

Smartwatch-Based Robot Control

by

Thibaut Roisin

A Thesis
Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2025 by the
Graduate Supervisory Committee

Heni Ben Amor, Chair
Nakul Gopalan
Ransalu Senanayake

ARIZONA STATE UNIVERSITY
December 2025

ABSTRACT

This thesis presents a teleoperation framework that uses a smartwatch-based motion capture with Dynamic Movement Primitives (DMPs) and In-Context Learning (ICL) for robot control. The system records human arm movements through a smartwatch connected to a smartphone, removing the need for traditional marker-based motion capture with special setup and equipment. Wrist positions are sent through a custom pipeline, filtered, and converted to end-effector coordinates through inverse kinematics using the Pinocchio library and run on a two-armed robot through Robot Operating System 2 (ROS2) controllers. DMPs encode the recorded motion into compact, mathematical representation with 105 weights (35 basis function per axis for x,y,z coordinates), allowing for smooth motion replay and customizable actions. The main goal is independent learning through ICL, where a large language model (LLM) creates new DMP weights by studying previous attempts and their results, weighted as rewards, creating a loop of improvements without traditional training methods.

Two experiments were conducted to test the system’s abilities and limits. While both of them shared a same problem, with the simulation tool, the first ball-bouncing experiment showed a key problem: when multiple hits at different times were encoded in a single DMP weight set, the LLM could not connect specific weights to specific results because of timing issues. This finding led to a redesigned experiment with a tetherball setup that provided natural timing consistency with a single weight set used for a single hit. It also solved some of the Gazebo physic issues. This experiment showed successful ICL-based weight improvements, with the model improving in creating better DMP parameters across five rounds despite the still unstable Gazebo rope physics. The thesis concludes that while the DMP framework smooths and replays human demonstration well, successful ICL use requires precise tasks with consistent

patterns. Future work should focus on moving to MuJoCo for better physics accuracy, automated testing systems allowing more than 100 improvement cycles and real world testing with camera-based ball tracking to remove simulation errors.

ACKNOWLEDGMENTS

I would like to express my gratitude to my thesis advisor, Professor Ben Amor, for his continuous support, guidance throughout these last months.

I am also so grateful to him for giving me the opportunity to work under his supervision in his Interactive Robotics Lab. This experience has been an incredible introduction to the field of Robotics. I have learned so much, both conceptually and practically, by working with various robotic systems and exploring their capabilities. I would also like to thank Professor Ben Amor for his valuable advice regarding my course selection, in particular in the choice of the course of EEE 515 Machine Vision and Pattern Rec. This guidance enabled me to complete an excellent internship at Indispense during last summer. I was able to apply the concepts acquired in this course to address the practical challenges faced by the company's robotic systems. It was rewarding to see that my solutions effectively resolved issues that their robots had been facing. The company was so satisfied by the results that they offered me a position upon completion of my Master's studies.

I must say that I have greatly enjoyed my Master's studies at Arizona State University. It was a very different experience from my Bachelor's studies in Computer Science in Belgium. This program has strengthened my passion for robotics and confirmed my aspiration to pursue my career in this field, ideally in the United States, where Robotics research and innovation are highly advanced. I would also like to thank the doctoral students in the Lab for their advice and assistance whenever needed.

Finally, I am also so grateful to my family for their support and constant encouragement in my decision to pursue my studies at Arizona State University.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	iii
	Page
LIST OF TABLES	vi
LIST OF FIGURES	vii
CHAPTER	
1 INTRODUCTION	1
2 LITERATURE REVIEW	4
Motion Capture	4
Similar work	5
Dynamic Movement Primitives	6
In-Context Learning	7
3 SYSTEM OVERVIEW	9
Teleoperation Hardware	9
Robot	9
ROS2	10
Gazebo Simulation	10
Inverse Kinematics	11
Pinocchio for Inverse Kinematics	11
Teleoperation Pipeline	12
Experiment Setup	13
First Experiment	13
Second and Final Experiment	16
4 METHODS	19
DMP	19
Mathematical Formulation	19

ICL.....	23
Data Collection and Processing.....	23
Reward function	24
Prompt Construction	26
5 RESULTS	32
DMPs Implementation and Trajectory Generation	32
Reward Function and Episode Assesment.....	33
First Experiment Setup	33
Final Experiment Setup	33
Analysis of Ball bouncing Failure Expirement	36
Analysis of Successful Tetherball Experiment	39
Simulation Environment and Ball Physics	45
6 DISCUSSION	46
DMP Framework Validation	46
First Experiment Setup	46
Final Experiment Setup	46
Limitation of In-Context Learning For the First Experiment	47
Tetherball experiment Conclusion	47
Conclusion	48
REFERENCES	50

