Skill Transfer from Multiple Human Demonstrators to a Robot Manipulator Using Neural Dynamic Motion Primitives

by

Geoffrey Hanks

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2024

© Geoffrey Hanks 2024

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Programming by demonstration, also known as imitation learning, has shown potential in reducing the technical barriers to teaching complex skills to robot manipulators. It involves obtaining one or more demonstrations of how to complete a task, often from a human, which are then transferred to a robotic system. Dynamic Motion Primitives (DMPs) are an efficient method of learning trajectories from individual demonstrations using second-order dynamic equations. Research has been done to overcome some of the limitations of DMPs, by generalizing over multiple demonstrations, sequencing multiple primitives to complete goals involving multiple sub-tasks, and adding via-points for increased control over complex motions. However, accomplishing more complex tasks using DMP sequencing and via-points requires task specific knowledge so that the demonstrations can be segmented or annotated, and the breakdown of some tasks may be unintuitive. This can further increase the time and effort required to collect demonstrations beyond the already demanding process of collecting physical demonstrations, decreasing the feasibility of learning from demonstration in certain situations.

This thesis applies state of the art Cartesian space DMPs that utilize physically collected and augmentation data to create a framework that can reduce the task specific knowledge and human effort required to teach robots multi-step tasks. DMPs that integrate neural networks are used, not only to generalize over multiple demonstrations from different demonstrators, but also to learn from complete demonstrations without requiring segmentation or annotation. For comparison, sequenced DMPs which require their demonstrations to be segmented into sub-tasks prior to learning are also implemented. Both techniques utilize physically collected demonstrations which are augmented to reduce the time and effort required to collect demonstrations, while ensuring sufficient samples for proper learning.

The framework was tested on a pouring task which could be split into sub-tasks, and was tested both in simulation and on a 7 degree of freedom Franka Emika Panda robot manipulator. The task involved reaching for and grasping a container with water, pouring water into another container placed in the workspace, and returning the pouring container to its original location. Both sets of models were tested on their ability to recall trajectories shown in training, and generalize to new inputs. They were then implemented on the physical robotic system, and both methods were successful in completing the task. The trade-offs between the models trained on full and segmented demonstrations are discussed. While the sequenced DMPs were found to have reduced average error and greater flexibility, they required extra work and task knowledge to generate the demonstrations, and were reliant on specific subtasks being defined.

It was determined that the models trained from full demonstrations using this framework could be an alternative to sequence primitives for more complex tasks. Despite a higher error between the demonstrations and predicted trajectories when compared to a sequence of DMPs, the full models are able to recall trajectories, generalize to new inputs well enough to complete the task on a physical robot. As such, they have the potential to reduce effort and task knowledge during demonstration preparation, and expand the applicability of imitation learning to a wider range of tasks.

Acknowledgements

I would like to thank my supervisor, Prof. Yue Hu for her support throughout my Master's degree. Her insights and guidance have not only been extremely valuable in completing this project, but have also helped me develop my research and technical abilities. I am very appreciative of the time and effort she has invested in helping to review and guide my work, and I look forward to continuing to work with her during my PhD.

I would like to express my gratitude to my colleagues and friends at the university of Waterloo and the Active and Interactive Robotics (AIR) lab. Their continued support through discussions, assistance with data collection, and feedback on presentations and ideas has been invaluable in completing my degree.

I am also appreciative of the support and guidance of Prof. Gentiane Venture and the members of the University of Tokyo GVLab during this project.

Finally, I would like to thank my family and my partner for their ongoing belief and support as I continue to pursue a career in research and robotics, without which this would have been immensely more difficult.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgements v

List of Figures ix

List of Tables xii

1 Introduction 1

1.1 Objectives 3

1.2 Contributions 4

1.3 Thesis Structure 5

2 Literature Review 7

2.1 Learning From Demonstrations in Robotics 7

2.2 Trajectory Representation and Generation Using Dynamic Motion Primitives 8

2.3 Learning Generalized Tasks using Dynamic Motion Primitives 10

2.4 Methods for Demonstration Collection 11

2.5 Techniques Used and Knowledge Gaps 13

3 Formulation and Training of Dynamic Motion Primitives and Robot Control **15**

3.1 Formulation of Dynamic Motion Primitives for Position Representation . . 15

3.2 Quaternion Formulation of Dynamic Motion Primitives for Orientation Representation . . . 17

3.3 Neural Dynamic Motion Primitives . . . 18

3.3.1 Position Model . . . 18

3.3.2 Orientation Model . . . 19

3.4 Model Training . . . 19

3.4.1 Using Cross Validation for Reduced Overfitting . . . 19

3.4.2 Hyperparameter Tuning . . . 20

3.5 Chaining Consecutive DMPs . . . 21

3.6 Robot Control . . . 22

4 Demonstration Collection and Dataset Creation **24**

4.1 Demonstration Collection . . . 24

4.2 Demonstration Segmentation for Multi-Stage Dynamic Motion Primitives . 27

4.3 Dataset Formulation and Augmentation . . . 28

4.3.1 Dataset Formulation . . . 29

4.3.2 Data Augmentation . . . 29

4.3.3 Training and Validation Sets for Cross Fold Validation . . . 31

5 Analysing Neural DMP Performance **32**

5.1 Model Losses . . . 32

5.2 Model Trained on Full Demonstrations . . . 34

5.2.1 Training Set Performance . . . 35

5.2.2 Testing Set Performance . . . 35

5.3 Model Trained on Segmented Demonstrations . . . 39

5.3.1 Training Set Performance . . . 39

5.3.2 Testing Set Performance . . . 41

5.4 Discussion on the Model Training and Validation Results . . . 46

6 Testing on Robot Hardware **48**

6.1 Testing Model Learned from Full Demonstrations 48

6.2 Testing Model Learned from Segmented Demonstrations 50

6.3 Discussion on Testing the Framework on a Robotic System 50

7 Conclusions and Future Work **55**

7.1 Conclusions 55

7.2 Future Work 57

References **59**

APPENDICES **66**

A Research Ethics Approval Letter **67**

B Cross Validation Losses **70**

C Image Sequences for C7 L2 **72**

List of Figures

1.1 Demonstrator performing a pouring task (left) and robot performing the learned pouring task (right). 5

3.1 Control structure overview, showing the transformation of the input features through the position and orientation neural networks to predict the weight and goal parameters of their respective DMPs, which then generate trajectories. Inverse kinematics generate a joint position trajectory from the Cartesian trajectory, which the robot follows using a joint impedance controller. 23

4.1 The front (left) and back (right) of how the Xsens Awinda IMU motion capture sensors are worn by the demonstrator. 25

4.2 Configurations with pouring container locations as rectangles and the goal container as a circle. 26

5.1 Trajectory components of the demonstrated position and the position predicted by the neural DMP for the model trained using the full demonstrations for samples from the training dataset. 36

5.2 Trajectory components of the demonstrated orientation and the orientation predicted by the neural DMP for the model trained using the full demonstrations for samples from the training dataset. 37

5.3 Trajectory components of the demonstrated position and the position predicted by the neural DMP for the model trained using the full demonstrations for samples from the test data set.Three simple graphs 38

5.4 Trajectory components of the demonstrated orientation and the orientation predicted by the neural DMP for the model trained using the full demonstrations for samples from the test data set. 38

5.5 Trajectory components of the demonstrated position and the position predicted by the neural DMP for the model trained using the segmented demonstrations for samples from the training dataset. 42

5.6 Trajectory components of the demonstrated orientation and the orientation predicted by the neural DMP for the model trained using the segmented demonstrations for samples from the training dataset. 43

5.7 Trajectory components of the demonstrated position and the position predicted by the neural DMP for the model trained using the segmented demonstrations for samples from the test dataset. 44

5.8 Trajectory components of the demonstrated orientation and the orientation predicted by the neural DMP for the model trained using the segmented demonstrations for samples from the test dataset. 45

6.1 Desired and actual position of the robot end effector for the models trained using full demonstrations. 49

6.2 Desired and actual orientation of the robot end effector for the models trained using full demonstrations. 49

6.3 Image sequence showing the first trajectory and water level (C1 L1) being performed by the robot using the neural DMPs trained on the full demonstrations with no segmentation. 51

6.4 Desired and actual position of the robot end effector for the models trained using segmented demonstrations. 52

6.5 Desired and actual orientation of the robot end effector for the models trained using segmented demonstrations. 52

6.6 Image sequence showing the first trajectory and water level (C1, L1) being performed by the robot using the neural DMPs trained on the segmented demonstrations. The first row corresponds to the first segment (grasping), the second row corresponds to the second segment (pouring) and the third row corresponds to the third segment (returning). 53

C.1 Image sequence showing a trajectory and water level (C7, L2) being performed by the robot using the neural DMPs trained on the full demonstrations. 73

C.2 Image sequence showing a trajectory and water level (C7, L2) being performed by the robot using the neural DMPs trained on the segmented demonstrations. The first row corresponds to the first segment (grasping), the second row corresponds to the second segment (pouring) and the third row corresponds to the third segment (returning). 74

List of Tables

4.1 Number of samples in each dataset before and after augmentation. 30

5.1 Training, validation and testing losses for each of the position models and the average loss across all segments weighted by segment lengths. Cross Validation losses are averaged across each fold shown in tab B.1 in Appendix B. 33

5.2 Training, validation and testing losses for each of the orientation models the average loss across all segments weighted by segment lengths. Cross Validation losses are averaged across each fold shown in tab B.2 in Appendix B. 33

5.3 Model parameters for the position and orientation neural DMPs trained on the full demonstrations 34

5.4 Model parameters for the position neural DMPs trained on the segmented demonstrations for each segment. 40

5.5 Model parameters for the position neural DMPs trained on the segmented demonstrations for each segment. 40

B.1 Average loss over each trajectory and dataset for each fold during training and validation for the position models. 70

B.2 Average loss over each trajectory and dataset for each fold during training and validation for the orientation models. 71

Chapter 1

Introduction

Robotic manipulators have the potential for many positive benefits in our society. They have been applied in industrial setting to improve productivity by automating repetitive tasks, improve efficiency, increase safety and add reliability [3], [17]. They have also been implemented in medical applications such are remote surgery [66], mental health diagnosis and treatment [41] and for rehabilitation [60] showing their applicability in diverse and complex environments. While autonomous robotic systems have been successfully applied in many situations, further development is required to increase their accessibility.

The implementation of traditional robotic systems for manipulators is often technically demanding and can require a large investment of time and resources. Systems are typically designed for a specific purpose, with task knowledge hard coded into the architecture [11]. In order to obtain the task knowledge, designers must first consult with experts to develop an understanding of how a task is structured, and how it can be completed. They must then have the technical knowledge and expertise in robotics and programming to implement algorithms and controllers to successfully transfer this skill to the robotic system [65].

This process can be difficult for complex tasks as it requires the task expert and robot designer to gain an understanding of skills and communicate across domains which may be outside of their area of expertise. For the designer to develop a task specific system, architecture, and models, they must understand the nuances and intuition of the tasks which an expert may have built over years of experience. Similarly, the limitation of the robotics system may be difficult for the designer to convey to the task expert. While this may be possible for simple tasks, it can be inefficient, and sometimes not feasible, for tasks that are complex, or that require large investments of time and effort to learn. For example, surgical tasks or complex maintenance tasks may vary significantly from case to

case, and may require intensive training to gain proficiency.

Learning from demonstration, also known as behavioural cloning or imitation learning, is a supervised machine learning technique designed to be a more intuitive and flexible method of robot programming [11]. Instead of a roboticist designing a specific control architecture based on input from a task expert, they design a system that the expert can interact with directly. Demonstrations performed by one or more experts are collected by the system, and are then used to generate a model of how the task should be completed. This model can then be used to complete the same task, and potentially generalize to variations in the task or environment. Not only is this more intuitive for the demonstrator, but it allows for a more flexible framework that can be applied to multiple tasks.

One benefit of learning from demonstration is the expert performing the demonstrations is less exposed to the technical implementation of the system. This is done by allowing task experts to interact directly with the robotic system to impart their skills instead of requiring their knowledge pass through the roboticist to be implemented. Similarly, the efforts of the robotics designer can be focused on the technical implementation of the robotic system. As such, the variety of tasks than can be learned, allowing for the automation of tasks beyond simple and repetitive tasks.

Another benefit of learning from demonstrations is the ability to model complex tasks and skills. Instead of manually defining mathematical models and trajectories to complete a task, learning from demonstration is a data driven approach to learn from expert demonstrators. Advances in machine learning techniques and computing abilities have contributed to the increase in popularity of learning from demonstration, as it is now possible to learn more complex and generalizable models with techniques such as neural networks [20]. This has somewhat increased to variety and complexity of tasks which can be transferred through learning from demonstration, but applications gnerally remain limited to simpler tasks.

While learning from demonstration can have many benefits, it still faces some challenges. Collecting a sufficient number of high quality demonstrations can be a challenge to overcome in learning from demonstration in robotics. The lack of available demonstrations could also be enhanced for tasks that occur infrequently, or may occur in dangerous locations, such as deep sea welding, or maintenance tasks in outer space. The number of training samples varies between different learning methods, meaning methods that are effective with few samples can reduce the time and effort required during data collection. Synthetically generated data has also been used to train models for certain tasks [5], although this is limited by the ability to simulate the completion of the objective and can often negate the modeling benefits obtained from learning by demonstration. Data aug-

mentation can also be used to circumvent this challenge, by increasing both the number and variety of samples in the datasets [32].

The method of collecting the demonstrations can influence the amount of post processing required, and may have an impact on how the expert performs the task. Some methods involve directly controlling the robot, which ensures that the generated trajectories are feasible in terms of the systems kinematic limitations. However, this requires the expert to learn a new control scheme for the robot [11]. Other methods allow demonstrations to be collected by observing the demonstrator on their own without the robot. While this limits the impact and interference on the demonstrators actions, it can require remapping to the robot's constraints [40].

Dynamic motion primitives (DMPs) are a method of representing trajectories using second order dynamic systems that have become popular in robotics [46]. While the details of this method are explored in depth in later sections, it has several characteristics which make it attractive for learning from demonstrations in robotics. They are a flexible framework that can by formulated in different task spaces, such as joint space or Cartesian space, making them suitable for a wide variety of purposes. They have guaranteed stability, and are able to scale in both spatially and temporally. They are also able to learn from a single demonstration, which can reduce the amount of time an resources required to collect training data. However, it is often desirable learn from a variety of demonstrations to increase generalizability, which is often done by using machine learning techniques such as neural networks, creating what is known as neural DMPs [42]. This and many other variations on DMPs have been explored to expand their capabilities [43].

1.1 Objectives

In this thesis, the objective is to reduce the technical barriers to programming robotic manipulators with learning from demonstration, and expand the feasibility of programming by demonstration to more complex tasks. In particular, it aims to reduce the amount of human effort and specific task knowledge required to generate demonstration datasets and train models for tasks that are composed of multiple stages. In doing so, this could provide a learning from demonstration option for tasks that require highly trained experts, or where demonstrations are scarce.

The presented framework aims to provide an alternative to current popular models by learning from full demonstrations without any segmentation or pre-processing. Splitting the demonstrations into sub-tasks before learning can provide more flexibility, but requires

task knowledge and can be unintuitive for some tasks. Expanding the complexity of movements that can be learned by motion primitives could reduce the complexity in generating demonstrations, and could make learning by demonstration applicable for a wider class of tasks. It could also reduce the cost of processing demonstrations, particularly when experts are from fields requiring extensive training and expertise.

This thesis also aims to augment and supplement physically collected demonstrations, which can be time and resource intensive to collect. While some tasks can rely on synthetically generated datasets, this is not feasible for many physical tasks as an accurate model can not be generated, so humans must be involved in the demonstration process. Therefore, ensuring that sufficient samples can be obtained to train data driven models while reducing the number of human generated demonstrations that need to be collected is an important advancement required to allow learning from demonstration to be applied in more complex and demanding situations.

1.2 Contributions

A framework for learning tasks in Cartesian space from demonstrations was developed, and two sets of models are trained. While the selected task could be broken into sub-tasks, the first set of models learned from the full demonstrations without segmentation, while the other learned from demonstrations that had been split into smaller segments to make up sub-tasks. The same training method and inputs were used for both sets of models so that their performance could be compared. The contributions presented in this thesis are as follows:

- Implementation of a learning from demonstration framework that can learn to generalize over a task from multiple demonstrations.
- Training of motion primitives to complete a multi-step process using neural networks without dividing the demonstration into sub-tasks.
- Comparison of the performance of a system that learned from the full demonstrations of a multi-step process to a system that was trained using demonstrations that were segmented into predefined sub-tasks.
- Utilization of data augmentation to reduce the time and effort required to collect physical demonstrations for a task where a synthetic dataset is not available or sufficient for machine learning.

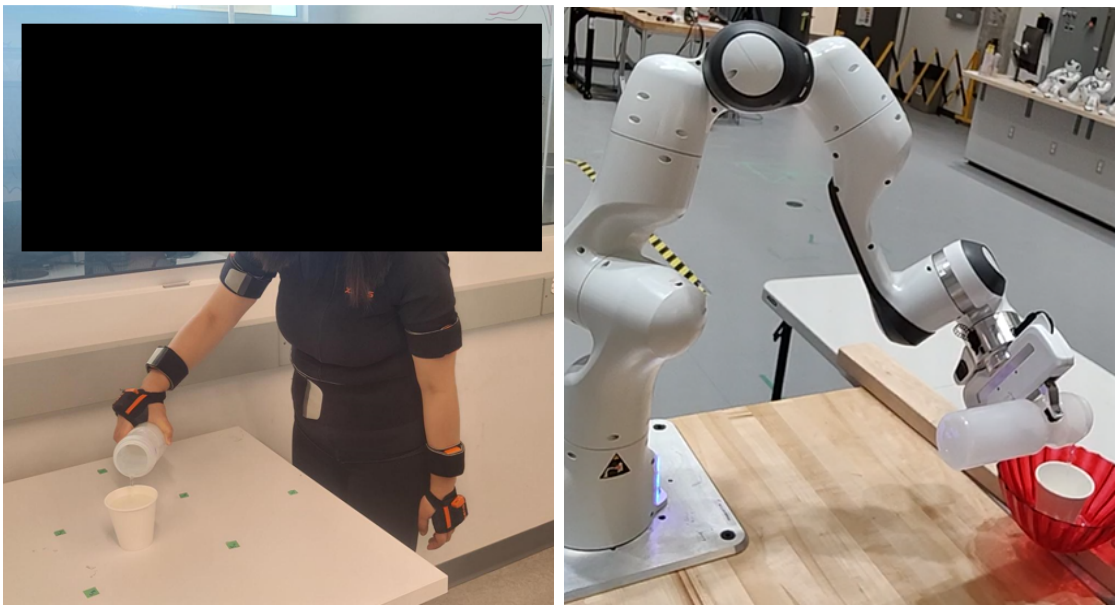


Figure 1.1: Demonstrator performing a pouring task (left) and robot performing the learned pouring task (right).

