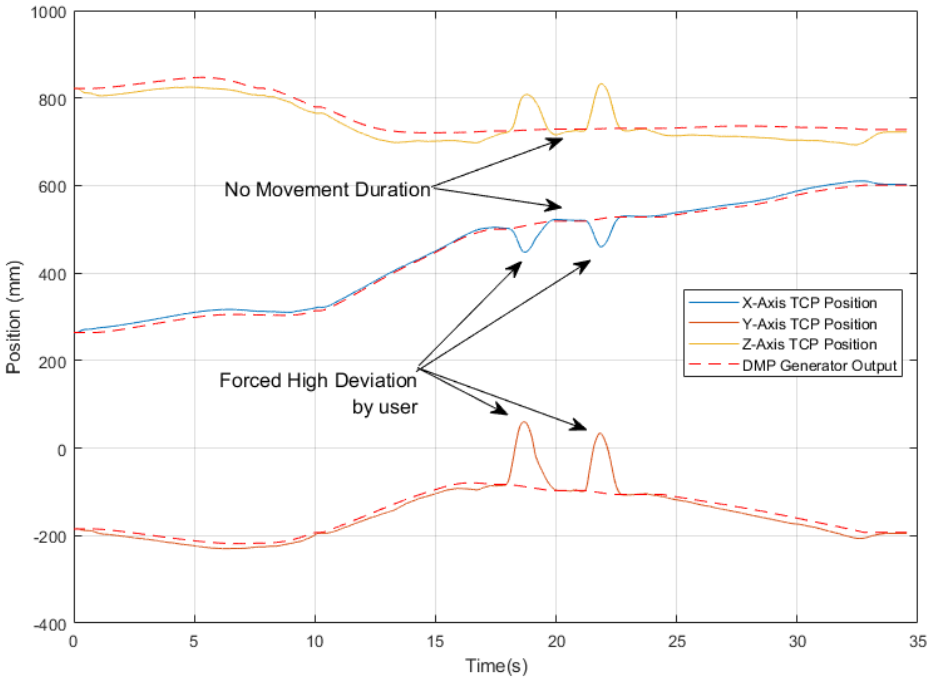

Figure 5.19: TCP Force during robot-guidance mode

Figure 5.19 shows the position of the TCP in world coordinates and it is visible how the DMP generator behaves and how it interacts with the user. It is important to note that the path generation is done by the use of the enhanced canonical system. If there is an error detected between the actual robot system and the desired points, the system will stop creating new points and wait until the position of the robot is closer to the primitives path. It differs than in autonomous where the value of α_s does not become negative, it is saturated at the value 0.

In the showed positions graph 5.19 it can be seen that during the robot's guidance there's always an offset from the DMP generator's commands. This is due to the impedance controller compliance which has low stiffness. As a result, the user doesn't feel a high force pulling but just a slight drag from the robot. Even though there is an offset, the path pattern is very similar to the desired one, which is represented by the learned primitive.

This test presents the robot-guidance mode as an application of the present dynamic movement primitives with the use of an enhanced canonical system to stop the generation of points when there is a high deviation. Showcasing the current framework's adaptability into other situations that might be useful.

5.6 Autonomous mode DMP test

The robot system will be used to perform a specific trajectory. Here, the user will teach the robot motion and the system will learn this motion and replicated through the autonomous mode. After the trajectories are performed in the learning phase, and the parameters are properly set. A new primitive is created. From here, the autonomous mode comes into play to use the parameters and perform the trajectory in the robot system.

This test was used on the Kuka Iiwa robot arm under the implemented autonomous mode. As mentioned before, the joint impedance control is used and the parameters of the controller were chosen as presented by the Table 5.6.

Joint #	Stiffness (Nm/rad)	Damping(unit-less)
A1	300	0.9
A2	150	0.9
A3	150	0.9
A4	150	0.9
A5	150	0.9
A6	150	0.9
A7	150	0.9

Table 5.6: Joint Impedance Controller's Parameters

The trajectories were taught before-hand through the hand-guidance mode, and the primitive was executed by the robot arm. The presented degrees of freedom are the world's coordinate at the TCP and its quaternion representation $[X, Y, Z, q_w, q_x, q_y, q_z]$. Starting from position $[136.8, 292.7, 821, 0.2362, 0.09794, 0.8349, -0.4877]$ and reaching the goal at $[593.6, -222.2, 368.8, -0.03794, 0.1293, 0.8711, 0.1293]$. The duration of this particular primitive is 15.93 seconds.

It is observable how the controller is following the DMP generator's output in Figure 5.20 and 5.21. Nevertheless there is slight offset after a couple of seconds, in