LIST OF TABLES

Table	Page
5.1 Top 3 episodes by reward.	37

LIST OF FIGURES

Figure		Page
3.1	Full Teleoperation Pipeline	13
3.2	Gazebo simulation of the	14
3.3	Gazebo simulation when the physics fail	14
3.4	Gazebo simulation of the Sphere/box	15
3.5	Gazebo simulation of force field	16
3.6	Gazebo Tether Ball Setup	17
3.7	Force Applied to the ball	18
3.8	Hit-Ready Position	18
4.1	Example of the checkpoints as quadrants	26
4.2	Prompt Example from ProPS paper	27
4.3	Good and Bad Episodes Prompt	30
5.1	Wrist Positions (3D points)	33
5.2	DMP rollout trajectory	33
5.3	Basic episode reward plot	34
5.4	Good episode reward plot	35
5.5	Bad episode reward plot	36
5.6	Weight comparison with generated weights	38
5.7	Prompted Examples	40
5.8	Good episode reward plot	41
5.9	Third Iteration plot	43
5.10	Fifth Iteration plot	44

Chapter 1

INTRODUCTION

Teleoperation has existed since the early days of robotics. In the 19th century, it was primarily use for military purpose but is now mainly used to control robots from a distance. In this work, the objective will be to mimic our human arm movements onto a bimanual, or humanoid robot.

With the recent progress in humanoid robotics, such as many Chinese companies, including the Unitree G1 robot, that took social media with all the uses possible, and Boston Dynamics, alongside Tesla that are known in the American robotic market. It is important to start researches on that new technology to not fall out of the technology race.

This thesis tries to overcome the main problem with available teleoperation devices, which are the technology and wearable needed to accurately record movements. Classical systems use one or multiple cameras, marker based motion capture or even full inertia suit. With the progress of AI and machine vision, motion capture requires less set up and is more efficient. However, there is still the set up problem. We introduce a way to be able to capture outside of a lab, using a smartwatch coupled with a phone.

We introduce a novel way to be able to capture outside of lab settings, using a SmartWatch coupled to a phone as the sole capturing tools. The main idea is to only use a smartwatch worn on either wrist, and a phone carried in a pants pocket as sensing sources. This set up doesn't need anything to be done as people mainly carry their phones in their pocket, making it an accessible way of tracking whenever the user might want it. The smartwatch provides continuous measurements that are then

used by the phone, as a coarse body reference in order to calculate and approximate trajectories and postures. The 3 big measurements being given by technology are the wrist position and quaternion, elbow position and quaternion, the shoulder position and quaternion and also the hips quaternion. As the technology is still new, some accuracy is lost, mainly due to the approximation and lack of accurate data taken from the watch, the tradeoff of not needing any camera or suit could be the start of a lot of new technologies.

In this thesis, the simulation will be made on a bimanual robot, with 6 Degree of Freedom. The shoulders having only pitch and yaw, a perfect teleoperation of the elbow is not feasible, which means that the wrist will be the main link to be used as the position constraint. With a better robot, the full arm teleoperation is possible. Additionally, a second watch will be implemented to have the simulation of a full bust.

The controle stack is built on ROS2 for middleware and uses Pinocchio to calculate the inverse kinematics to map task-space commands to joint motions. Constraints are made for position-only tracking of the wrist with collision checks. Those details are not the subject of study but they are important steps in order for the whole study to be tested on.

While the sensing is the baseline of the work, the thesis focuses on making captured motions reusable and adaptable. Some imitation learning will be done using the teleoperation. The purpose of it is to be able to train certain movements on the robot arms, that it will be able to replicate and also innovate, later. The learning will be made by coupling Dynamic Motor Primitive, well known in the robotic and machine learning field, and In Context learning, a new technology that took off since the advancement with Generative AI and the newest models.

- Dynamic Motor Primitives (DMP) gives strong and adequate equations in order

to go over a set of points and encode the shape of a trajectory. Those equations can then be used to change the trajectory at will, using the weights, for a set goal. It will make the movements, recorded from the robot, into a set of equations that will be able to modify in order to have the arm to move in another direction using the same motion that a human arm did. Those equation and weights will be used for the In Context Learning.