The framework was tested using a pouring task for a robotic manipulator. A container with water was placed in a location in the workspace, and the objective of the task was to fill another container in the workspace with water. This was selected as it is not a trivial task to model, but could be performed by most humans to reduce the difficulty of collecting demonstrations. Demonstrations were collected from multiple demonstrators, and the data was augmented to improve the generalizing capability of this model. Images of a person performing a demonstration and of the robot replicating the task are shown in fig 1.1. The learned skill was then implemented on a Franka Emika Panda robotic manipulator. While a specific task and robot platform were selected to evaluate this framework, it could be applicable to other applications and robots. The contributions described in this thesis have also been presented in a paper which has been submitted to the 2024 IEEE-RAS 23rd International Conference on Humanoid Robots (Humanoids).

1.3 Thesis Structure

This section provides an overview of the structure of the thesis, as well as a short summary of each of the chapters.

Chapter 2 explores works related to the topics being presented in this thesis, providing motivation for the elements used in the framework, and identifying knowledge gaps. The basics of learning from demonstration and dynamic motion primitives are presented, along with some examples of how DMPs have been extended. Methods used to generalize DMPs over multiple demonstrations are explored, along with the potential of neural networks combined with neural DMPs. Finally, methods used to collect datasets for learning from demonstration are analyzed, and summary of the techniques used in the thesis is presented.

Chapter 3 presents the theoretical formulation of neural DMPs and how they are integrated into the control architecture of the robot. It begins with the formulation of classical DMPs, and how they can be combined with neural networks. Next it describes the loss functions and methodology used for training, inluding techniques that were employed to reduce over fitting. Finally, it describes the techniques used to sequence several DMPs, and how they are integrated into the control of the robot.

Chapter 4 describes the method used to collect demonstrations and how the datasets are formulated. It begins by describing the configurations of containers and water levels used in the different datasets, then presents the method used to segment the demonstrations. Finally, it explains the data augmentation techniques used to increase the size of the datasets, and prevent over fitting.

Chapter 5 presents results of the neural DMPs for samples seen during training, and new samples not seen during training. This includes the average losses over the training and testing datasets, as well as showing specific trajectories from individual samples. This chapter also provides a comparison to similar existing works, and presents trade-offs between the two techniques used in this paper.

Chapter 6 shows the results of the models and control framework being implemented on a physical robotic system. This includes plots to show that the robot end effector tracks the desired outputs from the models, as well as sequences of images to show the task being completed by the robot. It finished by making remarks regarding the robot's performance, and comparing the different models.

Chapter 7 summarizes and describes potential future directions for the work presented in this thesis.

Chapter 2

Literature Review

This chapter examines works related to this project and presents the knowledge gaps in the current research landscape. It begins with examining learning from demonstration and different methods that can be used to learn from humans. Next DMPs are introduced, along with some of the existing variations to their formulations. Methods to learn from multiple demonstrations and to complete complex tasks composed of multiple sub-tasks are explored. It finishes with an overview of different methods of collection demonstrations, and summary of the methods used for the presented framework.

2.1 Learning From Demonstrations in Robotics

Learning from demonstration is an expansive field in the context of robotics that can utilize demonstrations collected in a variety of different ways, and encompasses different classes of algorithms [20], [40]. It is referred to by many different names, such as behavioural cloning and imitation learning. Two notable categories of algorithms are dynamics based algorithms, which model skills using dynamic equations, and stochastic methods, which generate policies using statistical representations [53], which can be used to represent low level skills and to plan high level tasks.

While there are many different methods of programming by demonstration for robotics, they can typically divided into statistical based and dynamics based methods [53]. Statistics based methods typically rely on hidden Markov models [24] [55] or Gaussian mixture models (GMMs) [7]. These methods allow for learning across multiple demonstrations to account for variations, but can require large datasets. Dynamics based systems, such

as Dynamic Motion Primitives (DMPs) [45], encode the dynamics of the demonstration. They have the advantage stability and spatial and temporal generalizability increasing their robustness in uncertain environments [43]. The developed systems aims to utilize the benefits of DMPs, while adding the capability to incorporating multiple demonstrations during the learning process to learn the underlying objectives of the task independent of demonstrator specific nuances.

Task planning in robotics using learning from demonstration often combines high level planning and representations of low level skills [40]. Xie et al. [59] demonstrate how both the high level planning and individual motion primitives can be learned from demonstration for bimanual tasks. Their two model framework, one to learn high level plans and the other to learn motion primitives, was able to perform peg in hole and table lifting tasks in simulation. Luo et al. [30] provide another framework where both high level planning and motion primitives are learned. Their system incorporates additional features, namely the ability to detect failures after each step to increase the likelihood of overall task success. The system presented in this thesis focuses on combining learning a full sequence of lower level motion primitives, removing the need for a distinct sequencer and set of motion primitives by combining them into a single primitive through machine learning. That being said, a simple motion primitive sequencer is used to test chained motion primitives to generate a comparison with a currently popular method.

2.2 Trajectory Representation and Generation Using Dynamic Motion Primitives

Dynamic Motion Primitives (DMPs) are a versatile method of learning from demonstration that represents trajectories using second-order dynamical systems [45]. They are composed of a set of basis functions, learned weights, and attractor dynamics which allow a model to be trained from a single demonstration. DMPs have been formulated for different spaces, such as joint spaces or Cartesian spaces and have been augmented to incorporate environmental feedback and external stimuli [43]. They are a proven method of effective learning from demonstrations, but can have certain drawbacks, such as only learning from a single demonstration.

Selecting the correct base DMP formulation is based on the nature of the task and trajectories to be learned. While traditional DMPs are formulated for joint or position control, formulations for both quaternion [1] and rotation matrix based orientations [56], as well as combined positions and orientation for Cartesian spaces [44], [48] have been developed.

Joint space DMPs tend to be popular when recording robot joint states during demonstrations, such as for kinesthetic teaching or teleoperation based interfaces [21]. While joint space DMPs can be used with passive observations, this often requires overcoming the correspondence problem, or mapping human joints to robot joint space, which can be challenging due to different kinesthetic makeups and redundancies [40]. As such, position, orientation or Cartesian space DMPs are popular when humans perform the demonstrations independent of the robot. Additionally, DMPs can represent point to point motions, periodic motions, or even have goals with a non-zero velocity [43]. While each method is rooted in similar theoretical formulations it is crucial to select the correct DMP for a particular task representation. For this project, Cartesian DMPs were selected as demonstrations were collected using passive observation, and control of the position and orientation of the end effector is required to complete the task. Point to point DMPs were selected as the demonstrations and segmented demonstrations had zero starting and ending velocities and accelerations.

Augmentations to DMPs that incorporate external feedback have been investigated to improve their versatility and robustness for use in real world applications. Planning and obstacle avoidance is a common problem in robotics, and has been incorporated into DMPs using sensor feedback and a coupling term [39]. Obstacle avoidance can also be incorporated using potential fields which rely on position [14], can incorporate velocity [15] and can utilize model predictive control [18]. This can be further extended to avoid collisions along an entire kinematic chain by combining Cartesian and joint space DMPs [25]. Another modification that can increase the robustness of DMPs are systems which can adapt to changing goals in real time [2], and expand the variety of tasks suited for DMPs. Force feedback has also been incorporated into DMP structures, further expanding the usefulness of DMPs to human interaction and contact rich tasks [13].

While modifications to adapt to uncertain environments serve to improve the robustness of DMPs, applying them to more complex tasks requires additional strategies. The integration of via-points into the DMP formulation allows further control when specific and known points in a trajectory are critical for task execution. These can be incorporated based on knowledge of the desired trajectory [64], and have also been implemented at run time for dynamic obstacle avoidance or changing environmental factors [50]. However, it should be noted that even with via-points, the amount which a motion primitive can be adapted to meet the constrained point in the trajectory is limited [64]. Further control over tasks with multiple distinct steps can be obtained by training and executing multiple consecutive DMPs is a common strategy [44]. While the segmentation can be done autonomously, typically using zero velocity crossing [6], this can require human oversight, which can become more difficult to obtain when highly trained experts are required. When

combined with a higher level planner which selects from a library of primitives [30], [59], this can provide further task generalization and adaptability. However, the system is still limited to the primitives that it has, and requires processing of environmental data to select the correct primitive.

Adding more control to DMPs through via points and DMP chaining introduces additional control for tasks, but require more effort to implement and train. Sidiropoulos & Doulgeri [50] utilize via points at real time, but this requires monitoring the environment for obstacles or changes in the environment that are known to impact the current task. Similarly, static via-points requires known points critical to the trajectory, which need to be selected in some manner [64]. While DMP chaining can be done through velocity zero crossing [6], it may still require human oversight to correct segmentation, or when specific augmentations are required. These techniques reintroduce task knowledge hand human input, albeit to a lesser extent, which are attempting to be avoided when using learning from demonstration.

2.3 Learning Generalized Tasks using Dynamic Motion Primitives

A key feature of DMPs is that they learn from a single demonstration which provides both advantages and disadvantages. Data driven models which to not require large amounts of data are useful in machine learning to reduce the required time and effort to collect datasets. In robotics, particularly for techniques that collect demonstrations from the robot, the ability to learn from small datasets is even more advantageous as it can reduce physical wear on the system. However, it is often desirable to learn using multiple demonstrations in order to integrate characteristics of different experts and generalize over tasks. As such, different methods have been explored to allow DMPs to learn from multiple demonstrations.

A variety of techniques have been used to train DMPs from multiple demonstrations. Early methods have explored adding an additional term to the DMP known as the style parameter [31]. Other approaches focused on training multiple DMPs and finding locally optimal solutions through quadratic optimization [22], [23]; however, these methods tend not to scale efficiently, as training and storing DMPs for each demonstration is required. A combination of Gaussian mixture models (GMMs) and Gaussian mixture regression (GMR) [8] have also been used to train DMPs for multiple demonstrations, including demonstrations of correct and incorrect behaviours [10]. Using DMPs along with a learned cost function in model predictive control (MPC) has also been used [18] and has been shown

to scale more efficiently with the number of demonstrations. Linear regression has also been used to learn unique weights throughout a task space based on multiple demonstrations [16].

Currently, neural networks are a popular method being used to generalize over multiple demonstrations [42]. Often referred to as neural DMPs, this techniques uses a neural network to approximate the weights of a DMP using environmental factors as inputs and making use of trajectory representation characteristics of the DMPs. End to end policies, which take images of the environments have been developed to avoid hand crafted features [5]. Neural DMPs for orientation are less common, but have been investigated [48], and the weights are learned by comparing the forcing functions. Different types of neural networks including traditional networks and recurrent neural networks (RNN) [38], as well as convolutional neural networks (CNNs) [36] have been used.

Different loss functions have been used while learning the weights for DMPs from neural networks. When this is done, it is crucial that it is differentiable with respect to the desired outputs, generally the weights and desired goal position, to ensure back-propagation can occur in the network. The forcing functions from the learned model and the demonstrations have been used to generate the loss function and neural network weight updates for both position and orientation representations [48]. Another method computes the loss function for training a position neural DMP by comparing the output trajectory to the demonstration, as opposed to comparing the weights which do not have a physical meaning [33], and is found to increase the performance of the networks. Auto-encoders have also been used to learn latent space representations of tasks for motion generation [9], so are also able to compare reconstructed trajectories to demonstrations. For this system, the loss function for the position was determined by comparing the demonstration trajectory and the modeled trajectory as in [33], while the orientation loss function compared the learned and modeled forcing function as in [48].

2.4 Methods for Demonstration Collection

Selecting a method to use to collect demonstrations is a crucial step in developing a learning from demonstration architecture [11]. The selected method will influence the type and quality of data, based on its capabilities and limitations. While synthetically generated data sets are convenient for some tasks, physical demonstrations from humans are often required. When this is the case, common collection methods include direct methods, which include kinesthetic teaching and teleoperation, and passive observations [49] [65], each of

which provide different benefits and drawbacks. In some cases, data augmentation can be used on physically collected datasets to increase the number of samples.

In some cases, synthetically generated data sets can be used to overcome the the barriers to demonstration collecting. Pahič et al. [33] use the MNIST data set to automatically generate writing trajectories when collecting demonstrations on how to write numbers. This allowed for efficient collection of thousands of samples which could be used to train and test models. Another method, used Bahl et al. [5], is to use a reinforcement learning agent to collect samples to train their neural DMP to write numbers. Both of these methods were successful, however the task was relatively simple and were helped by the existence of a large and available available dataset.

When tasks become more complex, or existing datasets are not available, collecting demonstrations based on human input is required. This is feasible for some methods such as GMMS [8] or optimization based approaches [23] that require fewer demonstrations. Methods for learning from existing or recorded videos have also been presented [29], but can still require large investments of resources to collect if no datasets already exist. However, it has also been shown that it is possible to train deep imitation learning models using auto-encoders and DMPs from motion capture data [9] and from teleoperation to train skills based on video inputs [62].

Kinesthetic teaching is a well investigated method of generating demonstrations for imitation learning and learning from demonstration. This involves a skilled user physically guiding a robot through the completion of a task, during which the robot and environment states are recorded [52], [21]. This has the benefit of ensuring that the demonstrations maintain kinematic feasibility on the robot, and that the demonstrations can be directly related to the final robot motions. However, the human presence in the robots workspace can add artifacts to some sensor data. Additionally, acquiring high quality demonstrations often requires both expertise in the task being demonstrated as well as familiarity with guiding the robot. [6].

Teleoperation based demonstration collection is another common method used for learning from demonstrations in robotics. In this method, the demonstrator controls the robot through a teleoperation interface to complete the desired task [49]. Similarly to kinesthetic teaching, the demonstrations are guaranteed to be feasible for the robot kinematics, but requires expertise in the system being used and task being completed. While this does remove the need for the demonstrator to be present in the same physical space as the robot [35], they still need to have access and perform the task on the robotic system. This can lead to shared control frameworks which increase the amount of automation to reduce user workload, but allows users to maintain some control [61].

Passive observation as a method of demonstration collection provides additional trade-offs. Demonstrations are collected using a variety of sensors, such as motion capture [8] or raw images [38], and can include full body motions, or motions limited to specific limbs [19]. As the demonstrations are not collected directly from the robot, further data processing is often required to ensure kinematic feasibility and to map human motions to robotic systems. However, passive observation typically imposes the fewest restrictions on the demonstrator when compared to kinesthetic and teleoperation based collection systems. Additionally, it does not require the demonstrator to have expertise in operating or physically guiding a robot. [58]

When the number of required samples exceeds the limitations of physical collection, data augmentation can be used to increase the number of samples. Prados et al. [37] explore methods of training an agent of demonstrations from humans, which allows for the generation of new trajectories. Misimi et al. [32] utilize two main methods to increase the robustness of their dataset, including shifting and rotating individual demonstrations within the workspace, and applying Gaussian noise. Not only did this serve to increase the number of samples, but added to the variety in the data seen while training the policy. This could be particularly useful in cases where demonstrations are difficult to obtain, such as for complex or dangerous tasks, or when highly skilled experts are required to perform them.

2.5 Techniques Used and Knowledge Gaps

This paper uses state of the art position and orientation formulations for neural DMPs as a learning from demonstration method to learn a task that could be split into multiple sub-tasks. This addresses a gap in current research, which currently focuses on utilizing neural DMPs to learn from multiple demonstrations, but does not fully utilize the generalization capabilities of neural networks to learn more complex tasks. As such the framework utilises models that learn from multiple demonstrations and different demonstrators without utilizing task knowledge to segment or annotate the demonstrations in any way. This is compared to the well explored method of sequencing multiple DMPs to complete the same task. Analysing the trade-offs between the two methods aims to assess whether the proposed framework could be a useful alternative to existing methods.

The demonstrations used to train the models in this thesis are collected through passive observation from human demonstrators. While methods that use neural DMPs rely on synthetically generated data, the method presented in this thesis used data augmentation techniques to increase the number of samples. This reduces typically high amount of

resources required to collect physical demonstrations, and has the potential to expand the capabilities of learning from demonstration techniques to new tasks.

Chapter 3

Formulation and Training of Dynamic Motion Primitives and Robot Control

This chapter describe the theoretical formulation and training methodology of this framework. The DMP formulations are presented, along with the loss functions for training and evaluating the resulting models. The method used to train the models is described, and the control architecture for the implementation on the physical robot is presented.

3.1 Formulation of Dynamic Motion Primitives for Position Representation

Dynamic motion primitives are a method of representing trajectories using second order dynamic equations originally introduced for humanoid robots [45]. The classical formulation for DMPs is briefly explained in this section [43].

Equations 3.1 and 3.2 show the transformation system and auxiliary variable respectively. Together, these form a second order differential equation which defines the shape of a trajectory while ensuring convergence to goal.

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z) + f(x) \tag{3.1}$$

$$\tau \dot{y} = z \tag{3.2}$$

where τ is the time scaling factor, g is the final goal point, y is the trajectory variable and z is the auxiliary variable as formulated in [43]. α_z and β_z are hyper-parameters which create a critically damped system when $\alpha_z = 4\beta_z$. The variable x is the canonical variable that serves to make the transformation system independent of time through the canonical system given by equation 3.3. This allows for easier deployment, analysis and control over the properties of the DMP. The forcing term $f(x)$ defines the shape of the trajectory between the initial position and the goal through a set of weights learned from a demonstration and a corresponding set of basis functions, and its formulation is shown in equation 3.4.

$$\tau \dot{x} = -\alpha_x x \tag{3.3}$$

The canonical system shown in eq. 3.3 is an exponential function that is described by the parameter α_x. Equation 3.4 shows the forcing term.

$$f(x) = \frac{\sum_{i=1}^{N} \omega_i \boldsymbol{\psi}_i(x)}{\sum_{i=1}^{N} \boldsymbol{\psi}_i(x)} x \tag{3.4}$$

It is composed of a set of Gaussian basis functions $\boldsymbol{\psi}(x)$ and learned weights ω_i, where N is the number of basis functions used. The basis functions are given by

$$\boldsymbol{\psi}_i(x) = exp(-h_i(x - c_i)^2) \tag{3.5}$$

where each c_i represents the center of a Gaussian basis function, and h_i controls its width. When τ is set to the duration of the demonstration, and N is known, the centers of the basis functions can be evenly spread by using

$$c_i = \exp(\frac{\alpha_x(i-1)}{N-1}) \tag{3.6}$$

$$h_i = \frac{1}{(c_{i+1} - c_i)^2}, \; h_N = h_{N-1} \tag{3.7}$$

To represent trajectories in multiple dimensions, multiple DMPs can be used and synchronized through a common canonical system. The weights are typically learned through

a Linear Weighted Regression (LWR) from a single demonstration, which performs a type of regression on the data where the points in the trajectory located closer to the query point receive a higher weight [4].