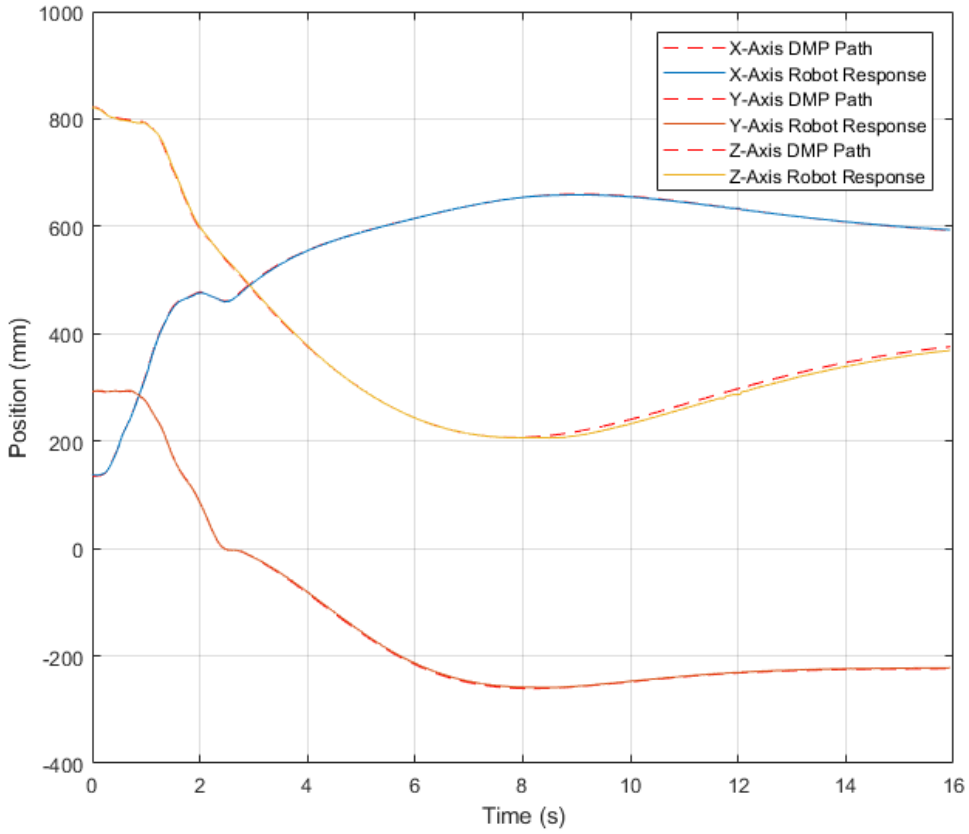

Figure 5.20: Kuka's robot response in world coordinates with a primitive

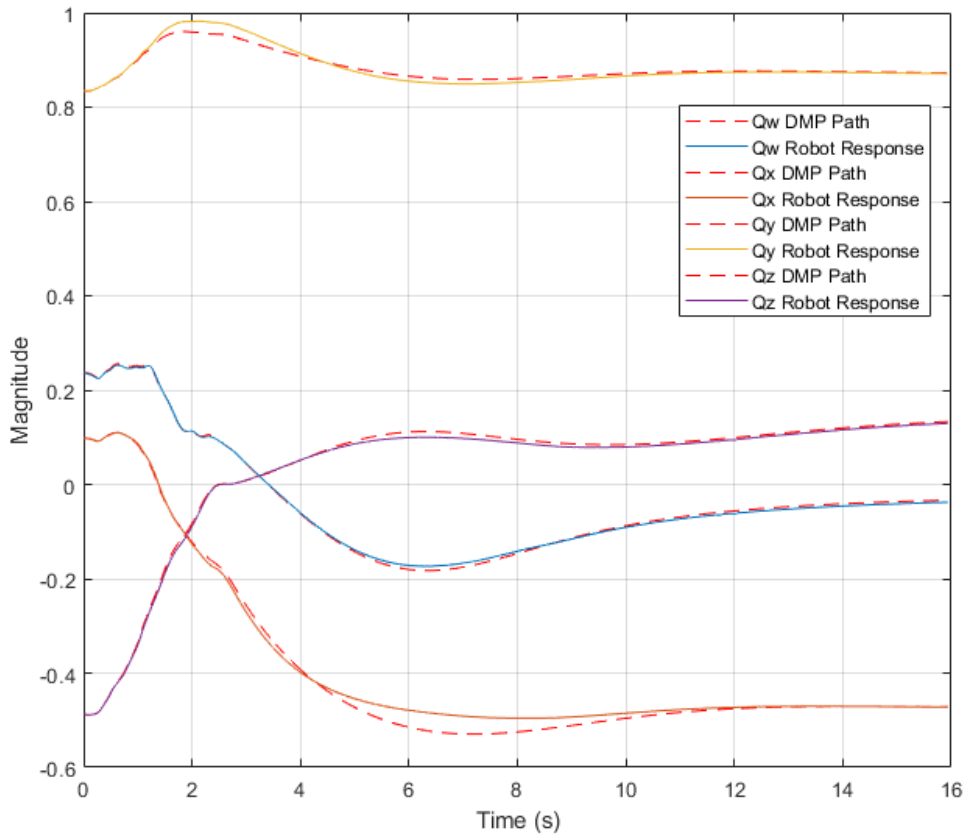

Figure 5.21: Kuka's robot response in Quaternions with a primitive

particular with the orientation. Due to the impedance controller, there is a tolerance of the position that needs to be defined by the system. This tolerance lets the robot have some leeway in the positioning response. If the stiffness were to increase then the position control can be more precise, but it also reduces the robot arm's compliance. Hence there is a relationship between the robot's precision and its compliance with the environment.

By using the impedance controller defined in Table 5.6, the resulting precision is very high as shown in Table 5.7.If the impedance parameter is reduced by 50% of the initial used value, to have a more compliant behaviour, the resulting precision is reduced as shown in Table 5.8.

State Variable	RMSE
X	0.9849
Y	1.4880
Z	5.3165
q_w	0.0046
q_x	0.0172
q_y	0.0094
q_z	0.0061

Table 5.7: RMSE of Dynamic Movement Primitive and actual Robot's position. Normal Impedance Values

State Variable	RMSE
X	2.1954
Y	3.2185
Z	3.2137
q_w	0.0060
q_x	0.0129
q_y	0.0152
q_z	0.0094

Table 5.8: RMSE of Dynamic Movement Primitive and actual Robot's position. Low Impedance Stiffness

Using the autonomous mode allows the system to perform specific primitives. Nevertheless, due to the Kuka's designed controller, the DMP Generator needed to slightly changes how the system should generate points. In this case, to create desired set-points at a reasonable rate, there was a time delay added in the program which allowed the robot to perform the desired commands. If this delay had not been added, the robot creates commands at an extremely high rate and jumps almost instantly to the trajectory's end. This time delay was added during generation of the command and constantly checks if the robot has reached the desired command or not.

5.7 Enhanced Canonical System stopping mechanism

This test is done to validate an enhance canonical system to stop the robot system during a short collision. The stopping mechanism was designed to make use of two