- In Context Learning is a new technology that uses the reasoning of Generative AIs to get personalized results, by giving a set of examples and the results of those examples as prompt. The AI will then be able to give a correct answer, following the logic behind the tests, when a new situation is being fed to the AI. Its strengths are the speed of training, which can be done in hours, compared to other training methods that could take days or months. Another strength is that it only uses temporary memory as, it only uses it when interacting with the AI, making it easy to change training or actions without having to model a whole database around it. It doesn't require much training, however, human feedback is good in order to guide the AI while it gives answers that could be wrong, until its logic becomes perfectly connected to the problem.

Chapter 2

LITERATURE REVIEW

Motion Capture

Optical motion capture is the most prevalent in the business world, however it comes with its own challenges. It requires high-quality cameras and careful calibration. Early work by Windolf et al. (2008) made a study on how hardware and parameters affect directly accuracy and precision in motion capture. They mostly focused on parameters, such as : camera setup, calibration volume and marker size, with attributes like marker roundness, reflection capacity, optical contrast and also lightning condition. All those parameters and attributes had to be optimized in order to have the most accurate results. In short, the measurement fidelity depends not only on the hardware and the algorithms but also on the optics and camera geometry themselves. Given our goal of fast and everyday teleoperation without dedicated capture spaces, these parameters are a constraint, which is why we pursue an easy-to-wear sensing approach. [14]

A more recent study by Schröder et al. (2022) showed that the best results come from stereo/multi-camera systems with 2d markers. Report below 20 microns in all directions for measured errors. Between the 2 papers, the setup is still a major part of the accuracy of the teleoperation. [12]

Both papers tested OptiTrack, which is a leader in precise motion capture, exemplifies the calibration-sensitive workflow. Cameras must be synchronized and calibrated beforehand to reconstruct a 3D capture using triangulation. The documentation emphasizes wandng, ground-plane refinement, and continuous/periodic recal-

ibration to maintain the accuracy over time. [1]

Vicon systems are also leading in terms of optical motion capture. To get the best results, Vicon recommends regular, careful calibration during the use of the system. They also provide standard procedures for doing it correctly [2]. However, even low-cost VICON setups can still be accurate if calibrated well. [5]

The takeaway for the thesis. Optical systems set the accuracy bar but it requires precise calibration and multi-camera infrastructure, which constrains everyday use. This motivates the search for a new way for the sensing, such as a wearable-only way that requires minimum setup and versatility. Accepting a small accuracy trade-off in exchange for minimal camera setup, no markers, close to no preparation while still giving sufficient value for wrist-position teleoperation.

Similar work

Sumilar research has been made on it. IMUPoser (CHI 2023) tries to tackle the problem stated for the motion capture and teleoperation problems, however they are taking a different approach. Their idea is to be able to do full body capture using as many wearable they can think of. In their paper, they are mainly talking about using a phone, a smart-watch and earbuds, estimating a full body pose using the device's IMU. They compared their result against Vicon for reference. As they stated, their design introduce a lower fidelity accuracy, reporting that they can have up to 12.08 cm velocity error. However, due to their purpose, being more of a full body motion capture, than for teleoperation, the error is not that dramatic. It is where our work diverge and use the technology, requiring precise data, which our wearmocap has.

Dynamic Movement Primitives

Dynamic Movement Primitives (DMPs) were first introduced by Ijspeert et al. (2002) as a way to represent and learn motor skills from demonstration [8]. The main idea behind DMPs is to encode movements as a set of differential equations that act like a spring-damper system. This means the system naturally pulls toward a goal position while smoothly slowing down, which makes the movement stable and predictable. Unlike storing raw trajectory points, which take a lot of data space, DMPs capture the shape and timing of a motion using a small set of parameters called weights. This makes them much more compact and flexible than traditional methods.

The key advantage of DMPs is their ability to change movements easily without needing to record new demonstrations. Schaal et al. (2003) showed that DMPs can adjust the speed of a movement, change where it starts and ends, and even modify the path shape, all by changing some simple parameters [11]. This flexibility makes DMPs very popular in robotics because a robot can learn one movement and then adapt it to different situations. For example, if a robot learns to reach for an object at one location, it can use the same learned motion to reach for objects at different locations, by only changing a few parameters.