3.2 Quaternion Formulation of Dynamic Motion Primitives for Orientation Representation

Different forms of dynamic motion primitives have been formulated to represent orientation in Cartesian space, including methods using quaternions [9] and rotation matrices [56]. The quaternion formulation is used in this framework, and was selected to have a method of representing orientations without singularities. Additionally, it only uses four parameters as opposed to the nine required for rotation matrices [56]. The formulation is described here, where a quaternion is represented by a vector in $\boldsymbol{q} = w + \boldsymbol{u} \in \mathbf{S}^3$ where w is the real component and $\boldsymbol{u} \in \mathbf{R}^3$ are the imaginary components. Equations 3.8 and 3.9 show the transformational system for the quaternion DMP.

$$\tau \dot{\boldsymbol{\eta}} = \alpha_q(\beta_q 2\text{Log}^q(\boldsymbol{g} * \bar{\boldsymbol{q}}) - \boldsymbol{\eta}) + \boldsymbol{f}_q(x) \tag{3.8}$$

$$\tau \dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{\eta} * \boldsymbol{q} \tag{3.9}$$

Here, $\dot{\boldsymbol{\eta}} \in \mathbf{R}^3$ represents the scaled angular velocity, $\dot{\boldsymbol{q}} \in \mathbf{R}^4$ is the quaternion derivative, $\bar{\boldsymbol{q}}$ is the quaternion conjugate, and α_q and β_q are parameters to tune the DMP. The $*$ operation represents a quaternion multiplication, and the quaternion logarithm $\text{Log}^q(\cdot)$ is shown in equation 3.10, in which $||\cdot||$ is the $l2$ norm of a vector.

$$\text{Log}^q(\boldsymbol{q}) = \begin{cases} \arccos(w)\frac{\boldsymbol{u}}{||\boldsymbol{u}||}, & \boldsymbol{u} \neq \mathbf{0} \\ [0\ 0\ 0]^T, & otherwise \end{cases} \tag{3.10}$$

The quaternion DMP uses the same canonical function, forcing term and basis function center and width equations shown in equations 3.3, 3.4, 3.6 and 3.7 respectively. As with classical DMPs, LWR is often used to learn the weights of the forcing term.

3.3 Neural Dynamic Motion Primitives

Methods of generalizing DMPs over multiple demonstrations have been studied, and range from using Gaussian Mixture Models [8] to optimally interpolating between different DMPs [23]. Additionally, complex motions have been achieved by chaining DMPs [44] or combining via points during training [50]. Combining neural networks with DMPs, also referred to as neural DMPs [42] is another promising method that can be used to generalize over multiple demonstrations. Neural DMPs have the potential to learn more complex trajectories without segmenting data as is required by some methods that utilize via-points [64].

This section overviews the loss functions used for both the position and quaternion neural DMPs. Some methods use neural networks to predict the forcing terms for Cartesian DMPs composed of both position and orientation [44]. However, it has been demonstrated that position models can have higher performance when their loss function is based on the Euclidean distance between the desired and predicted trajectories [33]. As research regarding similar formulations for quaternion DMPs is limited, and developing such a method is beyond the scope of this thesis, different loss functions were used to train the position and orientation neural DMPs. This means that while the outputs from both models are weights for DMPs, the position model is trained by comparing the actual trajectory to the demonstration, while the orientation model is trained by comparing the forcing terms between the predicted DMP and a DMP learned directly from the demonstration. This difference is demonstrated in the loss functions presented in this section, and in the formulation of the datasets in section 4.3.1.

3.3.1 Position Model

Neural networks that use the Euclidean distance between the trajectory and the predicted trajectory have been shown to be successful, and the parameters have been shown to the differentiable with respect to the loss allowing for the use of backpropogation [5], [33]. As such, the network is trained by predicting a set of parameters for the DMP, then simulating the resulting output trajectory through numerical integration. The loss is then calculated based on the difference between the predicted and demonstrated trajectories, as shown in equation 3.11:

$$L_{position,j} = \frac{1}{T_j} \sum_{i=1}^{T_j} ||\boldsymbol{y}(x_{i,j})_j^{DMP} - \boldsymbol{y}_{i,j}^{demo}|| \tag{3.11}$$

T_j is the number of points in the j_{th} sample, $\boldsymbol{y}(x_{i,j})^{DMP} \in \mathbf{R}^3$ and $\boldsymbol{y}_{i,j}^{demo} \in \mathbf{R}^3$ are the Cartesian position predicted by the DMP and from the demonstration respectively.

3.3.2 Orientation Model

Combining neural networks with DMPs for orientation involves learning the parameters of the DMP to approximate the forcing term. It has been shown that the weights for DMPs can be attained using a gradient descent approach on demonstration data [48]. Similarly to the position model, the method used in this paper uses a neural network to predict the basis function weights and goal orientation. However, as opposed to predicting the orientation, the forcing term is predicted and used in the loss function as shown in equation 3.12.

$$\begin{aligned} L_{orientation,j} =& ||\boldsymbol{q}_{g,j}^{NN} - \boldsymbol{q}_{g,j}|| + \\ & (\frac{1}{T_j} \sum_{i=1}^{T_j} ||\boldsymbol{f}(x_{i,j})^{NN} - \boldsymbol{f}(x_{i,j})||) \end{aligned} \tag{3.12}$$

where $\boldsymbol{q}_{g,j}^{NN}$ and $\boldsymbol{q}_{g,j}$ are the predicted and actual final orientations, $\boldsymbol{f}(x_{i,j})^{NN}$ is the forcing term generated using the predicted DMP weights, and $\boldsymbol{f}(x_{i,j})$ is the forcing term generated using weights learned from LWR. Methods for directly comparing the quaternion trajectories in the loss function of a neural network are not extensively researched, and developing this method is beyond the scope of this paper. However, when evaluating the final performance of the model's orientation trajectories, the logarithmic mapping shown in equation 3.10 is used.

3.4 Model Training

This section describes the method of training the neural DMPs. It begins by explaining how cross validation was used to select parameters that prevented over fitting, and how these were translated to the final model. It also describes the hyperparameter tuner that was used.

3.4.1 Using Cross Validation for Reduced Overfitting

This section describes the cross validation method used to select the hyperparameters for the model that would reduce overfitting. Originally, samples were selected at random to

be removed from the training data sets to be added to the validation data sets. However, it was found that while the training and validation sets appeared to have similar errors, the test set with completely new trajectories performed significantly worse. As such, the validation set and training strategy was modified to better select hyperparameters which could generalize to new inputs.

The hyperparameter selection for the models was done using a k-fold cross validation. For each fold, the validation data set was separated from the training data set by removing all trajectories from specific configurations. The specifics of the configurations, and how they were seperated for each fold is further described in chapter 4. A model was then trained on a subset of configurations, and validated using the remaining configurations. The training and validation losses used for evaluation of the models was an average across all folds. The number of epochs for which the model was trained was included in the hyperparamters, and was used to train the final model. The process of seperating the demonstrations for this project is further discussed in section 4.3.3.

Once the hyperparameters and number of epochs had been identified, the final model was trained using the full set of training data. This combined all the configurations in the training set to utilize all available data, and increase the models ability to generalize. The final model was trained for a number of epochs found while training the cross validation models to prevent over fitting.

3.4.2 Hyperparameter Tuning

To improve the efficiency of the model training process, the hyperparameter tuner Ray Tune [28] was used. This allowed for the definition of search spaces, and the built in Asynchronus Hyperband Scheduler was used. This terminated under-performing models early, allowing for the testing of more configurations. It also allowed for the implementation of early stopping, which analysed the validation metric over several epochs and terminated the model once the change in loss dropped below a specific threshold. The hyperparameter tuner evaluated models based on the average validation loss over a model trained on each of the folds.

A variety of parameters were tuned using the hyperparameter tuner. In addition to the number of epochs, the number of layers and layer sizes, learning rate and other optimizer parameters such as weight decay were tuned. Due to the difference in formulation of the loss function, some parameters, such as the β parameter in the DMP and the number of basis functions were only tunable by the hyperparameter tuner for the position model. This was tuned manually during dataset creating for the orientation models. Additional

parameters included the standard deviation and number of previous epochs used to check for early stopping were selected through trial and error.

3.5 Chaining Consecutive DMPs

The transition between consecutive motion primitives for robotic systems requires care to ensure the output accurately represents a coherent task and is within the physical constraints of the robot. As such, two main concerns when chaining multiple dynamic motion primitives are ensuring a smooth transition between consecutive primitives and guaranteeing that each primitive has completed before proceeding to the next. This is done through a combination of analyzing the output of the DMP, as well as its first and second derivatives to ensure a smooth transition, while monitoring the canonical system of the DMP to track completeness.

Ensuring a smooth transition between primitives is essential when chaining together motions for more complex tasks. A smooth transition consists of continuity of the DMP's output variable, along with its first and second derivatives at each transition point. Without this, the chained DMPs will not accurately represent the desired task, and have the potential to either damage the robot hardware, or trigger internal safety mechanisms in the robot's system.

The classical formulation of DMPs imposes several constraints which can be leveraged when chaining consecutive DMPs. Namely, the standard formulation of position and orientation DMPs represent trajectories with first and second derivatives of zero at the start and end of the trajectory. As such, if these formulations are used, as is the case for this system, a DMP can be started when the velocity of the previous DMP has dropped below a threshold, and sufficient time has passed to allow the DMP to progress through its full learned motion. Continuity in the position of the primitives can be ensured by initializing each new DMP with the final position reached by the previous primitive.

Ensuring the DMP has had sufficient time to complete its learned motion is essential, particularly with primitives that have varied first and second derivatives. Motions may include periods where the velocity approaches or crossed zero, but is not coming to the end of the primitive. If velocities alone are checked, these points in the trajectory would risk being evaluated as ending points, and parts of the trajectory would be missed.

The canonical system can be used to track the completion of the DMP. With the time scaling properties inherent in DMPs, the canonical variable will decay relative to the progression of trajectory, and independently to the passing of time. As such, the final value

of the canonical value can be calculated given the length of the demonstration and learned trajectory, and this value can then be used to check for the completion of the primitive regardless of what time scaling has been applied to the primitive.

The analysis of the outputs of the DMP and the monitoring of the canonical system are combined during the transition. First, the canonical system is tracked, and when it has decayed below the threshold calculated based on the demonstration, this signals that the primitive has reached its conclusion. At this point, the primitive continues until the first and second derivatives are below a threshold value. Once these values have dropped below their respective thresholds, the next DMP is initialized with the final output of the newly terminated primitive, ensuring a smooth transition from a fully completed motion.

The outputs of the DMP along with its first and second derivatives can be used to ensure smoothness between primitives. The canonical system provides a time independent method of assessing when a primitve has completed.

3.6 Robot Control

An overview of the entire framework, including the execution on a physical robot, can be seen in fig. 3.1. This demonstrates the process by which the input features are transformed by the neural networks to generate DMP parameters based on the current environment, and used to create Cartesian trajectories. The Cartesian space trajectory was converted into joint space using the Rigid Body Dynamics Library (RBDL) [12], and used to command the robot via joint position control (this was chosen for this paper, but different control modalities may also be employed).

The joint impedance controller for the robot is a standard PD joint impedance controller with compensation for gravity and Coriolis forces formulated as in equation 3.13. An impedance controller was selected to make the system more robust to the contacts that could occur with the table and container during this task.

$$\boldsymbol{\tau} = K_{coriolis} * \boldsymbol{\tau}_{coriolis} + \boldsymbol{\tau}_g + K_p(\mathbf{q}_d - \mathbf{q}) + K_d(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) \tag{3.13}$$

In this formulation, $\boldsymbol{\tau} \in \mathbf{R}^n$ is a vector of n joint torques, $K_{coriolis}$ is a tunable parameter to account for errors in Coriolis for estimation and $\boldsymbol{\tau}_{coriolis} \in \mathbf{R}^n$ and $\boldsymbol{\tau}_g \in \mathbf{R}^n$ are vectors of the joint torques due to the Coriolis forces and gravitational forces respectively. K_p and K_d are the proportional and derivative tuning constants. The variables $\mathbf{q}_d \in \mathbf{R}^n$ and $\mathbf{q} \in \mathbf{R}^n$

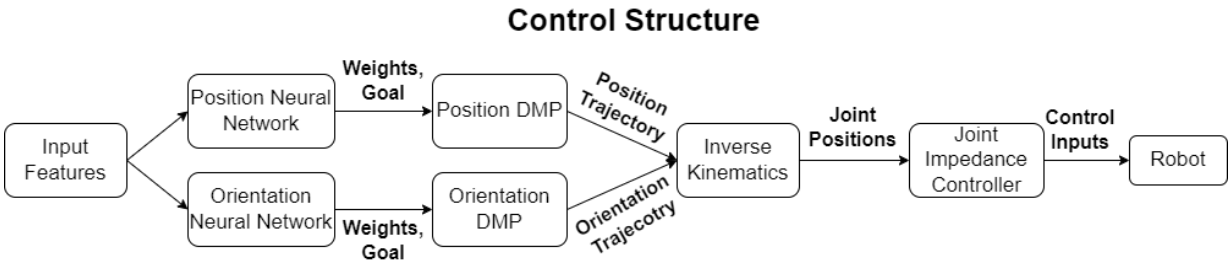


Figure 3.1: Control structure overview, showing the transformation of the input features through the position and orientation neural networks to predict the weight and goal parameters of their respective DMPs, which then generate trajectories. Inverse kinematics generate a joint position trajectory from the Cartesian trajectory, which the robot follows using a joint impedance controller.

are vectors containing the desired and actual joint positions respectively, while $\dot{\mathbf{q}}_d \in \mathbf{R}^n$ and $\dot{\mathbf{q}} \in \mathbf{R}^n$ are vectors containing the desired and actual joint velocities respectively.

As a rotational transformation was required to convert and align the axes of the hand from the Xsens IMUs from data collection to the robot end effector. The demonstration collection process is explained in more detail in section 4.1. This was implemented using rotations from the Scipy Python Library [57] through euler angle rotations.

Chapter 4

Demonstration Collection and Dataset Creation

This chapter describes how the demonstrations were collected and formulated into datasets which could be used to train the neural DMPs. It begins by describing the sensors used to collect the demonstrations and the details of the data collection process, including how the workspace was set up and what input features were recorded. Next, the process through which the demonstrations were segmented is described. This is followed by how the demonstrations were augmented, the size of each of the datasets how how the datasets were formulated to train the neural DMPs.

4.1 Demonstration Collection

Demonstrations were collected using the Xsense Awinda motion capture suit and software, which uses Inertial Measurement Unit (IMU) data to estimate human poses [47] at 60 frames per second. The sensors are worn by the demonstrator which is shown in fig 4.1. The Cartesian position and orientations were extracted by the Xsense Awinda software relative to a coordinate frame set during the calibration of the sensors.

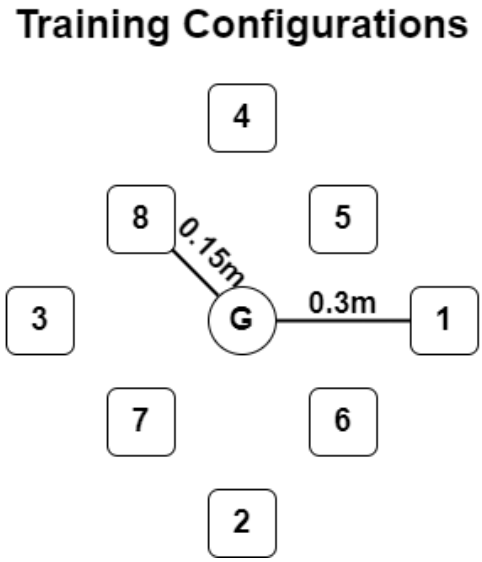
Demonstrations for the training set were collected from 4 demonstrators, each of which performed a pouring action in 8 different configurations, shown in fig. 4.2a, for three different amounts of water. Fig. 1.1 depicts an actual demonstration data collection session. The water levels were measured using weights, and had the following values: 799g, 522g, and 273g. One participant collected demonstrations from an additional 4



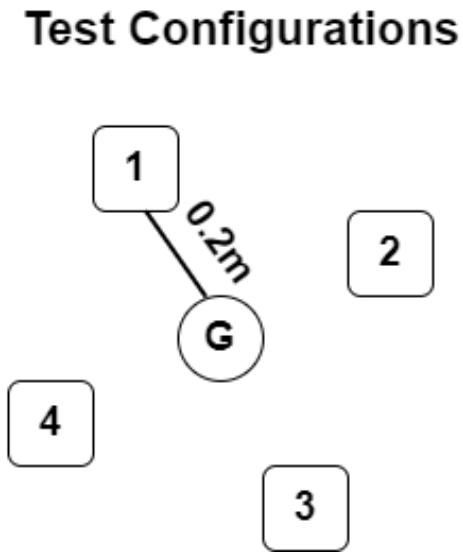
Figure 4.1: The front (left) and back (right) of how the Xsens Awinda IMU motion capture sensors are worn by the demonstrator.

configurations shown in fig. 4.2b and an additional 2 water levels of 397.5g and 660.5g to be used as a testing set. All our experiments received ethical approval from the University of Waterloo Human Research Ethics Board at the University of Waterloo, Ontario, Canada. The letter of approval for the demonstration collection is shown in Appendix A. Before the experiment, participants received proper information and gave informed consent to participate in the study.

One demonstration consisted of the demonstrator starting with their hands at their sides, reaching for the pouring container, pouring into the goal container until they judged it to be full (as only the 273g level could be entirely poured into the goal container) or the pouring container was empty, return the pouring container to its original location, and return their hand to their side. The demonstrators were also told to grip the container in a specific region that was suitable for a robot gripper to grasp on. While it was not a specific exclusion criteria, each of the demonstrators were right handed, and used their dominant hand to pour. The position of the pouring container was varied around the goal container as seen in fig. 4.2a and 4.2b, where goal position is shown as a circle, while the pouring container positions are shown as rectangles.



(a) Configurations used to collect demonstrations for training



(b) Configurations used to collect demonstrations for testing

Figure 4.2: Configurations with pouring container locations as rectangles and the goal container as a circle.

The trajectories were resampled using cubic splines to ensure consistent numbers of data points. The orientation trajectories of the demonstrator's pouring hand were used to learn a forcing function with LWR to create the labels for the orientation model, while the position trajectory of the demonstrator was used directly as the label for the positions model. The input features for the models were generated, and consisted of the starting position and orientation of the demonstrator's hand, the Cartesian position of the pouring container and the goal container, and the water level normalized to between 0 (empty) and 1 (largest water level). Additionally, the inputs included the position difference between the pouring container and the goal container, as well as between the starting hand position and the pouring container in each Cartesian axis.

4.2 Demonstration Segmentation for Multi-Stage Dynamic Motion Primitives

The process of segmenting the data for the chained DMPs described in this section required the integration of task knowledge and human effort to be successful. This was done to generate DMPs for the sub-tasks in this task so that the models trained from full demonstrations could be compared to an existing method. If only the models trained on full demonstrations were needed, this process would not be required.

The collected demonstrations were segmented into three motions for use in the chained sub-task neural DMPs. The first segment included reaching for the container with water, and required the end effector to navigate the environment, avoiding the goal container. The second segment began when the manipulator grasped the container, and consisted of transporting the container to the goal container, pouring the water, and returning the container to the original location. This segment required the associated DMP to capture the pouring motion, including factors such as the angle at which the pouring container is tilted and the location of the end effector to successfully pour into the receiving container, while also navigating the environment to avoid collisions with the goal container. The final segment started after the pouring container has been released and required the manipulator to to return to its initial position without colliding with any containers. This section describes the motivation for how the task was segmented, and process used to segment the data.

One consideration when selecting how to segment the task was aligning the sections with the available inputs to the system. Each of the segments begins and ends at one of the locations specified in the inputs. The first segment, reaching for the pouring container,

begins at the initial position and ends at specified position of the pouring container. The third segment, returning from the pouring container to the initial position, uses the same two positions but in opposite order. The second segment, starts and ends at the initial position of the pouring container. Breaking the task into segments with defined starting and ending points based on information from the environment helped ensure the neural DMP had the necessary inputs required to learn the desired trajectories.