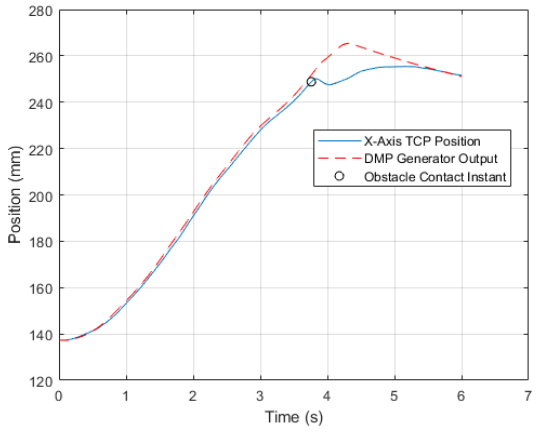

Figure 5.22: TCP's X position change in collision

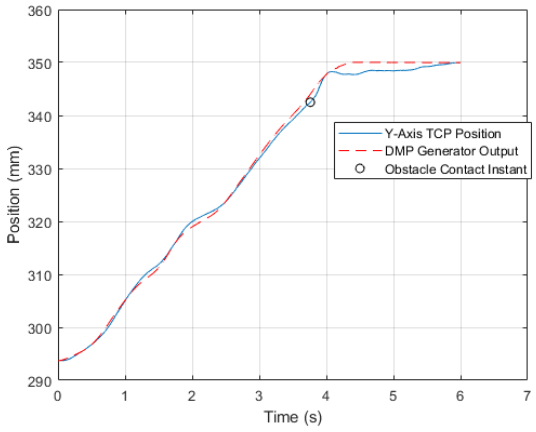

Figure 5.23: TCP's Y position change in collision

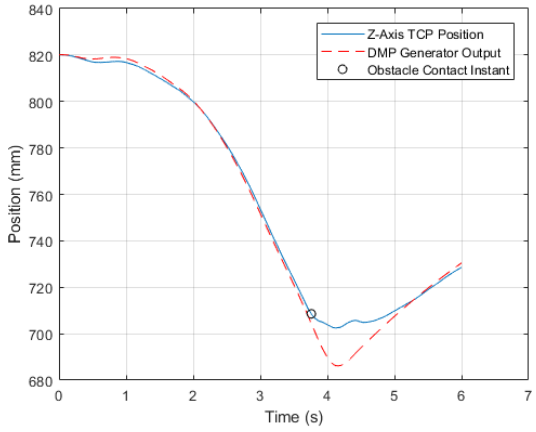

Figure 5.24: TCP's Z position change in collision

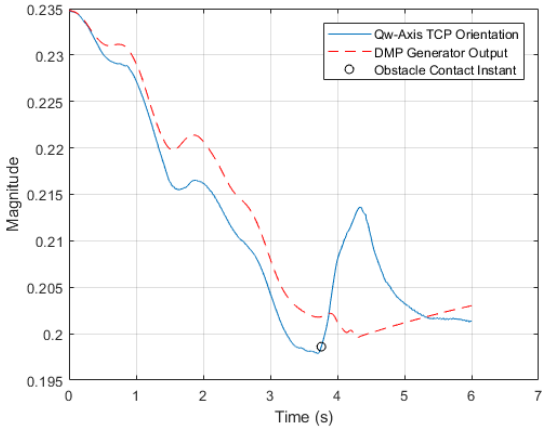

Figure 5.25: TCP's Qw orientation change in collision

elements. One is a modification of the canonical equation that takes into account the error between DMP generator and robot response, and second is force detection through the sensors. This test is done by putting an obstacle in the robot's path and observing the behaviour it produces when detected.

The test consist to initially let the system run normally with a specific primitive. Here the behaviour is that the robot moves from an upper. After approximately 3 seconds, an obstacle, which will be a stopping hand, will be put in the robot's TCP path and the behaviour will be observed. The system will have a force threshold that detects a collision condition and will activate the α_s parameter variation.

Figure 5.22 to 5.28 shows 7 graphs which contain each world coordinates of the TCP, both in XYZ and quaternions. The red line shows the DMP generator behaviour and the blue line is the actual TCP's robot position. At around 3.76 seconds the contact with a hand is detected and is indicated by a black circle in the graph.

There is a rapid deviation from the robot's position away from the expected DMP values, noticeable by the small bump in the TCP position. Most visible between coordinates X (Figure 5.22) and Z (Figure 5.24). This is due to the obstacle position in space and the direction in which the robot is moving to.