Paraschos et al. (2013) further developed DMPs by showing how they can be used with probabilistic models [10]. Their work demonstrated that DMPs can represent not just one movement, but a distribution of possible movements. This is useful when there is variation in demonstrations, which is not rare when humans perform the same task multiple times. In our work, we collect multiple demonstrations of hitting a ball, and each one is slightly different due to human variability. DMPs help us capture the common structure while handling these natural variations.

In-Context Learning

In-Context Learning (ICL) is a recent capability of large language models where the model can perform new tasks just by seeing a few examples in the prompt, without any traditional training or fine-tuning [4]. Brown et al. (2020) introduced this concept with GPT-3, showing that language models can solve math problems, translate languages, and answer questions simply by providing multiple examples in the input. This was not expected because traditionally, machine learning models need to be explicitly trained on thousands of examples with gradient updates. ICL works differently by allowing the model to learn patterns from just the examples shown in the prompt.

The key advantage of ICL for robotics is speed and flexibility. Traditional reinforcement learning methods might need hundreds or thousands of robot trials and take days or weeks to learn a new task. With ICL, a robot could potentially learn from just 10 to 20 demonstrations in a matter of minutes. Brohan et al. (2023) explored using large language models for robot planning and showed that models can understand robot tasks from just text descriptions and examples [3]. However, their work focused mainly on high-level planning rather than low-level motion generation, which is what our project addresses.

More recently, researchers have started applying ICL to generate robot motion parameters. Huang et al. (2024) introduced a method where language models generate motion primitives for manipulation tasks by learning from successful and failed demonstrations [6]. They showed that by providing the model with examples labeled as success or failure along with reward scores, the model could identify patterns and generate parameters for new situations. This approach inspired our work, where we give the language model multiple DMP weights along with the outcomes (number of

successful hits, ball height, drift) and ask it to generate new weights.

Wake et al. (2023) also explored using GPT-4 for robot task planning and found that, while language models are good at high-level reasoning, they need a clear structure in the information they receive [13]. When the input is too complex or has hidden structure, the model performance drops significantly. This matches our experience, later in the paper, where episode-level DMPs bundled too much temporal information together, making it hard for the language model to learn the relationship between weights and outcomes. Their work emphasizes the importance of breaking down complex tasks into simpler, easier pieces for the language model.

Chapter 3

SYSTEM OVERVIEW

Teleoperation Hardware

The hardware for the smartwatch and the phone are both Samsungs running on Android. Android is used for it is easiness to modify and run apps locally, where apple could make it more difficult. Insert the model of the watch, newer models always mean better and more accurate data. It uses complex Neural Network to estimate the position of the wrist and the elbow. The NN was made with the help of an Optitrack setup with a full combination of markers while the smartwatch was collecting data. Making it possible to match the watch positions with what the Cameras were seeing.

Robot

The robot used for the testing is a Robotis bimanual arm. The robot has 2 arms with 6 motors each, which results in 6 Degree Of Freedom. 2 in the shoulder, one in the elbow and 3 in the Wrist. The 2 DOF in the shoulder means that the robot will never be able to fully reach every state that a human arm can do. Most robot arms have 7 degrees of freedom. With another robot, different situations could be simulated, such as having a full elbow reach. The motors that are at each joint are motors with only a gain, where more recent motors have a PD controller or PID gains. This will result in having less smooth motions due to not being able to change the speed during an action.

ROS2

The robot will be controlled by ROS2 Jazzy, which acts as the main communication framework for the whole system.

Ros2 is used to connect all the part of the system. It starts with getting the joint states and information about the robot, to treat the values and use inverse kinematics later. And then sends the commands to the robot to make it move, using FollowJointTrajectory as an action and with other controllers that Ros2 gives us. The same code with ros can be used on the actual robot and on the simulation, which makes the shift from simulation to physical robot extremely easy and fast.

Gazebo Simulation

To simulate the robot and the environment, this work uses Gazebo because it connects easily with ROS 2. The same robot model, controllers and topics are used in simulation and on the real robot, which makes it simple to move from one to the other. However, this choice also has limits.

Another common simulation engine is MuJoCo, which is known for more accurate and stable physics, especially for contacts such as a ball hitting a racket. Later in this work, it is shown that the physics in Gazebo are not always reliable enough for certain tasks.

Because teleoperation also depends on accurate motion and timing, these limitations can affect the results.