While the first and third segment of this task include a single reaching motion, the second segment is more complex but was not segmented further. This was done to avoid additional calculation and pre-processing of the demonstrations. For example, the second segment could be further divided into an addition three sections such as moving from the pouring container to the receiving container, performing a pouring motion, and returning from the receiving container to return the pouring container. However, this would require the calculation of additional starting and ending points, such as the position and orientation of the pouring container when it start to pour, which are not explicitly determined by the configuration of the containers.

The data was segmented through a combination of automated analysis and manual point selection on the velocity profile of the demonstration. After applying a low pass filter to the demonstration, the positional velocity in each Cartesian axis were combined to find the squared absolute velocity using equation 4.1.

$$vel = vel_x^2 + vel_y^2 + vel_z^2 \tag{4.1}$$

The selection of points was automated by using peak detection from the python library SciPy [57] to locate known features that were common between each of the demonstrations. The key features were determined by analysing the trajectory and velocity profiles, and were then used to determine an initial estimate of how to split the demonstrations. However, to ensure successful segmentation, human supervision was used to verify each segmentation and to manually correct any incorrect segmentations.

4.3 Dataset Formulation and Augmentation

This section first describes the formulation of the dataset so it can be used to train the neural DMPs presented in section 3.3. It also describes how the demonstrations are augmented to reduce the amount of data collection that is required, and provides the sizes of the datasets with and without augmentations. Finally, it describes how the datasets

were split into folds for the cross validation training method described in section 3.4.1. The position and orientation models use different datasets formulation to account for the different loss functions, described in more detail in section 3.3.

4.3.1 Dataset Formulation

The training datasets for the position and orientation models are described in this section. The position dataset was formulated as follows:

$$\{\boldsymbol{u}_j, \boldsymbol{y}_j^{demo}\}_{j=1}^{M} \tag{4.2}$$

where $\boldsymbol{u}_j \in \mathbf{R}^K$ is the vector of input features with K elements which are described in section 4.1. $\boldsymbol{y}_j^{demo} \in \mathbf{R}^{T_j \times 3}$ is a vector of containing each point in the j^{th} Cartesian position trajectories with T_j elements for M training pairs. For the orientation dataset, the input vector remains the same as the position dataset, however, the output consists of 2 components as shown in equation 4.3 below:

$$\{\boldsymbol{u}_j, (\boldsymbol{f}_j^{demo}, \boldsymbol{q}_{g,j})\}_{j=1}^{M} \tag{4.3}$$

where $\boldsymbol{f}_j^{demo} \in \boldsymbol{R}^{T_j \times 3}$ are forcing functions for each of the three dimensions used for the quaternion DMP formulation calculated using LWR. $\boldsymbol{q}_{g,j}$ is the final orientation in the form of a quaternion, and was added to the dataset so the neural DMP could learn the goal orientation as well as the weights, and used during the simulation of the DMP. The final orientation is included in the dataset so that the model can learn the goal orientation and predict the weights based on the forcing function. This is not needed explicitly in the position dataset as this model learns directly from the position trajectory which includes the final position.

4.3.2 Data Augmentation

Each collected demonstration was augmented by adding a position offset. The offsets generated a grid in the horizontal x-y plane of Cartesian space, and was incremented by a constant value. This offset was added to each point in the trajectory, along with the goal container position, the position of the filled container and initial position of the demonstrator's pouring hand. This increased the number of training, validation and testing samples in a manner that promoted generalization across the workspace. However, this

augmentation technique results in the model seeing a large number of identical inputs during training, and could result in over-fitting to specific input values.

If we take the full set of collected demonstrations, with the positions dataset represented as equation 4.4, it can be represented by eq. 4.4.

$$\{\boldsymbol{u}_c, \boldsymbol{y}_c^{demo}\}_{c=1}^{C} \tag{4.4}$$

Here, c is the configuration number, and C is the total number of configurations. After applying the grid of positional offsets, we get

$$\{\{\boldsymbol{u}_c, \boldsymbol{y}_{c,w}^{demo}\}_{w=1}^{W}\}_{c=1}^{C} \tag{4.5}$$

where W is to total number of positional offsets applied, and is given by the expression $W = num_x_increments * num_y_increments$ for the grid pattern that was used. This increases the number of sample from C samples to $W * C$ samples. However, this will result in a large number of repeated input features. For example, when incrementing the y position, the x position will remain constant. To train the models used in this thesis, the demonstrations were augmented with 7 increments spread between $-0.2m$ and $0.2m$. The number of samples in the full training and testing set are shown in table 4.1.

To reduce the over-fitting caused by the exposure of the model to repeated identical input values, a small random deviation was added to each of the environment parameters that made up the inputs. This included the position of the goal container and pouring container, along with the initial position and orientation of the demonstrator's pouring hand. The deviation was sampled from a normal distribution, and served to ensure the model could generalize to different starting and environment parameters. This results in a set of samples with the same structure as in equation 4.5, but now each $\mathbf{u}_c$ is a unique set of input features.

Dataset	**Sample before augmentation**	**Samples after augmentation**
Full Train	96	4704
Full Test	8	392
Fold Train	72	3528
Fold Validation	24	1176

Table 4.1: Number of samples in each dataset before and after augmentation.

4.3.3 Training and Validation Sets for Cross Fold Validation

With the augmented datasets collected, the training and validation datasets for the training process described in section 3.4.1. For this task, the training dataset was split into four folds, with each fold removing two configurations for the validation set, and the remaining six configurations were used for training. The configurations removed for each fold were selected to be far from each other to ensure the model's training samples were still distributed across the workspace. This resulted in the following pairing of configurations to remove for each fold, with the configuration number denoted with a capital letter "C" followed by location of the starting container given by 4.2a.

- **Fold 1:** C1 and C8 used for validation
- **Fold 3:** C2 and C5 used for validation
- **Fold 3:** C3 and C6 used for validation
- **Fold 4:** C4 and C7 used for validation

The datasets were created from the augmented dataset, and the number of samples in the training and validation sets for each fold can be seen in table 4.1.

Chapter 5

Analysing Neural DMP Performance

This section presents and analyses the models' ability to recall trajectories seen during training and generalize to new inputs. The recall performance of the models is evaluated by revisiting inputs seen during training, while their ability to generalize is assessed using trajectories not seen during training. Evaluating the generalization using a test dataset with inputs and trajectories unused during training allows for an assessment of whether or not the models behave in a similar manner to the demonstrator when presented with new task parameters. The models are compared by by analysing the average loss from the models over the full training and test datasets, and by examining sample trajectories from each of the datasets.

The selected configurations and water levels correspond with those presented in section 4.1. The sample configurations taken from the training dataset are denoted by a "C" followed by the configuration number from fig. 4.2a, while the samples taken from the test sets are denoted with a letter "V" followed by the configuration number in fig. 4.2b. Water levels are denoted with a letter "L", and levels 1-3 correspond to the levels seen during training from least water to most water, while levels 4 and 5 correspond to the levels in the test set from least to most water. The exact amounts of water for each level are state in section 4.1.

5.1 Model Losses

This section presents the average losses over all trajectories in the training and test datasets for the full and segmented models for both position and orientation. The average losses

	Position			
	Cross Validation		Full	
Model	Avg. Train	Avg. Val.	Train	Test
Full	0.00142	0.00239	0.00123	0.00289
Segment 1	0.00112	0.00127	0.00104	0.00139
Segment 2	0.000903	0.00155	0.000866	0.000904
Segment 3	0.00133	0.00153	0.00129	0.00151
Avg. Segments	0.00106	0.00148	0.00102	0.00118

Table 5.1: Training, validation and testing losses for each of the position models and the average loss across all segments weighted by segment lengths. Cross Validation losses are averaged across each fold shown in tab B.1 in Appendix B.

	Orientation			
	Cross Validation		Full	
Model	Avg. Train	Avg. Val.	Train	Test
Full	0.415	0.459	0.404	0.463
Segment 1	0.399	0.455	0.367	0.447
Segment 2	0.441	0.536	0.329	0.456
Segment 3	0.322	0.406	0.300	0.335
Avg. Segments	0.401	0.484	0.332	0.424

Table 5.2: Training, validation and testing losses for each of the orientation models the average loss across all segments weighted by segment lengths. Cross Validation losses are averaged across each fold shown in tab B.2 in Appendix B.

from the position models are presented in table 5.1 and the average losses from the orientation models are presented in table 5.2. Both of these tables show the average losses for each of the models, including the model learned from the full demonstrations and the models learned from each of the segments. The average over the entire trajectory of the segmented models, weighted by the length of each segment, is also presented as a method of evaluating the loss over the full trajectory for the segmented models.

The loss for the position models was calculated using the Euclidean distance between the predicted and demonstrated trajectories, while the logarithmic mapping shown in equation 3.10 was used to measure the distance between the demonstrated and predicted orientation quaternion trajectories. The losses were averaged over each trajectory, then over the full dataset.

The average losses for each of the models were larger for the test set than for the training set, including during the cross validation. This is expected from neural network models, and is further discussed in section 5.4. Additionally, by comparing the weighted average loss of the segmented models to the full model, we can see that the segmented version has a lower average loss.

5.2 Model Trained on Full Demonstrations

This section describes the performance of the position and orientation models trained using the full demonstrations. First, the performance of the models and samples selected from the training dataset is analysed to determine their ability to recall configurations. This is followed by analysing samples from the testing set that were not seen during training. By comparing the generated trajectories for configurations and water levels not seen during training, the models' ability to generalize to new inputs and replicate the actions of a demonstrator can be assessed. When selecting samples from the training and testing set, one trajectory from each water level, each with a different configuration of pouring and receiving container was selected. This resulted in three samples and configurations selected from the training set, and two configurations being selected from the testing set.

The position and orientation models trained on the full demonstrations generated using the training method described in section 3.4 had the parameters shown in table 5.3. A similar number of basis functions were used for the position and orientation models, and each neural network was composed of three layers. The number of epochs varied from 40 epochs for the position model, and 7 epochs for the orientation model. Rectified linear unit (ReLU) activation functions were used following each layer in each of the networks.

Parameter	**Position**	**Orientation**
# layers	3	3
layer sizes	300, 200, 300	700, 400, 300
β	25	25
epochs	40	7
# basis functions	50	50

Table 5.3: Model parameters for the position and orientation neural DMPs trained on the full demonstrations

5.2.1 Training Set Performance

The three configurations of the pouring and receiving container selected from the training set for visualisation each had a different water level, and were selected to analyse the recall of the models. The Cartesian components of the position trajectories from the demonstrations, and the corresponding position trajectories generated from the neural DMP for the selected samples are shown in fig. 5.1. The corresponding quaternion orientation trajectories from the demonstrations and the predicted orientation trajectories from the neural DMP are shown in fig. 5.2. In both cases, the demonstrated trajectory components are shown by a solid black line, while the predicted trajectory components are shown with a dashed red line.

From the selected trajectories, we can see the neural DMPs that used the full demonstrations while training are able to recall the shape of trajectories from the training set. They are able to reproduce the shape of the demonstrations. However, there appears to be more error near the end of the demonstrations in both the position and orientations plots in figures 5.1 and 5.2.

5.2.2 Testing Set Performance

The two configurations of the pouring and receiving container selected from the testing dataset had different water levels and were selected to assess the generalization capabilities of the models. The Cartesian components of the position trajectories from the demonstrations, and the corresponding position trajectories generated from the neural DMP for the selected samples are shown in fig. 5.3. The corresponding quaternion orientation trajectories from the demonstrations and the predicted orientation trajectories from the neural DMP are shown in fig. 5.4. As with the training set, the demonstrated trajectory components are shown by a solid black line, while the predicted trajectory components are shown with a dashed red line for both figures.

Figures 5.3 and 5.4 show that while the models trained from the full demonstrations are able to maintain the general shape of the demonstrations of an expert, there is increased error compared to the demonstrations from the training set. This is expected to a certain extent as a byproduct of neural networks, which are not able to generalize perfectly. However, this is also likely a result of the neural networks learning from multiple demonstrators. By incorporating demonstrations from multiple humans, the neural network should learn the common elements between the demonstrators. As such, it may cause the models to some miss elements from single demonstrators when generalizing, as it has

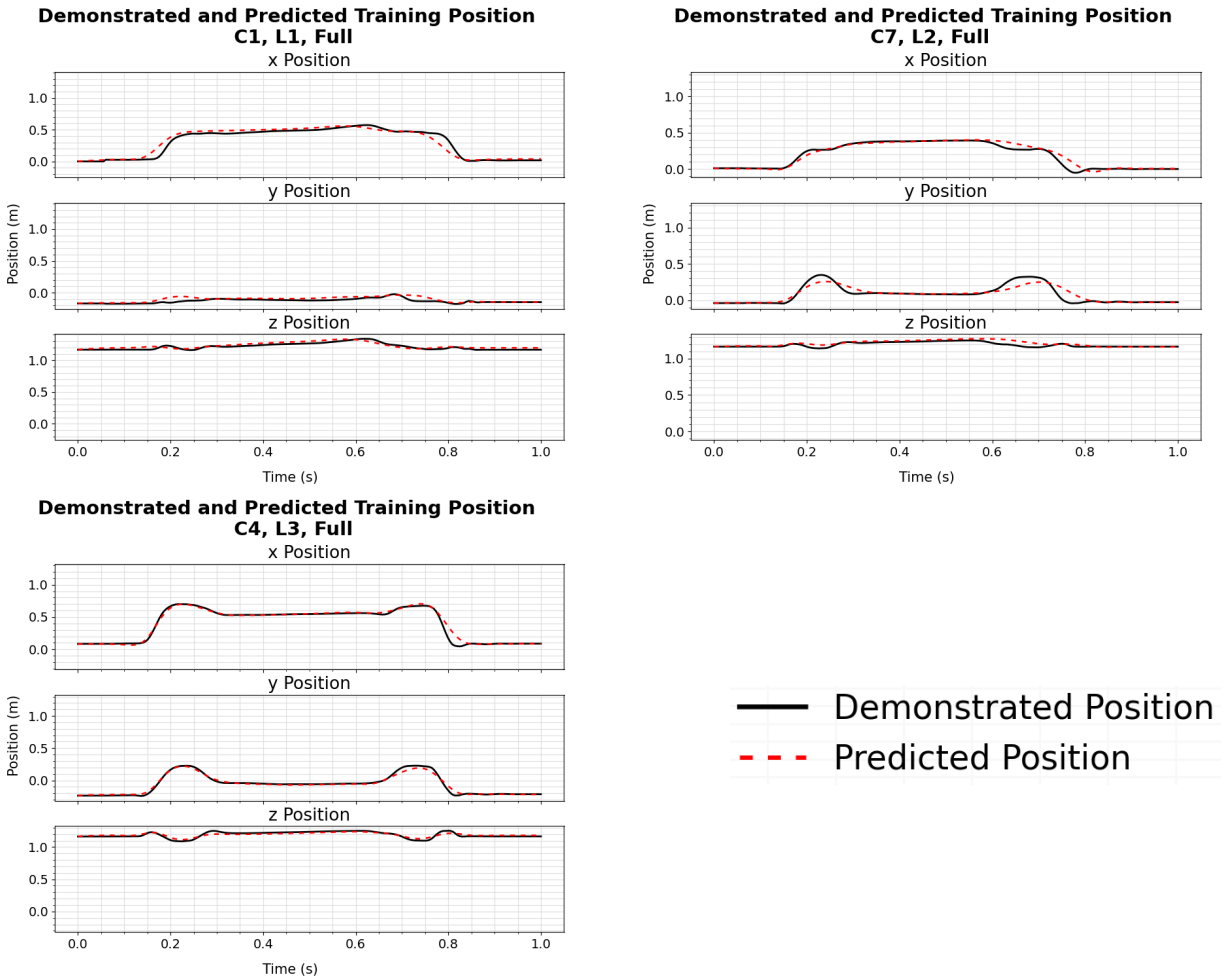


Figure 5.1: Trajectory components of the demonstrated position and the position predicted by the neural DMP for the model trained using the full demonstrations for samples from the training dataset.

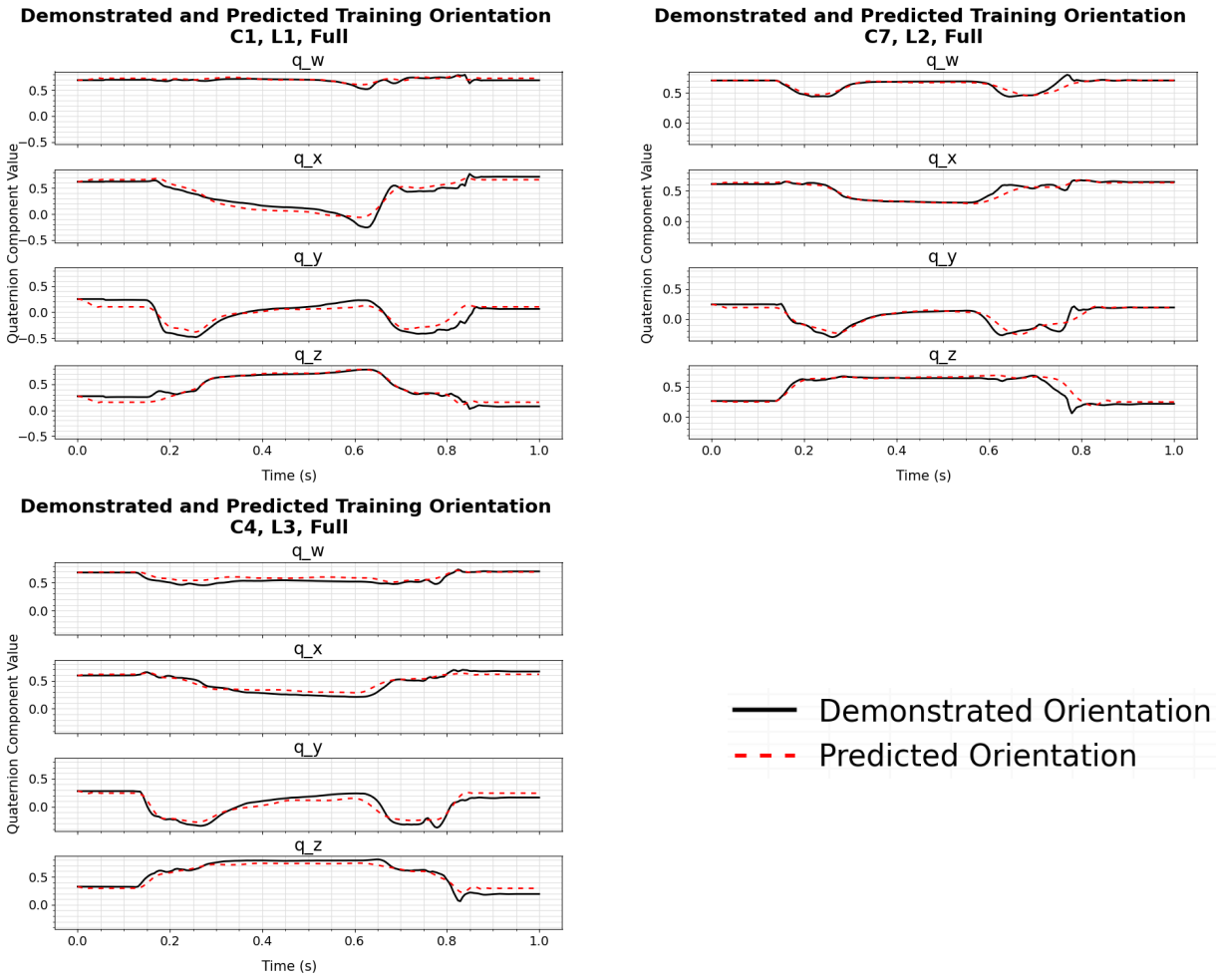


Figure 5.2: Trajectory components of the demonstrated orientation and the orientation predicted by the neural DMP for the model trained using the full demonstrations for samples from the training dataset.