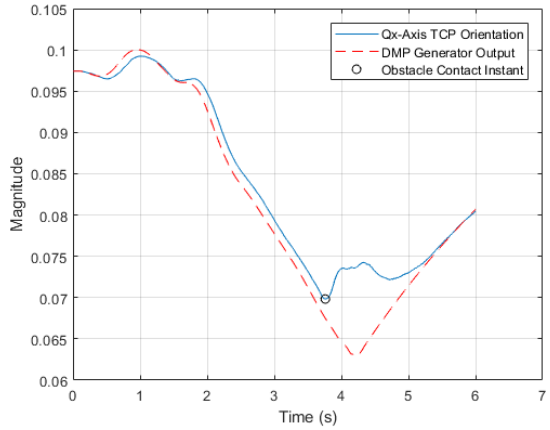

Figure 5.26: TCP's Qx orientation change in collision

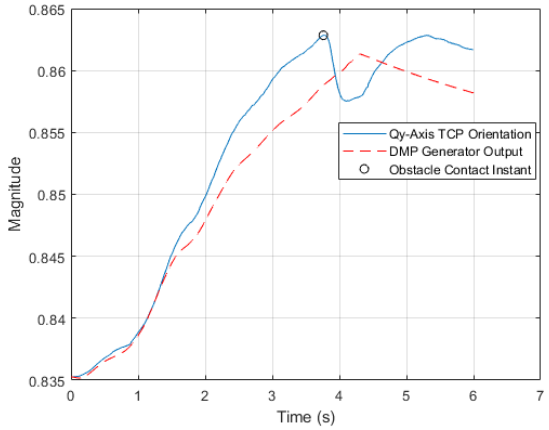

Figure 5.27: TCP's Qy orientation change in collision

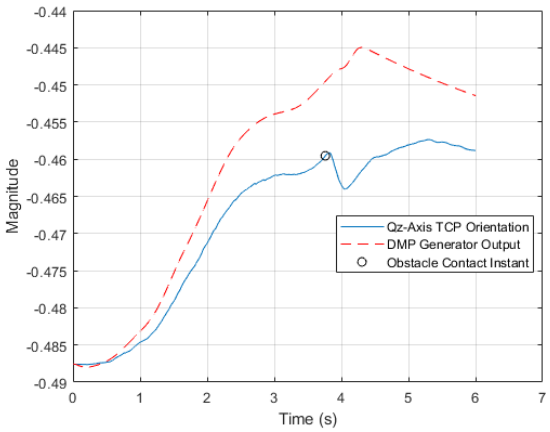

Figure 5.28: TCP's Qz orientation change in collision

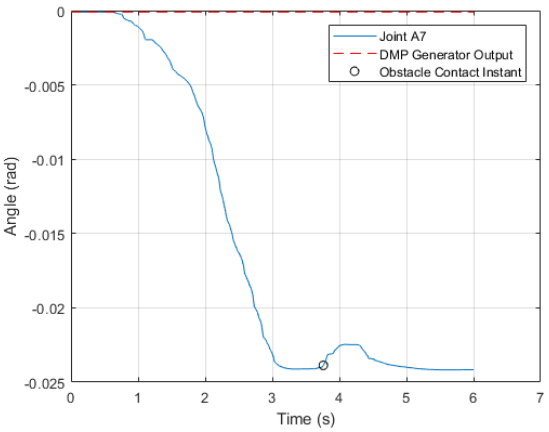

Figure 5.29: Joint A7 orientation change in collision

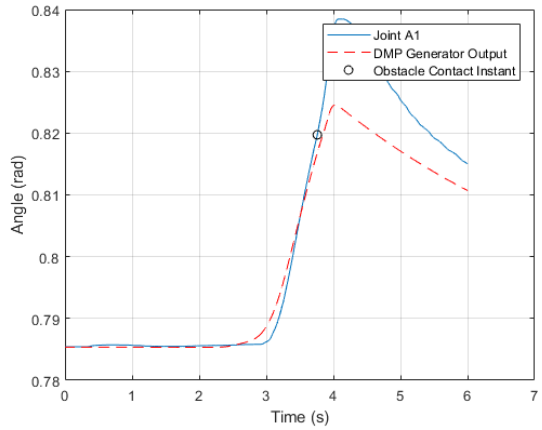

Figure 5.30: Joint A1 change in collision

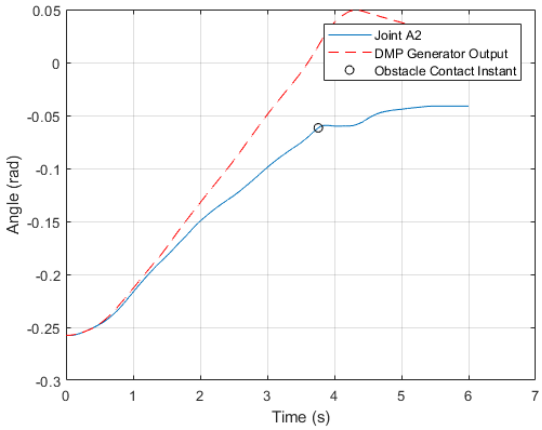

Figure 5.31: Joint A2 change in collision

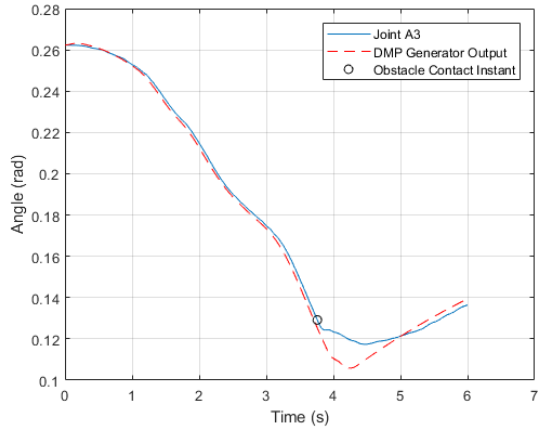

Figure 5.32: Joint A3 change in collision

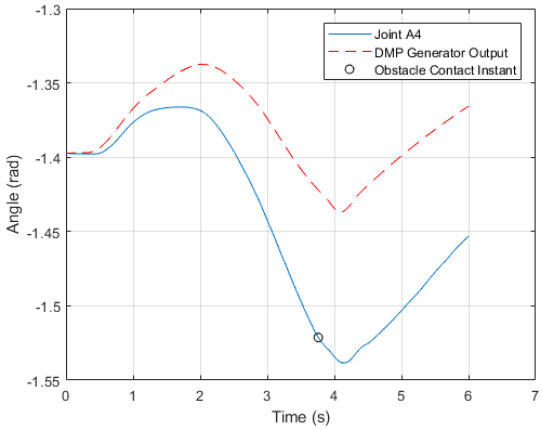

Figure 5.33: Joint A4 change in collision

It can be seen that in some axis even though the collision has happened, the DMP and robot response is moving without deviation. This behaviour is due to the robot being able to move in a position that the obstacle is not obstructing according to his learned path. Nevertheless, when using several positions to detect the collision, it creates a redundant detection where if several are failing, then everything will stop and try to move backwards.

The force detector is activated by a threshold and the DMP detects a collision. Using the dynamic equation presented before, the system takes a turn and stops. Notice how the DMP generator output in the graphs all change the direction they were going the moment the collision has been detected.

From Figure 5.29 to 5.30 the joint coordinates are shown. There are some deviation before the obstacle's contact, but it is expected due to the impedance controller's compliance. Similar than in Cartesian space, every joint-axis detects the collision and therefore the system starts to deviate from the desired position. The DMP generator responds accordingly and redirects the path to a different way.

When the obstacle is detected initially a force is felt by the sensor and an error rapidly increases. This affects the canonical system, and it slows it down until it stops generating new points. The error then becomes negative and the canonical system begins to move the transformation equation in the opposite direction due to the negative decaying rate.

It is important to note that when the robot starts to move backwards, the decay rate

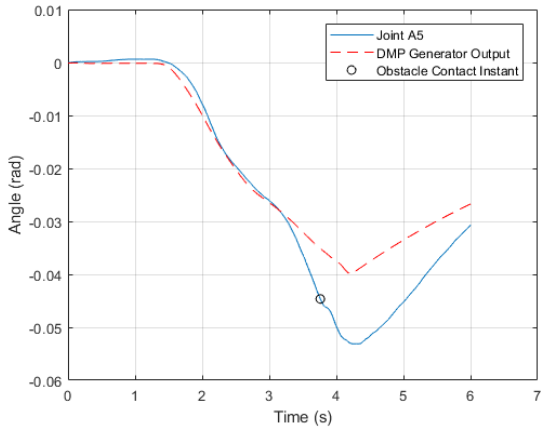

Figure 5.34: Joint A5 change in collision

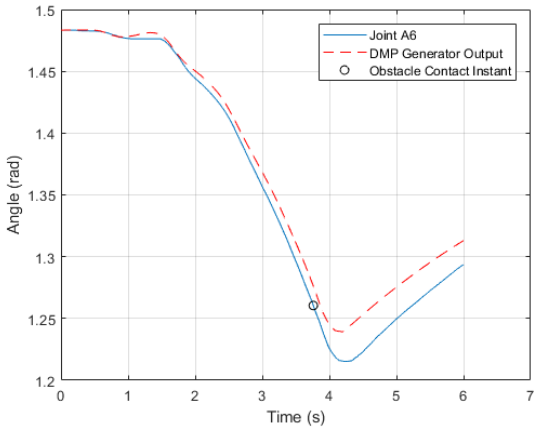

Figure 5.35: Joint A6 change in collision

must be latched into a negative value. Based on the Equation 3.17, when the error starts to decrease the decay value will start to be of positive sign. Then the system will begin moving forwards colliding once again with the obstacle. This is an undesired behaviour and therefore by latching the value into a negative sign, the system will just return to its original position.

The retreat is shown in Figures 5.22 to 5.29 in all coordinate systems. Notice after the collision point, that the movement is slower and hence the change in direction of the DMP generator signal is less steep. This comes from the lack of symmetry in the returning path and the negative change of the decay rate. Furthermore, the observed primitive shows that the path is not entirely the same as before. This is because of the modified transformation dynamics, which causes the shape to change.

After the robot has done some retreat, the system is then stopped by detecting a small time lapse after the collision and halting all computation of new points from the DMP generator.

Another element that must be noticed is that the generated path when in contact is not diminished immediately and therefore the offset force is only mitigated by the working impedance controller. The contact with an object lasted around 0.9 seconds. This proposed method might not be the best with fast moving robots, since the differential trajectory can be high during a short time and the current parameters of the impedance controller might not give a good enough compliance with the object. Tables 5.9 and 5.10 gives a look of the maximum error detected between the DMP generator command and the actual position of the robot.

The error between DMP and robot system is at its highest when α_s reaches 0 and the canonical system starts to increase in value. This moment can be seen in Figure **??** at the maximum drop near the 0.3 value. Nevertheless, if the parameter of stiffness in the controller is lowered, even though the robot precision drops and a compliance can become much better.

The drop of the parameter must be done quickly, otherwise the time of contact with the collided object will be too long. This is accomplished by creating a fast negative drop in the canonical decay rate.

Changing rapidly the decay rate comes with the cost of losing precision in the differential equations. The decaying rate is directly linked to the step size that the exponential system can perform. By creating big steps, the systems acceleration will increase as well, and the

actual position of the differential equations will start to diverge from the computed ones. This error affects the robot's position and can cause it to reach undesirable poses and is one of the biggest downside when trying to recover from a collision with the proposed modifications.