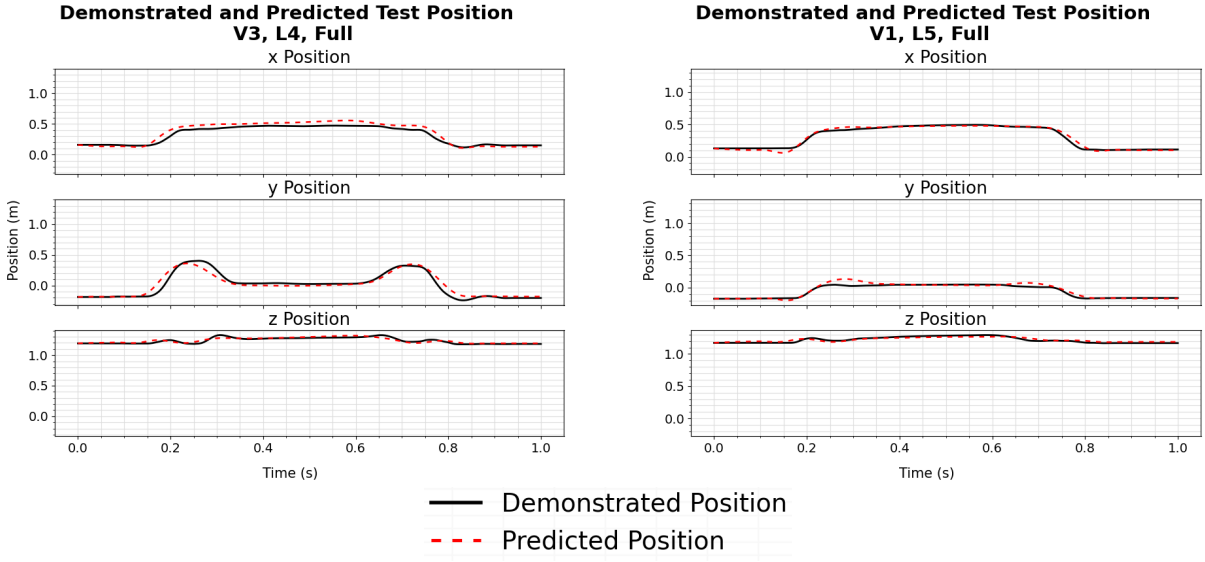


Figure 5.3: Trajectory components of the demonstrated position and the position predicted by the neural DMP for the model trained using the full demonstrations for samples from the test data set.Three simple graphs

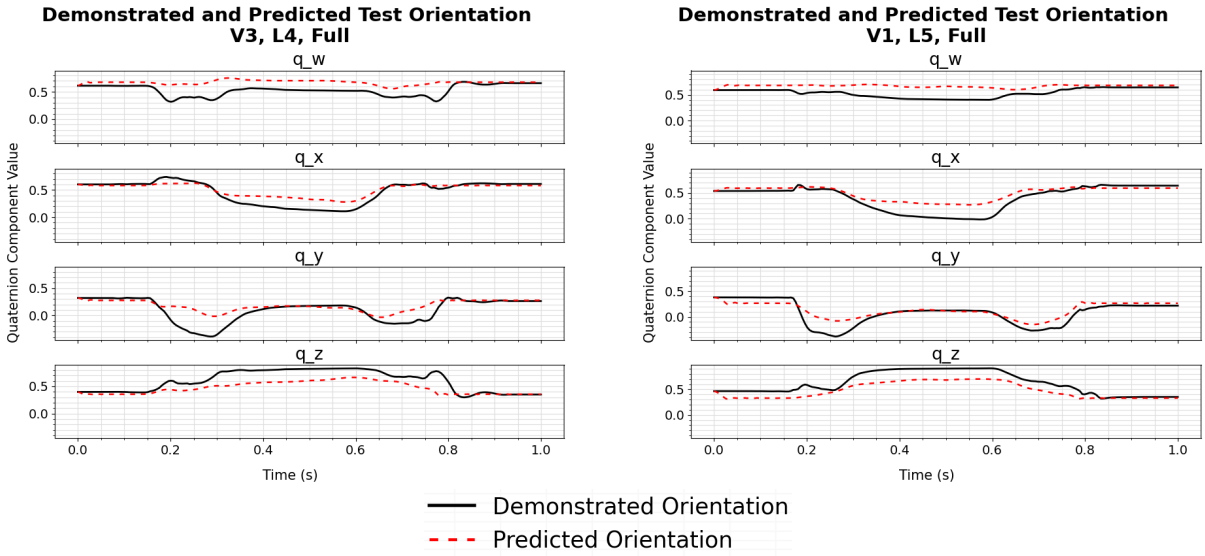


Figure 5.4: Trajectory components of the demonstrated orientation and the orientation predicted by the neural DMP for the model trained using the full demonstrations for samples from the test data set.

learned the commonalities across all the demonstrators, which could account for some of the error. Despite the increased error, examining these two plots shows the key movements stay aligned in time with the demonstrations. This shows that while the movements may not be exactly the same as the demonstrator, they maintain similar elements at similar times and maintain the structure of the underlying task. As such, this shows that the models have learned key features of the task, and the errors may be explained by learning the commonalities between demonstrators instead of replicating a single demonstrator.

5.3 Model Trained on Segmented Demonstrations

This section analyses the performance of the position and orientation models trained using the segmented demonstrations. Samples selected from the training dataset are analysed to visualize the models' ability to recall configurations. Next, the models' performance samples from the testing set that were not seen during training are shown. This allows for a comparison between trajectories for new container configurations and water levels to asses the ability to generalize to new inputs and generate trajectories similar to those of a demonstrator. The same configurations and water levels were selected for the segmented models as were selected for the models trained on the full demonstrations allowing for a comparison between the two methods. As such, three samples and configurations selected from the training set, and two configurations being selected from the testing set.

The position and orientation models trained for each segment were trained using the method described in section 3.4. The parameters for each position segment model are presented in table 5.4, and the parameters for each orientation segment model are presented in table 5.5. It can be seen that for both the position and orientation models, the first and third segments tend to have simpler models than than the second in terms of number of layers, number of nodes per layer and the number of basis functions. This is because they represent the more basic motions of reaching for the container and returning to the original manipulator position, while segment 1 represents a more complex motion that requires the actual pouring of the liquid. ReLU activation functions were used following each layer in each of the networks.

5.3.1 Training Set Performance

The trajectories and demonstrations from the same three configurations with the same water levels that were shown for the models generated from the full demonstrations are

	Position Models		
Parameter	**Seg. 1**	**Seg. 2**	**Seg. 3**
# layers	1	1	1
layer sizes	230	340	140
β	20	20	20
epochs	14	14	12
# basis functions	20	40	15

Table 5.4: Model parameters for the position neural DMPs trained on the segmented demonstrations for each segment.

	Orientation Models		
Parameter	**Seg. 1**	**Seg. 2**	**Seg. 3**
# layers	2	3	2
layer sizes	110, 110	200, 230, 200	170, 230
β	25	25	25
epochs	11	10	7
# basis functions	10	30	10

Table 5.5: Model parameters for the position neural DMPs trained on the segmented demonstrations for each segment.

displayed for the models trained using the segmented demonstrations. The Cartesian components of the position trajectories from the demonstrations, and the corresponding position trajectories generated from the neural DMP for the selected samples are shown in fig. 5.5. The corresponding quaternion orientation trajectories from the demonstrations and the predicted orientation trajectories from the neural DMP are shown in fig. 5.6. In both cases, the demonstrated trajectory components are shown by a solid black line, the predicted trajectory components are shown with a dashed red line and the locations where the demonstration segmentation occurred are shown using a vertical blue dashed line.

Figures 5.5 and 5.6 show that the neural DMPs trained using the segmented models are, for the most part, able to accurately reproduce previously seen trajectories. When comparing the third orientation trajectory (bottom left) from the segmented demonstration method in fig. 5.6 to the corresponding trajectory from the full demonstration method in fig. 5.1, it appears to have a larger error, particularly near the middle of the trajectory. This seems to contradict the average losses presented in section 5.1 are typically lower for the segmented models. However, this serves to demonstrate that while average losses may be lower for the segmented models, there may be individual trajectories that are better represented by the models trained using full demonstrations. Also, unlike the models that used the full demonstrations, the segmented models appear to have more consistent errors throughout the trajectory as opposed to increased error near the end of the demonstration.

5.3.2 Testing Set Performance

The same configurations and water levels that were used to demonstrate the test performance for the neural DMPs that were trained using the full demonstrations are shown to demonstrated the generalization capabilities of the segmented models. The Cartesian components of the position trajectories from the demonstrations, and the corresponding position trajectories generated from the neural DMP for the selected samples are shown in fig. 5.7. The corresponding quaternion orientation trajectories from the demonstrations and the predicted orientation trajectories from the neural DMP are shown in fig. 5.8. As with the training set, the demonstrated trajectory components are shown by a solid black line, the predicted trajectory components are shown with a dashed red line and the segmentation points are shown with dashed blue lines for both figures.

Figures 5.7 and 5.8 position and orientation generated by the segmented neural DMPs are similar to those of the demonstrator. However, just as with the models trained on the full demonstrations, there is an increased error, which is expected as a result of neural networks not generalizing perfectly. Similarly, this is also likely due to the models having

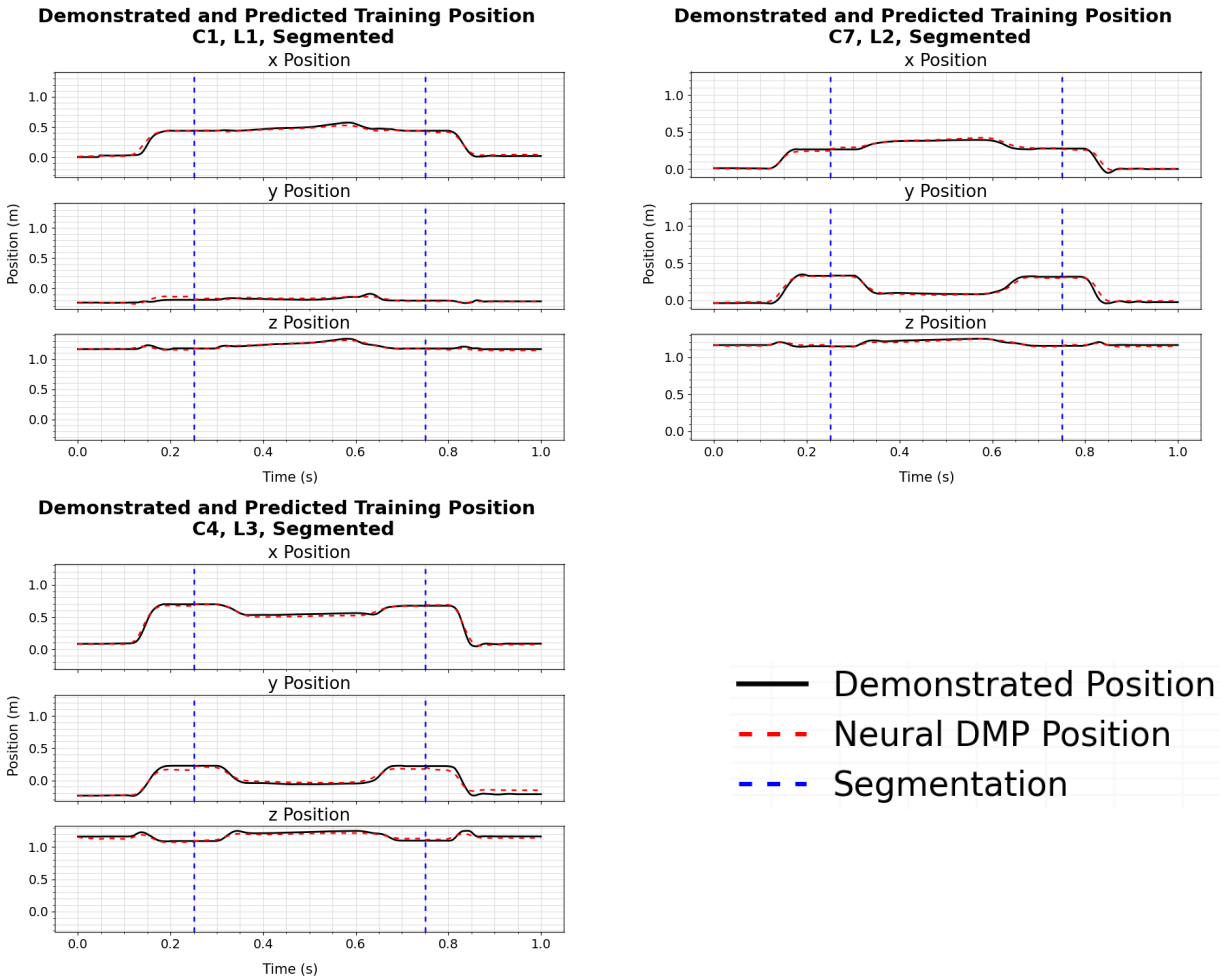


Figure 5.5: Trajectory components of the demonstrated position and the position predicted by the neural DMP for the model trained using the segmented demonstrations for samples from the training dataset.

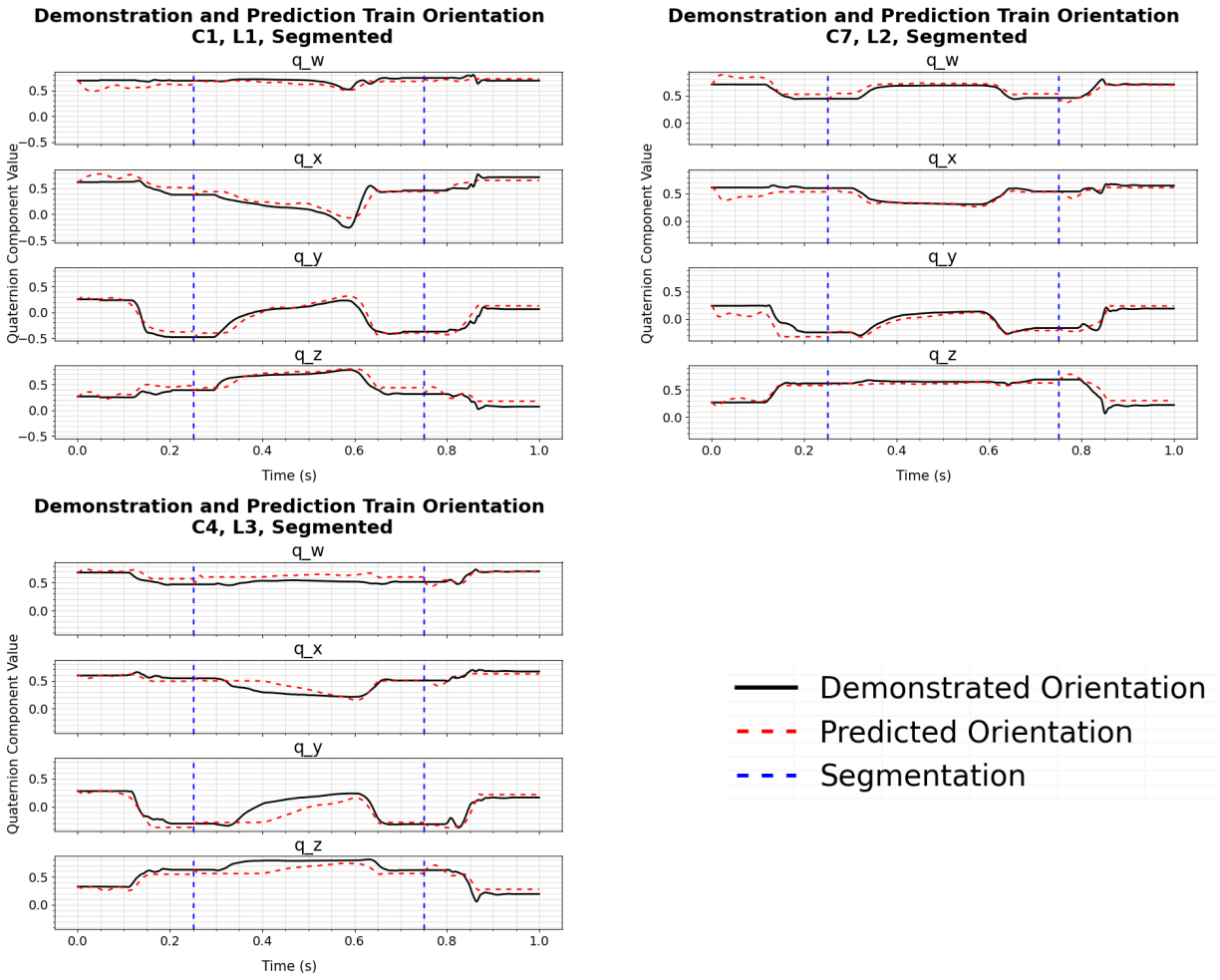


Figure 5.6: Trajectory components of the demonstrated orientation and the orientation predicted by the neural DMP for the model trained using the segmented demonstrations for samples from the training dataset.

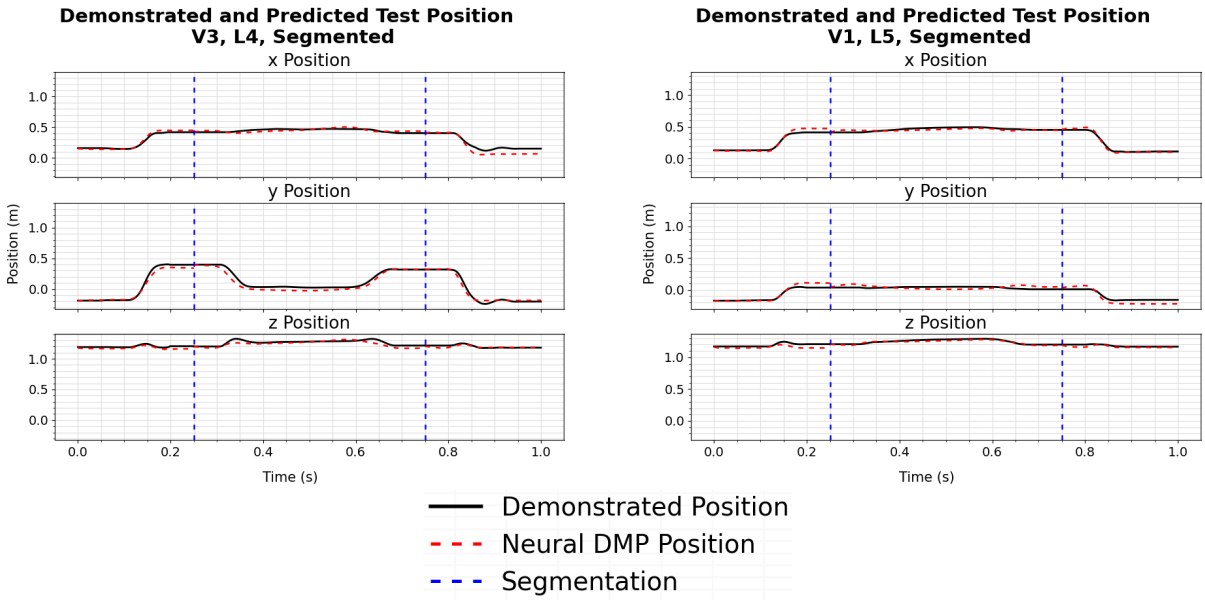


Figure 5.7: Trajectory components of the demonstrated position and the position predicted by the neural DMP for the model trained using the segmented demonstrations for samples from the test dataset.

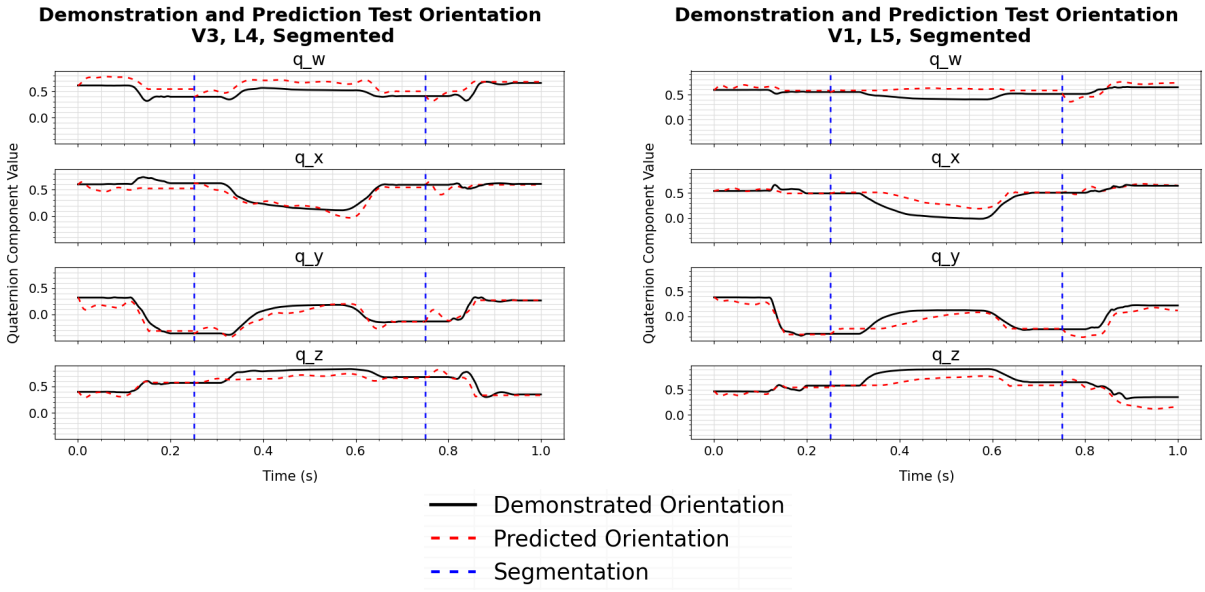


Figure 5.8: Trajectory components of the demonstrated orientation and the orientation predicted by the neural DMP for the model trained using the segmented demonstrations for samples from the test dataset.

learned from different demonstrators, thus incorporating elements from the training trajectories from various demonstrations. This means that it may not perfectly align with a single demonstrator when generalizing, which could account for some of the error. Despite the increased error, examining these two plots shows the key movements stay aligned in time with the demonstrations. This shows that while the movements may not be exactly the same as the demonstrator, they maintain similar elements at similar times. As with the models learned from full demonstrations, this shows that while the movements may not be exactly the same as the demonstrator, they maintain similar elements at similar times and maintain the structure of the underlying task. As such, this shows that the models have learned key features of the task, and the errors may be due to learning base explained by learning the commonalities between demonstrators instead of replicating a single demonstrator.

5.4 Discussion on the Model Training and Validation Results

This section analysis the results presented in this chapter. This includes a comparison of the sample trajectories to previous, comparable works, to determine if the errors achieved are reasonable. It also presents several potential tradeoffs between the two methods of training models.

Two works that present tasks in Cartesian space are presented by Saveriano et al. [44] for transferring sugar to a container, and by Saveriano et al. [43] to demonstrate generalized Cartesian DMPs, both of which learn from single demonstrations. The errors in the training sample plots appear similar to those resulting from the Cartesian DMPs in these works. However these do not incorporate neural networks, and only learn from a single demonstration, so it follows that only the ability to recall previously seen trajectories would be similar. For both the models that learned from full and the models that learned from segmented demonstrations, the error is higher in the testing set than for the training set. This is shown in both the average losses for the models presented in tables 5.1 and 5.2 as well as the plots shown in the previous sections. When compared to the work done by Zhang et al. [63], who integrate neural networks into DMPs, the results show a similar reduced performance when generalizing to new inputs.

The orientation results in [44] and the position and orientation results presented int [43] show some similar trends to the samples presented in the previous sections. In particular, they show increased error in later sections of the trajectories which is similar to those seen

in both the position and orientation in the training and testing samples from the models trained using full demonstrations. This can be seen in figures 5.1, 5.2, 5.3 and 5.4 for the training position, training orientation, testing position and testing orientation sample trajectory components respectively. However, this shift in the errors is not as evident in the training and testing samples trajectories for the models trained using segmented demonstrations shown in figures 5.5, 5.6, 5.7, 5.8. As such, this may be a result of the length of the trajectories learned. The reduced error later in the longer DMPs provides a potential tradeoff between the segmented and non segmented models. While the segmented models appear to have less error later in their trajectories, they require more effort and task knowledge to create usable segmented demonstrations.

Another aspect to consider between the models that learned from the full demonstrations and those that learned from the segmented demonstrations is the ability to combine the primitives in different orders. A relatively simple high level planner was used in this paper to achieve the objectives of analysing the functionality of the model learned from segmented demonstrations, but a more complex planner could allow for combining DMPs in different orders. Such planners have been explored, for example by Luo et al. [30] who even allow for retrying primitives if they are observed to have failed. In the case of this task, a more complex planner could be used to pick up and return the pouring container to different locations, and while this could theoretically be achieved using neural DMPs trained on single demonstrations, it would require modifications to the dataset collection and input structure.

Chapter 6

Testing on Robot Hardware

This chapter demonstrates the framework's functionality on a physical robotic platform. The models trained on full and segmented demonstrations are deployed on a 7DoF Franka Emika Panda robot manipulator with different sets of inputs. The controller and control architecture are further described in chapter 3.6. The trajectories are shown both through plots showing the robot's tracking of the desired end effector pose, and sequences of images showing the robot as it moves through the trajectories. Finally, a discussion on the performance of the framework during testing on the physical robot is presented.

As with the model testing and validation in simulation, the selected configurations and water levels correspond with those presented in section 4.1.

6.1 Testing Model Learned from Full Demonstrations

This section presents the results of the models trained on the full demonstrations tested on the physical robotic manipulator. The desired position generated from the neural DMP along with the actual robot end effector position are shown in fig 6.1, and the desired orientation generated from the orientation neural DMP along with the actual robot end effector orientation is shown in fig. 6.2. These show two trajectories with different starting points relative to the goal and different water levels. In both figures, the output generated from the neural DMP is shown with the solid black line, while the actual robot state is shown with the red dashed line.

Figures 6.1 and 6.2 demonstrate accurate tracking of the desired trajectory of the robot. This validates the use of the controller described in chapter 3.6, and it is sufficient for this

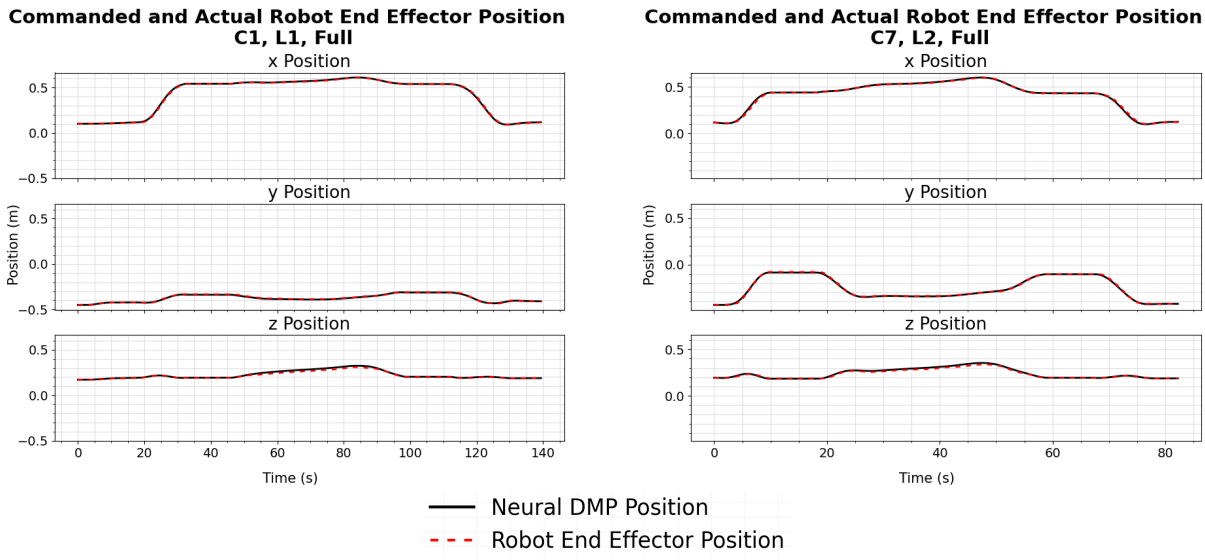


Figure 6.1: Desired and actual position of the robot end effector for the models trained using full demonstrations.

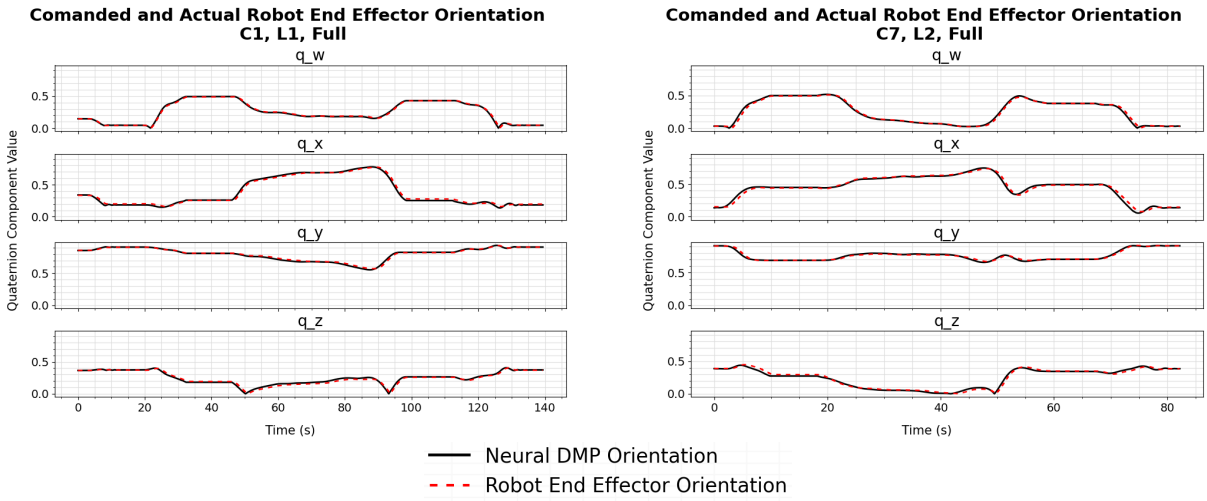


Figure 6.2: Desired and actual orientation of the robot end effector for the models trained using full demonstrations.

task There is also minimal noticeable increase in disturbance or error from the added weight of the pouring container when it is collected. These plots also demonstrate that the models are able to learn and perform a full task from a full demonstration with no required segmentations. This is further proven by the image sequences which are shown next.

Figure 6.3 shows a sequence of images of the robot performing the first tested trajectory using the neural DMPs trained on the full demonstrations with no augmentations. This sequence shows that the robot is able to successfully complete the task by grasping the water container, moving to the goal container, pouring the water without spilling, returning the water container to its original position, and returning the robot to its initial position. A second sequence of images corresponding to the second plotted configuration and water level is shown in Appendix C in fig. C.1.

6.2 Testing Model Learned from Segmented Demonstrations

This section presents the results of testing the models trained from the segmented demonstrations using chained DMPs, as described in 3.5. Figure 6.4 shows the commanded position generated from the neural DMP trained with the segmented demonstrations and the actual robot end effector position. Similarly, fig. 6.5 shows the commanded orientation generated from the neural DMP trained with the segmented demonstrations, and the actual robot end effector orientation. In both cases, the trajectory element generated by the DMP is represented by the black solid line, while the actual robot state is shown by the red dashed line.

These plots demonstrate that while it required more task knowledge to segment the demonstrations for training, it is able to complete the task. As with the full model, the images show that the robot is able to grasp the pouring container, pour the water without spilling, return the container and return to its original pose. A second sequence for the second configuration and water level is shown in Appendix C in fig. C.2.

6.3 Discussion on Testing the Framework on a Robotic System

Testing the neural DMPs that were trained on full demonstrations and segmented demonstrations were shown to be able to perform the desired tasks. This included pouring from

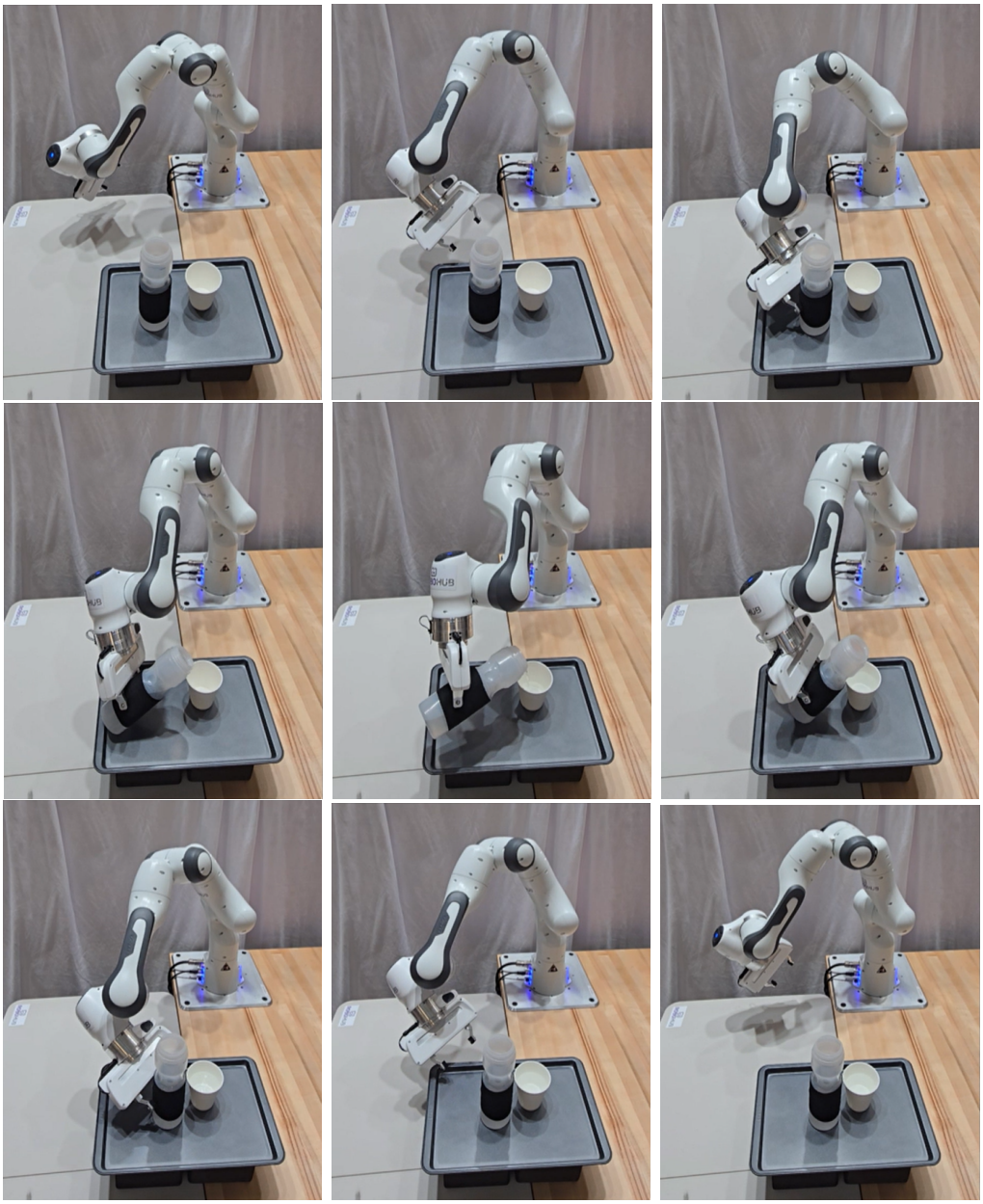


Figure 6.3: Image sequence showing the first trajectory and water level (C1 L1) being performed by the robot using the neural DMPs trained on the full demonstrations with no segmentation.

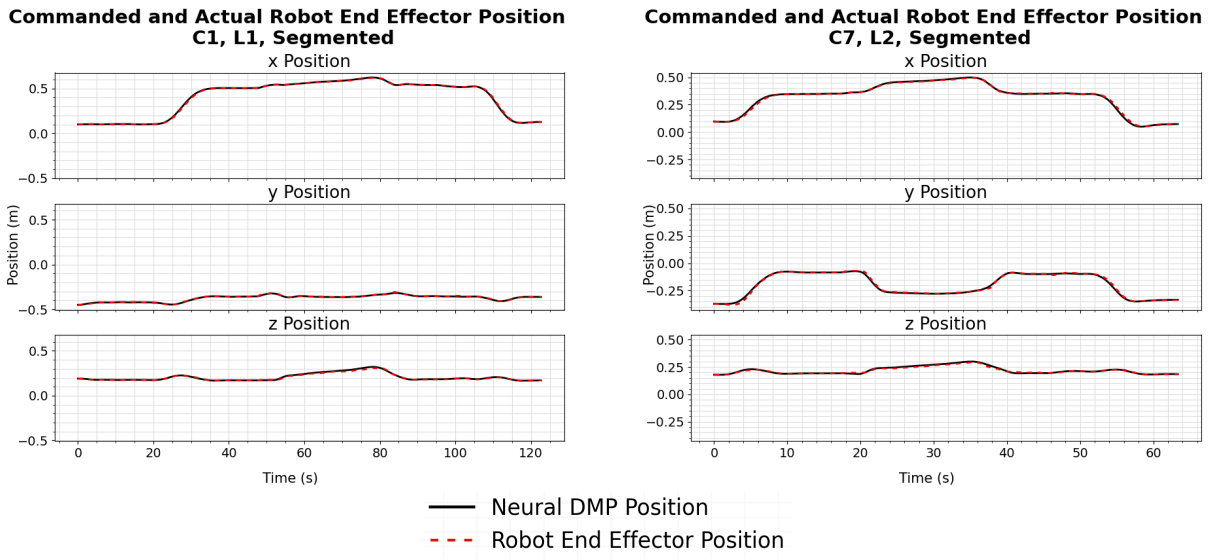


Figure 6.4: Desired and actual position of the robot end effector for the models trained using segmented demonstrations.