Coordinate	Error (mm)	Coordinate	Error (unit-less)
X	15.3611	q_w	0.0139
Y	2.3297	q_x	0.0107
Z	16.7684	q_y	0.0044
		q_z	0.0150

Table 5.9: XYZs and Quaternions Maximum Errors detected during collision

Joint #	Error (degrees)
A1	0.8568
A2	6.2442
A3	0.8112
A4	6.1200
A5	0.8205
A6	1.7434
A7	1.3739

Table 5.10: Joint Maximum Errors detected during collision

Due to the transformation system's properties, the retreating path can go through a singularity which can be detrimental. This is due to the pattern that has been learned and will be reproduced in the system. Nevertheless, in this test after detecting a collision, the robot successfully retreated into another position and stopped. Derived not having high velocities and the proposed canonical system's response to the collision.

5.8 Multiple DMP task - Unscrewing a burndy connector

Initially, it was mentioned about the Collimator's disconnection task that is performed at the CERN's accelerators. A disconnecting device is set on top of the beam pipe to remove it. To accomplish this task, several repetitive tasks must be followed and each one of them is described in the pattern explained next.

1. Robot needs to reach close near the screw's or object's area.
2. The end-effector needs to slowly move towards the screw in order to fit in the tool.
3. Use the tool (e.g. grab).
4. Move away the end-effector from the area.

To validate the presented framework, this task will be done in autonomous mode and different primitives will be learned to do steps 1,2 and 4. Step 3 will be done separately by

Position	X (mm)	Y (mm)	Z (mm)
Start	268.7	180.5	809.3
Original Goal	682.14	-268.03	293.06
New Goal	519.89	576.94	281.24

Table 5.11: Position values for the Reach-Object-Vicinity primitive

another device which is going to be the gripping of an object. The object will be a circular burndy connector (Figure 5.36). Additionally, an additional action will be included in step 4 which is turning the tool. This is done to unscrew the burndy connector.

Initially three primitives will be learned by the system. They are going to be stored in the library and will be chosen to perform this task. For the first part of the task, that is to reach the objects area, the primitive will be defined as "reach-object-vicinity". To test the dynamic adaptability of the framework, this particular primitive, will be taught with a different goal location. The action will be to reach the area, but not specific to the task's object.

Figure 5.38 illustrates the change of the robot's TCP path using the old primitive with a goal position set to the new one. The Table 5.11 indicates the original values of the goal position and the new one. For the orientation, no values were changed since both had the desired orientation at the path's end (TCP aimed down).

A second primitive will be name as "go-down-extended-pose". The robot will normally be extended in this primitive, but not near the singularity zone. Then the end-effector will move slowly towards the burndy connector and be in position to grasp it with the tool.Finally, the third primitive will be the "turn-and-pull" action. In this part, the end-effector will slowly turn to the desired direction to unscrew the connector, and pull once the rotation is finished.

The path for all the primitives performed is illustrated by the response of the robot system in Figure 5.41. Each color represents a dynamic primitive. The time duration for each one of them is presented in the Table 5.12.

The primitive for go-down-extended-pose, presents a movement not entirely straight as it can be seen in Figure 5.41 by the cyan color. This was due to how the primitive was learned, which was near the position and moving the robot arm down. The movement does not go down in a straight line but it does reach the destination. These kind of

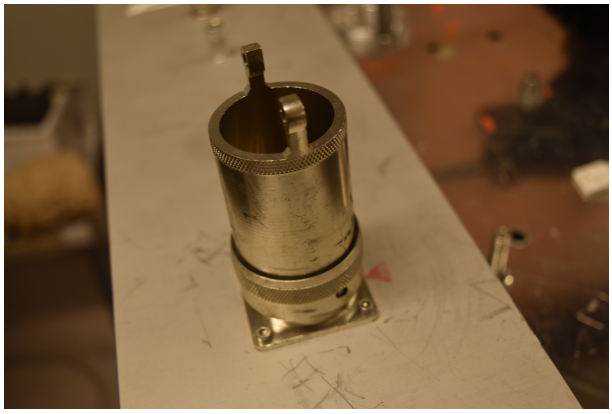

Figure 5.36: Burndy connector for DMP task execution test

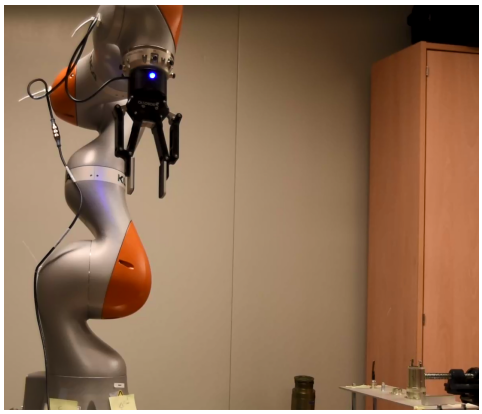

Figure 5.37: Kuka robot in starting position

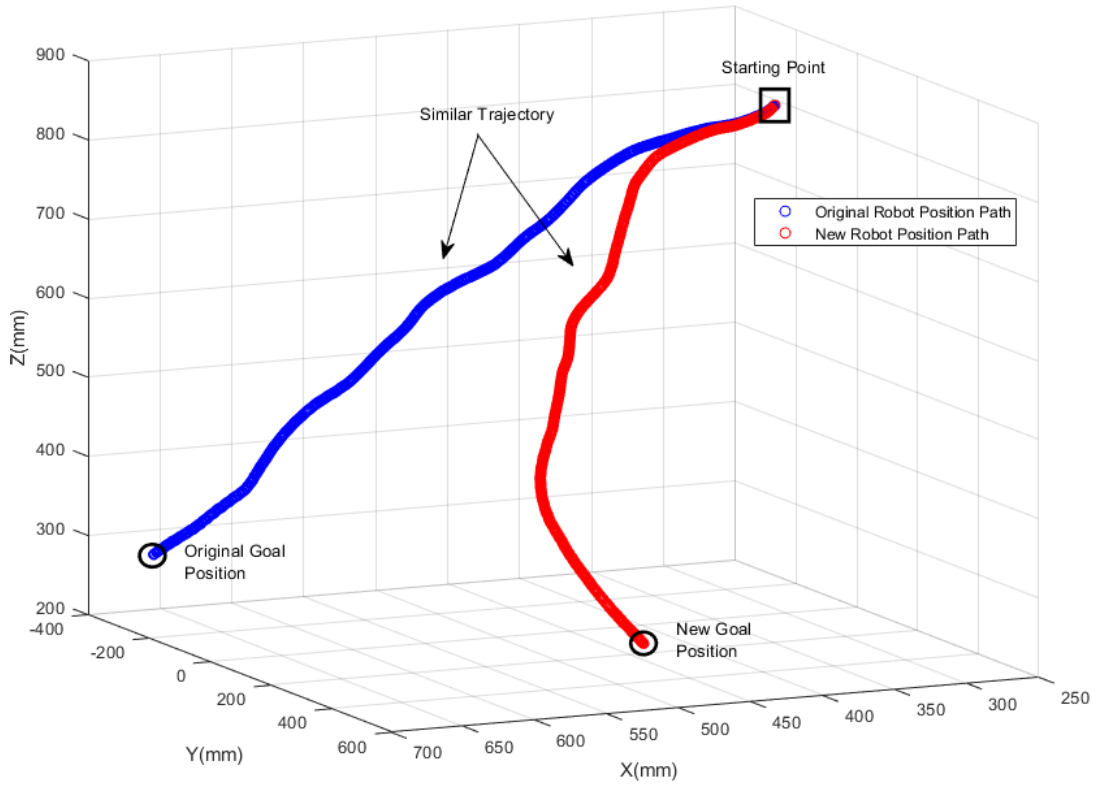

Figure 5.38: Robot response to goal change in Reach-Object-Vicinity primitive

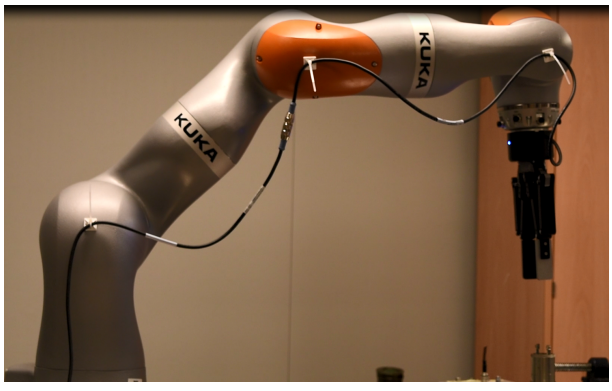

Figure 5.39: Kuka robot finishing Reach-Object-Vicinity Primitive

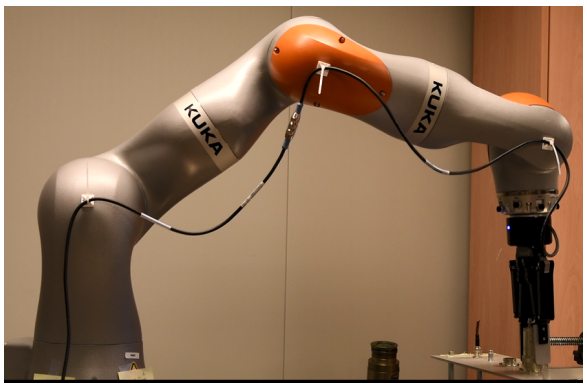

Figure 5.40: Kuka robot finishing Go-Down-Extended Primitive

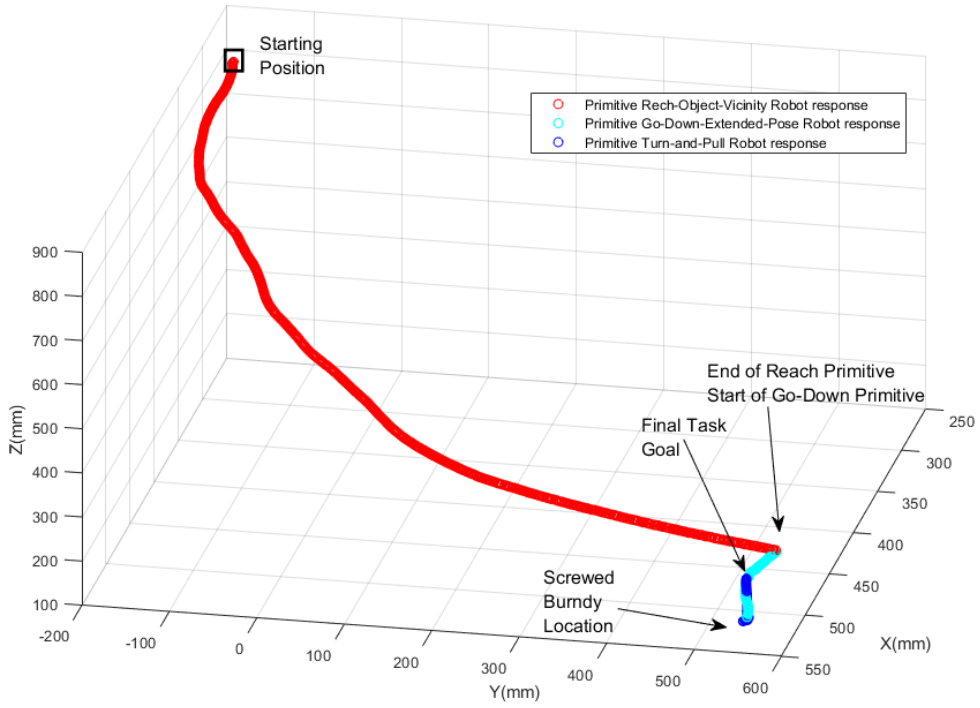

Figure 5.41: Full path of TCP performed by KUKA robot with 3 primitives for task completion

Primitive	Duration (s)
Reach-Object-Vicinity	12.40
Go-Down-Extended-Pose	8.31
Turn-and-Pull	21.27

Table 5.12: Duration for each Reach-Object-Vicinity primitive

behaviour is expected when using kinesthetic learning to accomplish very precise task that can be difficult for an operator to perform on a robot. Nevertheless, the gripper reached the burndy connector properly.

Once the gripper grabs a burndy connector, the third primitive comes into play. Turning and pulling can be a dangerous primitive since it does not guarantee that the turn completely unscrew the connector. Nevertheless, in this test the learned trajectory is done successfully. The pull was done as well and the robot managed to pull the burndy connector out of its socket.

The completed task's duration took around 45 seconds including the time to grasp the object. The longest was on the turn-and-pull action, due to the level of carefulness that the robot needs to have. This is reflected in the learned primitive since the operator teaches this task slowly to have more precision in its moves.

This test shows the usefulness of an autonomous mode altogether with a library of primitives. Its versatility is due to the capacity of adapting to new situations from already learned information. Nevertheless, parts of the task took several tries as the robot configuration managed to pass through singularities and did not perform well in these cases. Particularly the turn and pull presented problems since the action pulling

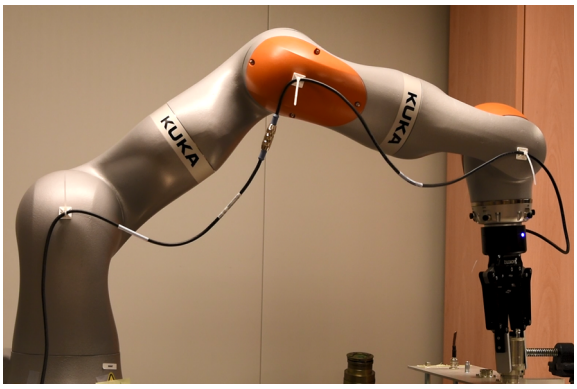

Figure 5.42: Kuka robot during turning action of primitive

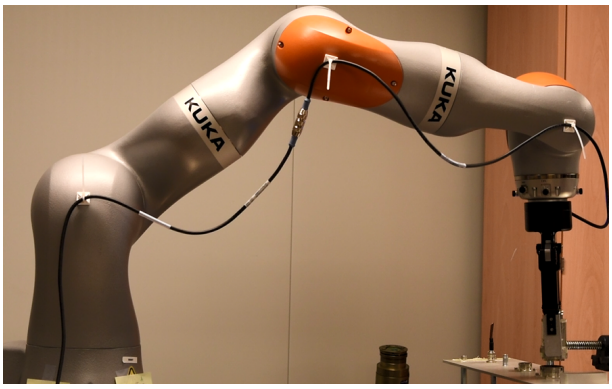

Figure 5.43: Kuka robot pulling action of primitive

in an very extended position caused the robot to reach a singularity easily. This kind of problem can be solved using only joint positioning for the robot's control, but the task space manipulation is lost in the way and therefore also part of the frameworks desired functionality.