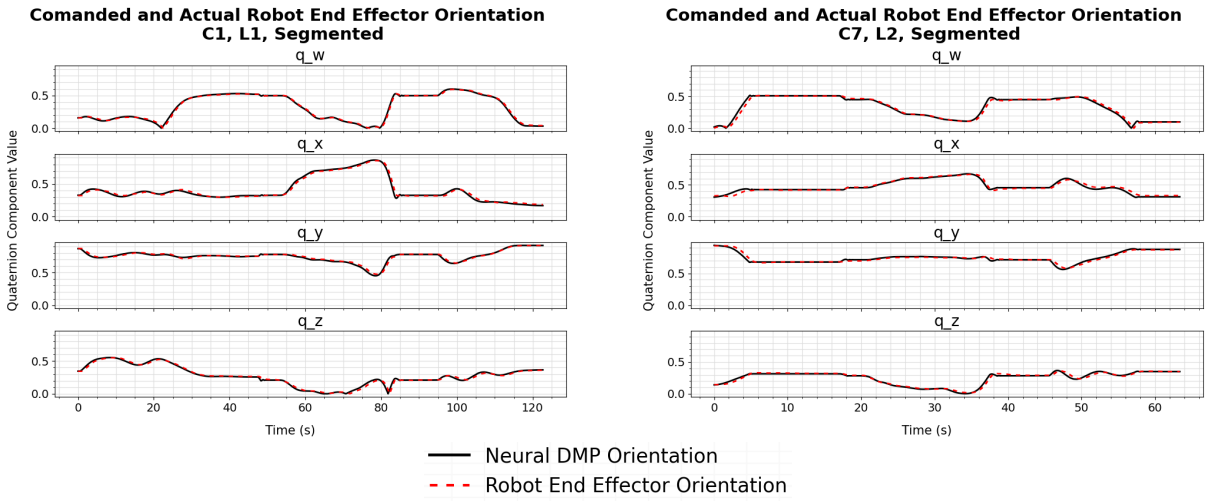


Figure 6.5: Desired and actual orientation of the robot end effector for the models trained using segmented demonstrations.

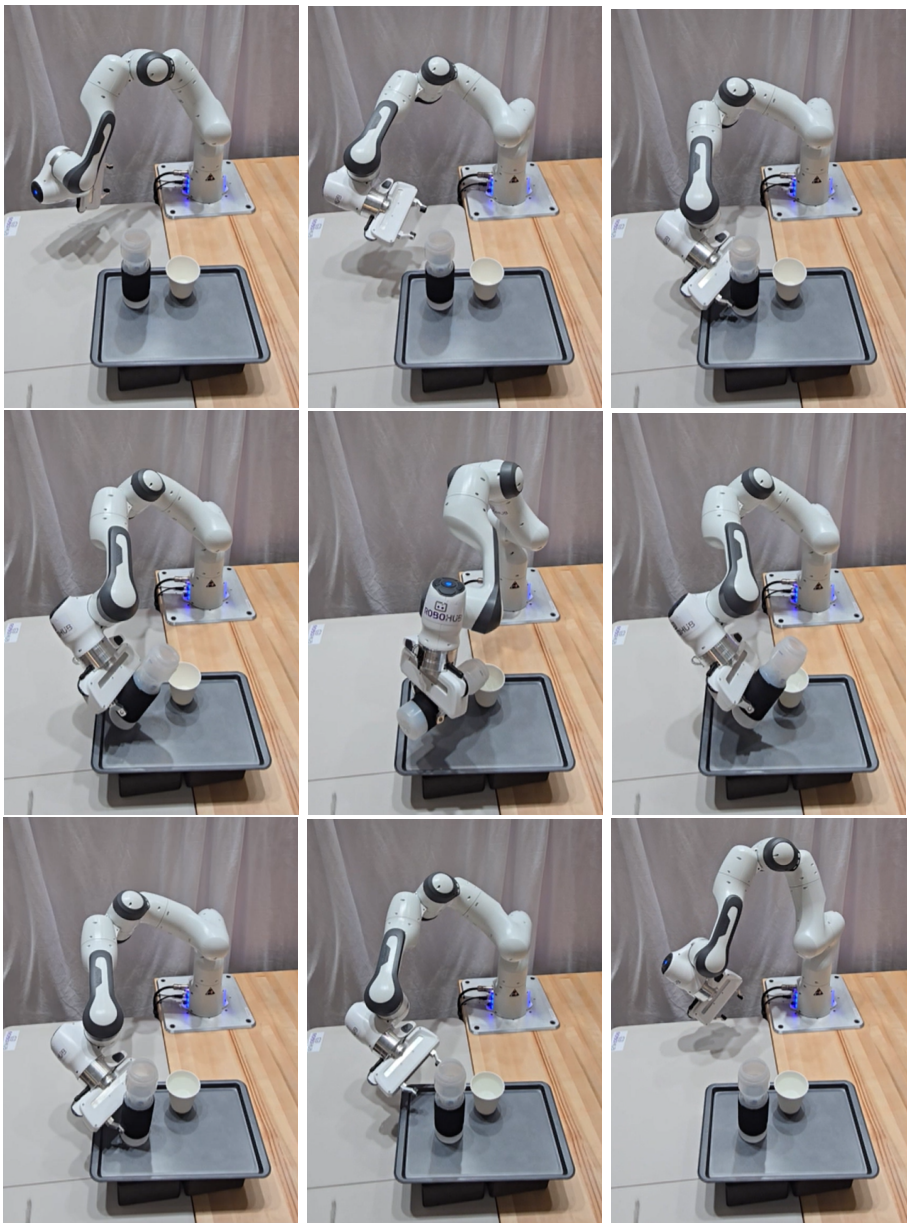


Figure 6.6: Image sequence showing the first trajectory and water level (C1, L1) being performed by the robot using the neural DMPs trained on the segmented demonstrations. The first row corresponds to the first segment (grasping), the second row corresponds to the second segment (pouring) and the third row corresponds to the third segment (returning).

containers located at different starting positions relative to the receiving container, and for different levels of water in the pouring container.

While implementing the testing on the system, the time scaling was adjusted to optimize the performance of the systems. The speed was reduced for to ensure that the water in the container did not spill while being moved. However, even thought the pouring motion is captured by the models, if the trajectory is performed too slowly the pouring was not successful. This most often resulted in spilling as the water would run down the side of the container instead of being properly poured. This occurred for both the models that were learned from full demonstrations, and the models learned from the segmented demonstrations. Investigating methods of inferring the time scaling parameter from the demonstrations themselves and incorporating them into the models could be the subject of future work.

During the experiments, several observations were made regarding the robot's performance and the current potential limitations of our framework. Firstly, selecting pouring container and goal container positions that kept the robot inside a feasible workspace for the robot was essential to successfully completing the task. This was due to the model not taking into account the kinematic and workspace constraints as it is intended as general framework that can be applied to a variety of tasks and robots. However, this resulted in some adjustments to the workspace such as raising the containers off the table, as can be seen in the image sequences. Additionally, as the framework did not integrate any real-time feedback, the pouring and goal containers needed to be precisely placed for the manipulator to successfully grasp, pour and release. Integration of workspace and robot constraints into the training process, and adding sensor feedback to the trajectory generation to improve robustness could be useful additions to this model and the subject of future work.

Chapter 7

Conclusions and Future Work

This section summarizes the contributions and results of this thesis, and presents the conclusions that can be drawn. It also describes some potential directions in which this research could be continued.

7.1 Conclusions

This thesis presents a framework for learning from demonstration that can reduce the effort and task specific knowledge required by learning from full demonstrations instead of segmented demonstrations, and by augmenting physically collected demonstrations. After discussing related works and knowledge gaps, the formulation of the framework using neural DMPs is shown, as well as the training methodology and robot control architecture. The process of collecting demonstrations and how they are used to create datasets is then described. The performance of the trained models is analysed through the average loss over the training and test datasets, as well as by visualizing specific demonstrations and the trajectories predicted for the same inputs. Finally, the framework was implemented on a physical robotic system to ensure it could perform the desired task.

The time and effort required for the collection of physical demonstrations reduced by the use of data augmentation. For this task, each set of demonstrations required approximately one hour. However, this was for a relatively simple task, and the time required per demonstration would vary depending of factors such as the length of the demonstration and the specific task. Physical demonstrations were best suited for this task as it could easily be performed by demonstrators, but would be difficult to model for synthetic datasets.

However, collecting sufficient samples to effectively train the neural networks would have been much more difficult if all the data had been collected from demonstrators. The data augmentation used in this thesis resulted in 49 times the number of physically collected samples, but this could be modified based on the number of demonstrations available and the number that are needed for the learning technique. Using data augmentations could increase the viability of tasks in which demonstrations are difficult to obtain, such as with highly skilled tasks like surgical procedures, or tasks in dangerous environments such as space exploration.

This framework was analysed using a pouring task, which could be segmented into sub-tasks. This allowed two sets of models to be tested: one that learned from the full demonstrations with no segmentation or annotation from task related knowledge, and one that utilized task specific knowledge to segment the demonstrations into sub-tasks before learning. Both methods were shown to be able to recall the training trajectories, and showed some ability to generalize to new inputs to generate trajectories similar to how a demonstrator would have. Both sets of models were effective in reproducing the task when implemented on a physical robotic system.

While both models were able to perform the task, they both came with different benefits and drawbacks. The model trained using the segmented demonstrations typically had lower losses on average over the training and testing sets, indicating that they were generally more accurate at following the demonstrator's trajectory. However, the models trained using the full demonstrations were still successful at completing the task on the physical system, indicating that they were able also able to effectively learn the skills. While it was not tested, the segmented DMPs provides the potential to change the order of the primitives for increased customization. This could potentially be implemented by training from full demonstrations, but would need to be considered in all steps of the design process, namely dataset collection and input generation.

A major drawback of the models trained using the segmented dataset was the time required to preprocess the demonstrations. In this task, the segmentation added approximately 1 week of additional work for development and tuning of the automated segmentation and validation on the resulting datasets. However, it is important to note that this will vary with many factors, such as the task, type of demonstration, sensor data used, number of segmentations and previous experience with the task. While this can be partially automated, it requires knowledge of the task, effort to supervise and verify that it has been done correctly, and it may be unintuitive to select optimal sub-tasks. With these considerations in mind, the presented framework provides a potential alternative which may be useful for some tasks where obtaining expert reviews may be expensive or not feasible on a large scale due to scarcity of people with sufficient expertise. This could include

complex tasks with demonstrations that lack clear sub-task divisions such as in surgical or maintenance tasks.

The thesis met the objectives set out in section 1.1. The presented framework was able to learn from full demonstrations which did not require any segmentation or annotation based on task knowledge, and was able to perform the learned task on a physical system. The physical demonstrations were augmented, which saved further time and effort in the process of training the models and to reduce the number of demonstrations that needed to be collected. Successfully utilizing data augmentation and using full demonstrations for tasks with multiple sub-tasks resulted in a framework that could expand the feasibility and accessibility of learning to demonstration for a wider variety of situations. This could include tasks where expert human oversight is expensive or largely unavailable due to substantial training and expertise requirements, such as in surgical tasks and complex maintenance tasks. It could also includes tasks in dangerous areas such as deep sea welding or space exploration for which an abundance of demonstrations cannot be collected.

7.2 Future Work

While the framework presented in this thesis provides evidence of the use of neural DMPs to learn more complex behaviours, further work could be done to improve the capabilities of the system. This section provides an overview of some potential future research directions for this area.

For demonstration purposes of this paper, we placed objects in known workspace locations. In the future, to target a real-world application, sensor feedback will be incorporated to determine real-time task parameters to improve the overall robustness of task executions. For neural neural DMPs this is commonly done through feature extractors using CNNs [36] or encoder-decoders [33] to analyse images, but other inputs such as force feedback [63] have also been demonstrated. This would increase the robustness of the framework for real-world applications, and remove the need for hand crafted features.

To fully prove the usefulness of the framework presented in this thesis, it should be tested on different tasks. This should include tasks with few available demonstrations, and that are difficult to segment. Possible tasks could include maintenance tasks for vehicles, medical procedures or tasks for sample collection in hazardous environments such as space.

As the models in this framework were developed independently of a specific robotic system, kinematic and workspace constraints for specific robots were not considered. This imposed limitations on the selection of task parameters when the framework was applied

to a specific robot. Imposing kinematic constraints with DMPs has been explored [51], and implementing such methods could improve the robustness of the framework on the physical robotic system. Future research could incorporate robot specific parameters and constraints into the training process, or the implementation of an additional control layer that could account for such constraints to make the framework more reliable on physical systems.

One of the potential benefits of using sequenced motion primitives is the ability to combine primitives in different orders. The increased generalization that this provides could be explored for learning from full demonstrations by increasing the variety of demonstrations, and potentially utilising different augmentation techniques [37] to create more varied demonstrations.

Learning from demonstration and dynamic motion primitives have been used to assist with reinforcement learning by narrowing the search space [5]. When using DMPs for reinforcement learning, learning from full demonstrations could be useful to reduce the time and effort spent on this intermediate step. In addition, combining DMPs suitable for more complex tasks could increase the complexity of tasks that reinforcement learning methods could learn.

References

[1] Fares J Abu-Dakka, Bojan Nemec, Jimmy A Jørgensen, Thiusius R Savarimuthu, Norbert Krüger, and Aleš Ude. Adaptation of manipulation skills in physical contact with the environment to reference force profiles. *Autonomous Robots*, 39:199–217, 2015.

[2] Akhil S Anand, Andreas Østvik, Esten Ingar Grøtli, Marialena Vagia, and Jan Tommy Gravdahl. Real-time temporal adaptation of dynamic movement primitives for moving targets. In *2021 20th International Conference on Advanced Robotics (ICAR)*, pages 261–268, 2021.

[3] Janis Arents and Modris Greitans. Smart industrial robot control trends, challenges and opportunities within manufacturing. *Applied Sciences*, 12:937, 2022.

[4] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. *Locally Weighted Learning*, pages 11–73. Springer Netherlands, Dordrecht, 1997.

[5] Shikhar Bahl, Mustafa Mukadam, Abhinav Gupta, and Deepak Pathak. Neural dynamic policies for end-to-end sensorimotor learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5058–5069. Curran Associates, Inc., 2020.

[6] Riccardo Caccavale, Matteo Saveriano, Alberto Finzi, and Dongheui Lee. Kinesthetic teaching and attentional supervision of structured tasks in human–robot interaction. *Autonomous Robots*, 43:1291–1307, 8 2019.

[7] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):286–298, 2007.

[8] Chuize Chen, Chenguang Yang, Chao Zeng, Ning Wang, and Zhijun Li. Robot learning from multiple demonstrations with dynamic movement primitive. In *2017 2nd International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 523–528, 2017.

[9] Nutan Chen, Justin Bayer, Sebastian Urban, and Patrick van der Smagt. Efficient movement representation by embedding dynamic movement primitives in deep autoencoders. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 434–440, 2015.

[10] Shuai Dong, Zhihua Yang, Weixi Zhang, and Kun Zou. Dynamic movement primitives based on positive and negative demonstrations. *International Journal of Advanced Robotic Systems*, 20:17298806231152997, 2023.

[11] Bin Fang, Shidong Jia, Di Guo, Muhua Xu, Shuhuan Wen, and Fuchun Sun. Survey of imitation learning for robotic manipulation. *International Journal of Intelligent Robotics and Applications*, 3:362–369, 12 2019.

[12] Martin L. Felis. Rbdl: an efficient rigid-body dynamics library using recursive algorithms. *Autonomous Robots*, pages 1–17, 2016.

[13] Andrej Gams, Bojan Nemec, Auke Jan Ijspeert, and Aleš Ude. Coupling movement primitives: Interaction with the environment and bimanual tasks. *IEEE Transactions on Robotics*, 30:816–830, 2014.

[14] Michele Ginesi, Daniele Meli, Andrea Calanca, Diego Dall'Alba, Nicola Sansonetto, and Paolo Fiorini. Dynamic movement primitives: Volumetric obstacle avoidance. In *2019 19th International Conference on Advanced Robotics (ICAR)*, pages 234–239, 2019.

[15] Michele Ginesi, Daniele Meli, Andrea Roberti, Nicola Sansonetto, and Paolo Fiorini. Dynamic movement primitives: Volumetric obstacle avoidance using dynamic potential functions. *Journal of Intelligent & Robotic Systems*, 101:1–20, 2021.

[16] Michele Ginesi, Nicola Sansonetto, and Paolo Fiorini. Overcoming some drawbacks of dynamic movement primitives. *Robotics and Autonomous Systems*, 144, 10 2021.

[17] Claudia González, J Ernesto Solanes, Adolfo Muñoz, Luis Gracia, Vicent Girbés-Juan, and Josep Tornero. Advanced teleoperation and control system for industrial robots based on augmented virtuality and haptic feedback. *Journal of Manufacturing Systems*, 59:283–298, 2021.

[18] Yingbai Hu, Mingyang Cui, Jianghua Duan, Wenjun Liu, Dianye Huang, Alois Knoll, and Guang Chen. Model predictive optimization for imitation learning from demonstrations. *Robotics and Autonomous Systems*, 163:104381, 2023.

[19] Rui Huang, Chenguang Yang, Fan Yang, and Zhijun Li. Robot hand learning from multiple demonstrations using dynamics motor primitives. In *2017 International Conference on Advanced Mechatronic Systems (ICAMechS)*, pages 1–6, 2017.

[20] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50, 4 2017.

[21] Ravi P Joshi, Nishanth Koganti, and Tomohiro Shibata. Robotic cloth manipulation for clothing assistance task using dynamic movement primitives. In *Proceedings of the 2017 3rd International Conference on Advances in Robotics*. Association for Computing Machinery, 2017.

[22] Robert Krug and Dimitar Dimitrov. Model predictive motion control based on generalized dynamical movement primitives. *Journal of Intelligent & Robotic Systems*, 77:17–35, 2015.

[23] Robert Krug and Dimitar Dimitrovz. Representing movement primitives as implicit dynamical systems learned from multiple demonstrations. In *2013 16th International Conference on Advanced Robotics (ICAR)*, pages 1–8, 2013.

[24] Jean-Francois Lafleche, Shane Saunderson, and Goldie Nejat. Robot cooperative behavior learning using single-shot learning from demonstration and parallel hidden markov models. *IEEE Robotics and Automation Letters*, 4(2):193–200, 2019.

[25] Clemente Lauretti, Francesca Cordella, and Loredana Zollo. A hybrid joint/cartesian dmp-based approach for obstacle avoidance of anthropomorphic assistive robots. *International Journal of Social Robotics*, 11:783–796, 2019.

[26] Junjun Li, Zhijun Li, Xinde Li, Ying Feng, Yingbai Hu, and Bugong Xu. Skill learning strategy based on dynamic motion primitives for human–robot cooperative manipulation. *IEEE Transactions on Cognitive and Developmental Systems*, 13(1):105–117, 2021.

[27] Zhijun Li, Ting Zhao, Fei Chen, Yingbai Hu, Chun Yi Su, and Toshio Fukuda. Reinforcement learning of manipulation and grasping using dynamical movement primitives for a humanoidlike mobile manipulator. *IEEE/ASME Transactions on Mechatronics*, 23:121–131, 2 2018.