Chapter 6

Conclusions & Future Work

In this work, the dynamic movement primitives framework was implemented in a real high redundancy robotic system. The compliant behaviour was achieved through the use of an impedance controller with an enhanced canonical system which gives extra functionality to an already existing primitives framework. The following are several points taken from these studies which are conclusions coming from this work.

- Teaching through kinesthetic learning is very useful to create complex motions without the need to simulate kinematics. The hand-guidance mode uses a low stiffness impedance controller which makes the robot arm easy to move.

- The used learned primitives are highly dependable on the demonstration because of the Locally Weighted Regression algorithm, and teaching through human-robot interaction. If the demonstration is badly performed, the system will response badly as well. Nevertheless, these methods of learning are very fast and can be constantly repeated.

- Automated tasks, such as unscrewing, can be done through dynamic movement primitives due to their simple and flexible nature. The framework is adaptable to changes in parameters which makes them versatile in different kinds of situations, such as goal variation, without becoming unstable. Furthermore, many different primitives can be used for accomplishing a complex task.

- The compliant behaviour implemented can stop robot motion by using both the enhanced canonical system and an impedance controller. It can also retreat to a position out of collision. However, the path is not always the most optimal and it can even put the robot in an undesired pose or location.

- Singularities are extremely dangerous in a high-dimensionality robots when performing demonstrations that come from kinesthetic teaching. The user can unknowingly pass through a bad position, or singularities, which result in primitives not usable in task space.

- Controlling a robot in world coordinates through the autonomous mode can be both beneficial and detrimental. Its representation in world space is highly desirable which makes the adaptation of primitives simpler and easier to use in other works. However, the inverse kinematics dependency to control the robot system creates high accelerations near singularities.

- A compliant behaviour is achieved with the use of both joint impedance control and the proposed enhanced canonical system. The decay's rate dynamics allows for the generation points to slow down, and it can also be used to create a retreating path when a collision is detected. Nevertheless, the path is not guaranteed to be the most optimal and can cause undesirable robot poses.

- The system is able to learn and smooth complex trajectories through the use of locally weighted regression for non-linear estimations and ordinary differential equation. The mixture of canonical and transformation systems allows robot learning with high degrees of freedom.

- Dynamic Movement Primitives are adaptable to various applications. Such as the proposed experimental robot-guidance mode, where the operator can use a primitive and be guided by the robot.

6.1 Future Work

Much more work can be added in this implemented framework which can be expanded and improved further. Its adaptability to new elements is one of the fortes in this system and the addition of different functionalities can extend its working range and applications.

The use of primitive libraries is very useful in the successful completion of tasks. But in this work, the user needs to select the primitives separately, which takes extra work especially in an automated task. Mülling et al. (2013) uses a mixture of motor primitives (MoMP) to select and generalize among primitives, where they are weighted by their suitability in a given task or context. Its work is used to teach table tennis to a robot arm, where it automatically selects the best primitive depending on sensor inputs.

Using a fixed value for the hyper-parameters, such as number of kernel functions and bandwidth, can give a good representation of the demonstrated data. Nevertheless, if these are not selected correctly then approximation errors can appear. A promising algorithm to solve this issue is the Receptive Field Weighted Regression (RFWR) Schaal and Atkeson (1998). Depending on the trajectory's shape, the algorithm can select learning hyper-parameters automatically. It can also be used to handle redundant data to fuse new demonstrations into an already existing one and create a single primitive from several different demonstrations. Schaal et al. (2002) is another source where various non-parametric techniques are used in real-time robot learning and might be useful to apply.

An integration with a vision system to handle position parameter changes by means of a perceptual coupling is a highly desired feature in this work. Works such as Pastor et al. (2009) and Samant et al. (2016) use vision algorithms to communicate with the dynamic movement primitives and accomplish different task. Variations of primitive parameters through sensors brings high adaptability to new situations and allows to test real-time modulation.

Robot task space control is accomplished in this work through kinesthetic learning and inverse kinematics computation to an impedance controller. However, singularities are always problem when using high redundancy robots which can make certain primitives not work in specific situations. Due to the adaptable nature of the primitives and freedom of teaching through the hand-guidance mode, it is possible to cross singularities in the robot's path very easily. A solution of this problem could be to include an obstacle

avoidance motion generation system such as the ones presented by works of Stavridis et al. (2017) and Hoffmann et al. (2009) to avoid singularities. It will allows the robot to perform paths while dodging specific spots which are considered beforehand through some method of simulation or computation as a singularity spot, and therefore improve the robotic system path planning. Promising as it is, the implication of implementing this method still needs to be studied further.

Furthermore, the experimental compliant mode can be extended to tele-operated tasks. The general purpose of this mode is to use learned dynamic movement primitives to perform a specific task while the user is moving the robot arm. It will assist the operator in performing the task, but will not function automatically. Coupled with vision and force feedback the system can be an addition to reduce an operators stress in high-risk tele-operated activities. The implementation of such work will require a bi-lateral control scheme between dissimilar robot structures and the coupling of different systems using the same primitives. Work like the one presented by Umlauft et al. (2014) make use DMPs for cooperative manipulation between two arms, and can be useful for tele-operated activities if further researched and tested.

The current enhanced canonical system can be used to detect collisions through errors in the primitives and actual status of the robot. Even though it generates a path to retreat collisions, the modification should also be extended to the transformation system. Due to time restrictions, this implementation was developed to just move backwards, and it brings errors from its original path. This means that is possible to reach undesirable poses or locations. Changing the behaviour of a primitives' transformation system when detecting a collision would be an interesting approach for further development in dynamically retracing a path.

Bibliography

B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009. ISSN 09218890. doi: 10.1016/j.robot.2008.10.024. URL http://arxiv.org/abs/1710.08789.

F. Burkart. *Beam Loss and Beam Shape at the LHC Collimators*. Masterarbeit, GOETHE UNIVERSITÄT FRANKFURT, 2012. URL http://cds.cern.ch/record/1447801/files/CERN-THESIS-2012-046.pdf.

S. Calinon. A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, 9(1):1–29, 2016. ISSN 18612784. doi: 10.1007/s11370-015-0187-9. URL https://www.researchgate.net/profile/Sylvain{_}Calinon/publication/282414847{_}A{_}Tutorial{_}on{_}Task-Parameterized{_}Movement{_}Learning{_}and{_}Ret
links/561cba0808aea80367252117.pdf.