[28] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

[29] YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1118–1125, 2018.

[30] Jianlan Luo, Charles Xu, Xinyang Geng, Gilbert Feng, Kuan Fang, Liam Tan, Stefan Schaal, and Sergey Levine. Multistage cable routing through hierarchical imitation learning. *IEEE Transactions on Robotics*, 40:1476–1491, 2024.

[31] Takamitsu Matsubara, Sang-Ho Hyon, and Jun Morimoto. Learning stylistic dynamic movement primitives from multiple demonstrations. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1277–1283, 2010.

[32] Ekrem Misimi, Alexander Olofsson, Aleksander Eilertsen, Elling Ruud Øye, and John Reidar Mathiassen. Robotic handling of compliant food objects by robust learning from demonstration. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6972–6979, 2018.

[33] Rok Pahič, Barry Ridge, Andrej Gams, Jun Morimoto, and Aleš Ude. Training of deep neural networks for the generation of dynamic movement primitives. *Neural Networks*, 127:121–131, 7 2020.

[34] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[35] Affan Pervez, Arslan Ali, Jee-Hwan Ryu, and Dongheui Lee. Novel learning from demonstration approach for repetitive teleoperation tasks. In *2017 IEEE World Haptics Conference (WHC)*, pages 60–65, 2017.

[36] Affan Pervez, Yuecheng Mao, and Dongheui Lee. Learning deep movement primitives using convolutional neural networks. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 191–197, 2017.

[37] Adrian Prados, Santiago Garrido, and Ramon Barber. Learning and generalization of task-parameterized skills through few human demonstrations. *Engineering Applications of Artificial Intelligence*, 133:108310, 2024.

[38] Peteris Racinskis, Janis Arents, and Modris Greitans. A motion capture and imitation learning based approach to robot control. *Applied Sciences*, 12(14):7186, 2022.

[39] Akshara Rai, Franziska Meier, Auke Ijspeert, and Stefan Schaal. Learning coupling terms for obstacle avoidance. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 512–518, 2014.

[40] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual review of control, robotics, and autonomous systems*, 3:297–330, 2020.

[41] Laurel D. Riek. Chapter 8 - robotics technology in mental health care. In David D. Luxton, editor, *Artificial Intelligence in Behavioral and Mental Health Care*, pages 185–203. Academic Press, San Diego, 2016.

[42] Jože M Rožanec and Bojan Nemec. Neural dynamic movement primitives – a survey. *arXiv:2208.01903*, 2022.

[43] Matteo Saveriano, Fares J Abu-Dakka, Aljaž Kramberger, and Luka Peternel. Dynamic movement primitives in robotics: A tutorial survey. *The International Journal of Robotics Research*, 42(13):1133–1184, September 2023.

[44] Matteo Saveriano, Felix Franzel, and Dongheui Lee. Merging position and orientation motion primitives. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7041–7047, 2019.

[45] Stefan Schaal. *Dynamic Movement Primitives -A Framework for Motor Control in Humans and Humanoid Robotics*, pages 261–280. Springer Tokyo, Tokyo, 2006.

[46] Stefan Schaal, Peyman Mohajerian, and Auke Ijspeert. Dynamics systems vs. optimal control — a unifying view. In Paul Cisek, Trevor Drew, and John F. Kalaska, editors, *Computational Neuroscience: Theoretical Insights into Brain Function*, volume 165 of *Progress in Brain Research*, pages 425–445. Elsevier, 2007.

[47] Martin Schepers, Matteo Giuberti, Giovanni Bellusci, et al. Xsens mvn: Consistent tracking of human motion using inertial sensing. *Xsens Technol*, 1(8):1–8, 2018.

[48] Weiyong Si, Ning Wang, and Chenguang Yang. Composite dynamic movement primitives based on neural networks for human–robot skill transfer. *Neural Computing and Applications*, 2021.

[49] Weiyong Si, Ning Wang, and Chenguang Yang. A review on manipulation skill acquisition through teleoperation-based learning from demonstration. *Cognitive Computation and Systems*, 3:1–16, 3 2021.

[50] Antonis Sidiropoulos and Zoe Doulgeri. Dynamic via-points and improved spatial generalization for online trajectory generation with dynamic movement primitives. *Journal of Intelligent & Robotic Systems*, 110, 01 2024.

[51] Antonis Sidiropoulos, Dimitrios Papageorgiou, and Zoe Doulgeri. A novel framework for generalizing dynamic movement primitives under kinematic constraints. *Autonomous Robots*, 47:37–50, 2023.

[52] Aljaž Kramberger Christoffer Sloth and Iñigo Iturrate. Towards easy setup of robotic assembly tasks. *Advanced Robotics*, 34:499–513, 2020.

[53] Sylvain, Dillmann Rüdiger, Schaal Stefan Billard Aude, and Calinon. *Robot Programming by Demonstration*, pages 1371–1394. Springer Berlin Heidelberg, 2008.

[54] Minija Tamosiunaite, Bojan Nemec, Aleš Ude, and Florentin Wörgötter. Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives. *Robotics and Autonomous Systems*, 59:910–922, 2011.

[55] S.K. Tso and K.P. Liu. Hidden markov model for intelligent extraction of robot trajectory command from demonstrated trajectories. In *Proceedings of the IEEE International Conference on Industrial Technology (ICIT'96)*, pages 294–298, 1996.

[56] Aleš Ude, Bojan Nemec, Tadej Petrić, and Jun Morimoto. Orientation in cartesian space dynamic movement primitives. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2997–3004, 2014.

[57] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[58] David Vogt, Simon Stepputtis, Steve Grehl, Bernhard Jung, and Heni Ben Amor. A system for learning continuous human-robot interactions from human-human demonstrations. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2882–2889, 2017.

[59] Fan Xie, Alexander Chowdhury, M Clara De Paolis Kaluza, Linfeng Zhao, Lawson Wong, and Rose Yu. Deep imitation learning for bimanual robotic manipulation. In H Larochelle, M Ranzato, R Hadsell, M F Balcan, and H Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2327–2337. Curran Associates, Inc., 2020.

[60] Fengpei Yuan, Elizabeth Klavon, Ziming Liu, Ruth Palan Lopez, and Xiaopeng Zhao. A systematic review of robotic rehabilitation for cognitive training. *Frontiers in Robotics and AI*, 8:605715, 2021.

[61] Martijn JA Zeestraten, Ioannis Havoutis, and Sylvain Calinon. Programming by demonstration for shared control with an application in teleoperation, 2018.

[62] Tianhao Zhang, Zoe McCarthy, Owen Jowl, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5628–5635. Institute of Electrical and Electronics Engineers Inc., 9 2018.

[63] Ying Zhang, Miao Li, and Chenguang Yang. Robot learning system based on dynamic movement primitives and neural network. *Neurocomputing*, 451:205–214, 2021.

[64] You Zhou, Jianfeng Gao, and Tamim Asfour. Learning via-point movement primitives with inter- and extrapolation capabilities. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4301–4308, 2019.

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

[66] Wang Ziheng, Isabella Reed, and Ann Majewicz Fey. Toward intuitive teleoperation in surgery: Human-centric evaluation of teleoperation algorithms for robotic needle steering. In *2018 IEEE International Conference on Robotics and Automation (ICRA) : May 21-25, 2018, Brisbane, Australia.*, 2018.

APPENDICES

Appendix A

Research Ethics Approval Letter

The following letter is the approval letter for the user study ethics clearance from the university of Waterloo Research Ethics Board. It approves the collection of demonstrations from human demonstrators.

UNIVERSITY OF WATERLOO

Notification of Ethics Clearance to Conduct Research with Human Participants

---

Principal Investigator: Yue Hu

Student investigator: Geoffrey Hanks

File #: 46187

Title: Learning from demonstration data collection

The Human Research Ethics Board is pleased to inform you this study has been reviewed and given ethics clearance.

Initial Approval Date: 04/02/24 (m/d/y)

University of Waterloo Research Ethics Boards are composed in accordance with, and carry out their functions and operate in a manner consistent with, the institution's guidelines for research with human participants, the Tri-Council Policy Statement for the Ethical Conduct for Research Involving Humans (TCPS2 2022), the Ontario Personal Health Information Protection Act (PHIPA), and all laws and regulations of the province of Ontario (as applicable). Additionally, CREB operates in a manner consistent with the International Conference for Harmonization of Technical Requirements for Pharmaceuticals for Human Use (ICH) Guidance E6(R2): Good Clinical Practice , the International Organization for Standardization of Good Clinical Practices (GCP) as set out by ISO 14155 - Clinical investigation of medical devices for human subjects, Part C, Division 5 of the Food and Drug Regulations, Part 4 of the Natural Health Products Regulations, Part 3 of the Medical Devices Regulations. Both Boards are registered with the U.S. Department of Health and Human Services under the Federal Wide Assurance, FWA00021410, and IRB registration number IRB00002419 (HREB) and IRB00007409 (CREB).

Expiry Date: 04/03/25 (m/d/y)

Multi-year research must be renewed at least once every 12 months unless a more frequent review has otherwise been specified. Studies will only be renewed if the renewal report is received and approved before the expiry date. Failure to submit renewal reports will result in the investigators being notified ethics clearance has been suspended and Research Finance being notified the ethics clearance is no longer valid.

Level of review: Delegated Review

Signed on behalf of the Human Research Ethics Board



Heather Dekker, Ethics Advisor, hdekker@uwaterloo.ca, 519-888-4567, ext. 41506

This above-named study is to be conducted in accordance with the submitted application and the most recently approved versions of all supporting materials.

Documents reviewed and received ethics clearance for use in the study and/or received for information:

file: SOP_Xsens_Imu_ore43485_v2021-06-03.docx

file: SOP_Xsens_Imu_ore43485_v2021-06-03.docx

file: e-mailScript_version1_20240130.pdf

file: AppreciationLetter_version1_20240131.pdf

file: OnlineConsentForm_version2_20240401.pdf

file: InformationLetter_version2_20240401.pdf

Approved Protocol Version 2 in Research Ethics System

This is an official document. Retain for your files.

You are responsible for obtaining any additional institutional approvals that might be required to complete this study.

Appendix B

Cross Validation Losses

This appendix contains the losses for each fold during training and testing during the training process. Table B.1 shows the losses for the position models, and table B.2 shows the losses for the orientation models.

	Position Models							
	Fold 1		Fold 2		Fold 3		Fold 4	
Model	Train	Val.	Train.	Val.	Train	Val.	Train	Val.
Full	0.00128	0.00348	0.00150	0.00197	0.00145	0.00133	0.00146	0.00277
Seg. 1	0.000945	0.00189	0.00126	0.00106	0.00113	0.000876	0.00112	0.00126
Seg. 2	0.000920	0.00201	0.000858	0.00196	0.000970	0.000766	0.000864	0.00147
Seg. 3	0.00132	0.00168	0.00135	0.00148	0.00151	0.00119	0.00114	0.00178

Table B.1: Average loss over each trajectory and dataset for each fold during training and validation for the position models.

	Orientation Models							
	Fold 1		Fold 2		Fold 3		Fold 4	
Model	Train	Val.	Train.	Val.	Train	Val.	Train	Val.
Full	0.377	0.380	0.359	0.584	0.545	0.353	0.379	0.518
Seg. 1	0.372	0.475	0.433	0.577	0.387	0.387	0.405	0.381
Seg. 2	0.459	0.517	0.417	0.605	0.419	0.419	0.467	0.606
Seg. 3	0.307	0.383	0.331	0.472	0.298	0.298	0.351	0.471

Table B.2: Average loss over each trajectory and dataset for each fold during training and validation for the orientation models.

Appendix C

Image Sequences for C7 L2

Here, a second configuration tested on the robot is shown for the full and segmented models. Figure C.1 shows the image sequence of the models trained using the full demonstrations, while fig. C.2 shows the image sequence for the models trained using the segmented demonstrations.

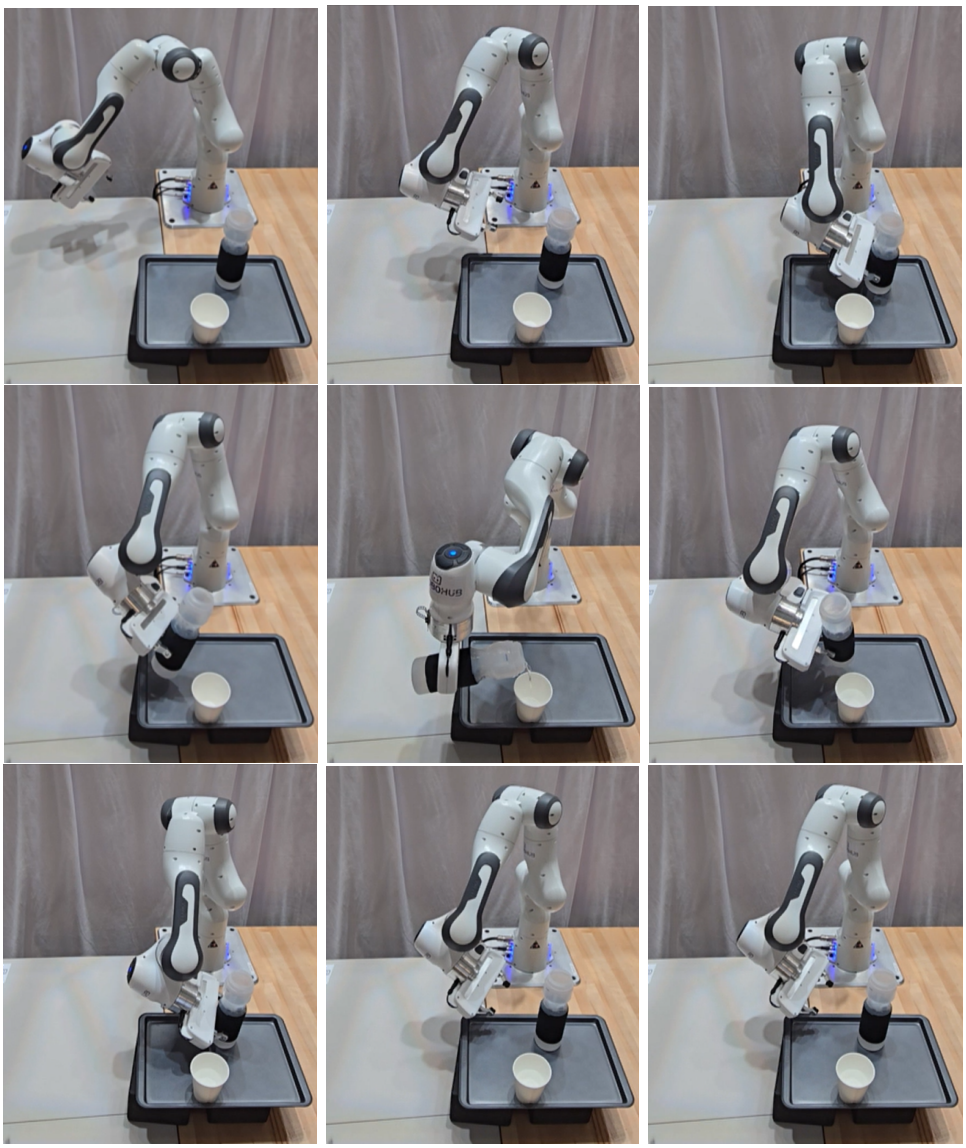


Figure C.1: Image sequence showing a trajectory and water level (C7, L2) being performed by the robot using the neural DMPs trained on the full demonstrations.

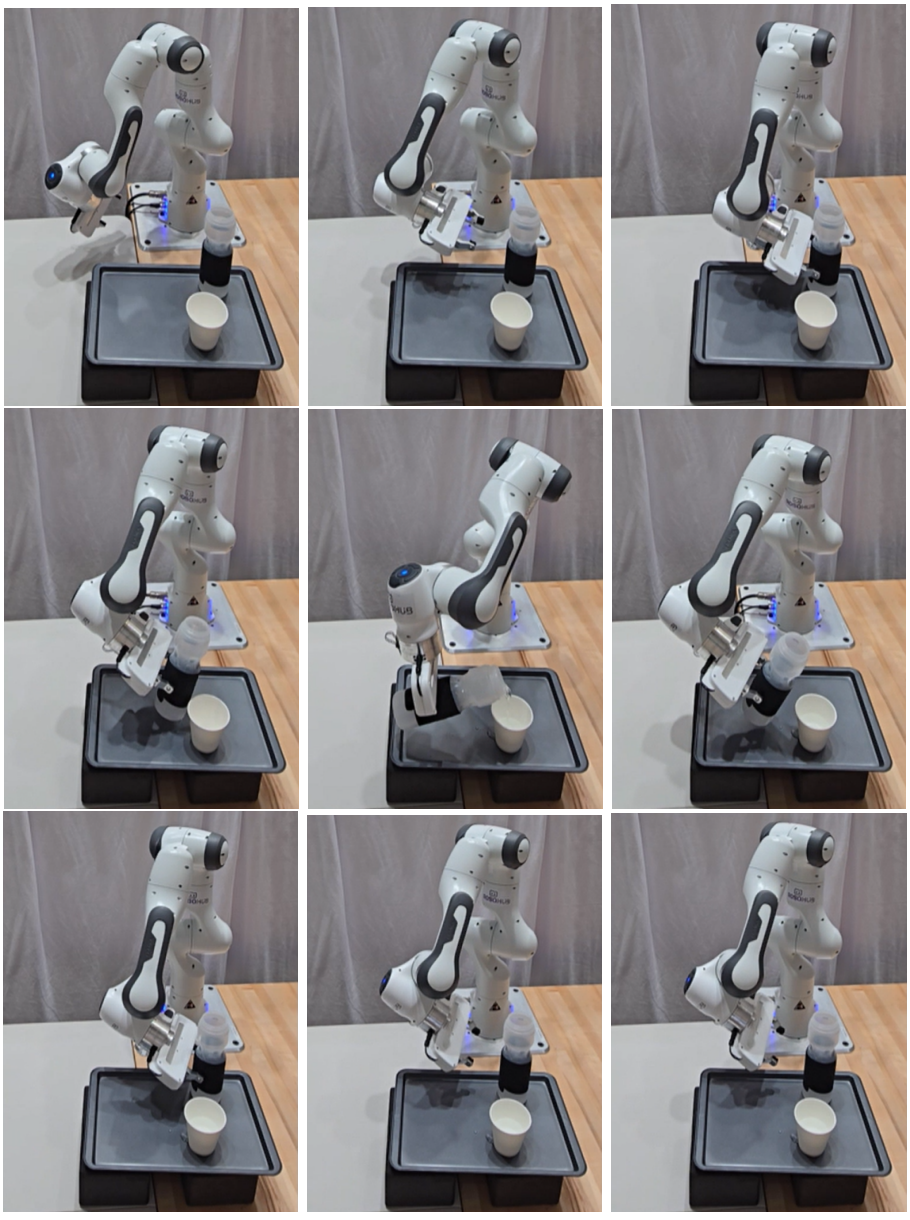


Figure C.2: Image sequence showing a trajectory and water level (C7, L2) being performed by the robot using the neural DMPs trained on the segmented demonstrations. The first row corresponds to the first segment (grasping), the second row corresponds to the second segment (pouring) and the third row corresponds to the third segment (returning).