Q. Chen, H. Cheng, C. Yue, R. Huang, and H. Guo. Step length adaptation for walking assistance. *2017 IEEE International Conference on Mechatronics and Automation, ICMA 2017*, pages 644–650, 2017. doi: 10.1109/ICMA.2017.8015892.

S. Chernova and A. L. Thomaz. Special Issue on Robot Learning from Demonstration. *International Journal of Social Robotics*, 4(4):317–318, 2012. ISSN 18754791. doi: 10.1007/s12369-012-0165-8.

A. Gams, B. Nemec, A. J. Ijspeert, and A. Ude. Coupling movement primitives: Interaction with the environment and bimanual tasks. *IEEE Transactions on Robotics*, 30(4):816–830, 2014. ISSN 15523098. doi: 10.1109/TRO.2014.2304775.

K. Glatz. *Adaptive Learning from Demonstration using Dynamic Movement Primitives*. PhD thesis, Hochschule Ravensburg-Weingarten University of Applied Sciences, 2012. URL http://link.springer.com/10.1007/BF03192151.

R. I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, 1997. ISSN 01628828. doi: 10.1109/34.601246.

H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal. Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance. *2009 IEEE International Conference on Robotics and Automation*, pages 2587–2592, 2009. ISSN 1050-4729. doi: 10.1109/ROBOT.2009.5152423.

A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation Learning. *ACM Computing Surveys*, 50(2):1–35, 2017. ISSN 03600300. doi: 10.1145/3054912. URL http://dl.acm.org/citation.cfm?doid=3071073.3054912.

A. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, 2:1398–1403, 2002. ISSN <null>. doi: 10.1109/ROBOT.2002.1014739. URL http://ieeexplore.ieee.org/document/1014739/.

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, 2013. ISSN 0899-7667. doi: 10.1162/NECO_a_00393. URL http://www.mitpressjournals.org/doi/10.1162/NECO{_}a{_}00393.

M. Karlsson. *On Motion Control and Machine Learning for Robotic Assembly*. PhD thesis, Lund University, 2017. URL http://portal.research.lu.se/ws/files/27898719/lic{_}mk.pdf.

J. Kober and J. Peters. Imitation and reinforcement learning. *IEEE Robotics and Automation Magazine*, 17(2):55–62, 2010. ISSN 10709932. doi: 10.1109/MRA.2010.936952. URL http://eprints.pascal-network.org/archive/00008047/.

KUKA Roboter GmbH. KUKA LBR iiwa Specifications. pages 1–69, 2016a. URL https://www.kuka.com/-/media/kuka-downloads/imported/48ec812b1b2947898ac2598aff70abc0/spez{_}lbr{_}iiwa{_}de.pdf.

KUKA Roboter GmbH. KUKA Sunrise OS 1.11 KUKA Sunrise Workbench 1.11. pages 1–557, 2016b. URL http://www.oir.caltech.edu/twiki{_}oir/pub/Palomar/ZTF/KUKARoboticArmMaterial/KUKA{_}SunriseOS{_}111{_}SI{_}en.pdf.

C. Lauretti, F. Cordella, E. Guglielmelli, and L. Zollo. Learning by Demonstration for Planning Activities of Daily Living in Rehabilitation and Assistive Robotics. *IEEE Robotics and Automation Letters*, 2(3):1375–1382, 2017. ISSN 2377-3766. doi: 10.1109/LRA.2017.2669369. URL http://ieeexplore.ieee.org/document/7856889/.

R. J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. SIAM, Society for Industrial and Applied Mathematics, 1 edition, 2007. ISBN 978-0-89871-629-0. doi: 10.1137/1.9780898717839. URL http://epubs.siam.org/doi/book/10.1137/1.9780898717839.

F. Meier, P. Hennig, and S. Schaal. Efficient Bayesian local model learning for control. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, number Iros, pages 2244–2249. IEEE, sep 2014. ISBN 978-1-4799-6934-0. doi: 10.1109/IROS.2014.6942865. URL http://ieeexplore.ieee.org/document/6942865/.

K. Mülling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, mar 2013. ISSN 0278-3649. doi: 10.1177/0278364912472380. URL http://journals.sagepub.com/doi/10.1177/0278364912472380.

M. W. Otte. A Survey of Machine Learning Approaches to Robotic Path-Planning. *Cs.Colorado.Edu*, 2015. URL http://www.cs.colorado.edu/{~}mozer/Teaching/ComputationalModelingPrelim/Otte.pdf.

P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. *2009 IEEE International Conference on Robotics and Automation*, pages 763–768, 2009. ISSN 1050-4729. doi: 10.1109/ROBOT.2009.5152385. URL http://ieeexplore.ieee.org/document/5152385/.

R. Samant, L. Behera, and G. Pandey. Adaptive learning of dynamic movement primitives through demonstration. *Proceedings of the International Joint Conference on Neural Networks*, 2016-Octob:1068–1075, 2016. ISSN 2161-4393. doi: 10.1109/IJCNN.2016.7727316.

S. Schaal. Dynamic Movement Primitives -A Framework for Motor Control in Humans and Humanoid Robotics. *Adaptive Motion of Animals and Machines*, pages 261–280, 2003. doi: 10.1007/4-431-31381-8_23. URL http://link.springer.com/10.1007/4-431-31381-8{_}23.

S. Schaal and C. G. Atkeson. Constructive Incremental Learning from Only Local Information. *Neural Computation*, 10(8):2047–2084, 1998. ISSN 0899-7667. doi: 10.1162/089976698300016963. URL http://eutils.ncbi.nlm.nih.gov/entrez/eutils/elink.fcgi?dbfrom=pubmed{&}id=9804671{&}retmode=ref{&}cmd=prlinks{%}5Cnpapers3://publication/uuid/2494BCB3-514B-4133-8EB1-2E8A8F2EE129.

S. Schaal, C. G. Atkeson, and S. Vijayakumar. Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, 17(1):49–60, 2002. ISSN 0924669X. doi: 10.1023/A:1015727715131.

S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Control, Planning, Learning, and Imitation with Dynamic Movement Primitives. *Workshop on Bilateral Paradigms on Humans and Humanoids, 2003 IEEE International Conference on Intelligent Robots and Systems IROS*, pages 1–21, 2003.

S. Stavridis, D. Papageorgiou, and Z. Doulgeri. Dynamical System Based Robotic Motion Generation With Obstacle Avoidance. *IEEE Robotics and Automation Letters*, 2(2):712–718, apr 2017. ISSN 2377-3766. doi: 10.1109/LRA.2017.2651172. URL http://ieeexplore.ieee.org/document/7812711/.

J. Umlauft, D. Sieber, and S. Hirche. Dynamic Movement Primitives for cooperative manipulation and synchronized motions. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 766–771, 2014. ISSN 10504729. doi: 10.1109/ICRA.2014.6906941.

Wikipedia contributors. Gradient descent — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Gradient_descent&oldid=847897318, 2018. [Online; accessed 8-January-2018].

List of Figures

List of Tables