For this project, Gazebo is kept for it is easy ROS 2 integration, but a future step should be to move the setup to MuJoCo (or a similar engine) to get more realistic physics for detailed contact tasks. A recent review of physics engines for RL also reports that MuJoCo is one of the most reliable engines in terms of physics

quality, while Gazebo is often chosen for it is ROS integration rather than it is contact accuracy [9].

Inverse Kinematics

Rather than controlling the robot in joint space, this work uses Cartesian space control by specifying end effector position for the wrist. Inverse Kinematics solves for the joint angles required to reach that position.

The IK solver operates iteratively. Starting with the current joint configuration and a certain target position, it computes the minimum joint displacement, over a few iterations, needed to reach the goal. This is done by using kinematics constraints such as the joints limits, that we get with the urdf, and singularity avoidance. If possible, the solver returns a joint configuration within the reachable workspace. For teleoperation, a fast IK computation is important to maintain responsiveness. With the delay from the watch to the phone, to the computer calculating the position, to the ros2 script receiving those position, a fast IK is critical.

Pinocchio for Inverse Kinematics

This work uses Pinocchio as the IK solver. Moveit is implemented into the project, to go with gazebo, however, the few inverse kinematics algorithms that go with moveit aren't precise enough, such as KDL, TracIK, etc... Pinocchio is a fast library that lets us just use the robot's URDF description and everything will be ready to go. The forward kinematics, jacobian and everything is part of the library and just need to call them one after the other to get a solid inverse kinematics result. As always, with teleoperation, fast results mean better results.

Teleoperation Pipeline

The teleoperation connects the smartwatch motion to the robot end-effector in real time.

The smartwatch and phone software sends an estimated wrist position at about 40 Hz (roughly one new target every 25 ms). This raw signal is too noisy to use directly, so a low-pass filter is applied. The filter removes fast spikes and jitter from small sensor errors, while keeping the main motion. After filtering, the wrist trajectory is smooth enough to be used as a position target for the robot.

On the ROS 2 side, a short horizon of recent filtered positions is stored. Instead of reacting to a single sample, the controller looks at this small window. This helps:

- avoid sudden jumps when one sample is bad,
- keep a consistent direction of motion,
- make the robot follows the human motion in a stable way.

For every points received from the watch, 2 more points added in between the last point and the new one, to get three times the amount of points to be inputted into the robot, thus making it smoother. The pipeline will then run at 120hz to make sure to have the most recent point possible.

At each control step, the pipeline runs as follows:

1. Read the current joint states of the robot from ROS 2.
2. Read the latest filtered target from the smartwatch pipeline (using the horizon buffer to smooth it).
3. Use inverse kinematics (Pinocchio) to compute a set of joint angles that moves the wrist toward this target, while respecting joint limits.

4. Build a short joint trajectory (current position \rightarrow target position) with a small time horizon.
5. Send this trajectory to the standard ROS 2 controller using the `FollowJointTrajectory` action.

This loop runs continuously while teleoperation is active. Because new targets arrive at 40 Hz, the filter and horizon keep them smooth, and IK is solved at each step, the robot end-effector follows the user motion in a stable and continuous way, without sharp jumps or unsafe commands.

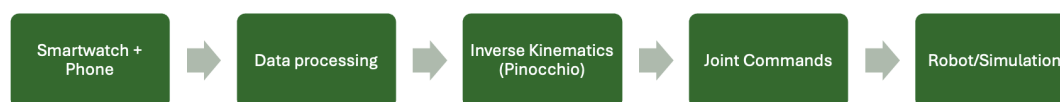


Figure 3.1: Full Teleoperation Pipeline

Experiment Setup

Due to the limitation of gazebo's physics, an experiment was firstly tried but ended up in a failure, which made us rethink about a second experiment that would make it less physics dependent.

First Experiment

For the experiment, a ping pong racket will be added at the end of the wrist of the robot in both the simulator and the robot.

What we will be trying to achieve is making the robot be able to make a ping pong ball bounce, so, in this first experiment, we will have a ping pong ball hanging, attached to a string.

To simulate the ball on gazebo, multiple ways were possible, but only one was achievable.

The first one, and most straightforward one, is to make a rope with multiple links and joints, with the ball attached to the last join. In theory it is a good way for a more visual kind of rope, however, in our case, the rope physics is important to have the ball follows a correct path after being hit. This result in having the ball often start spiraling because of the stress that the joints build up, preventing the ball from moving freely like it would in the real world. As stated before, Mujoco has way better physics than gazebo, and the rope being built with that simulation engine would probably have given way better result.

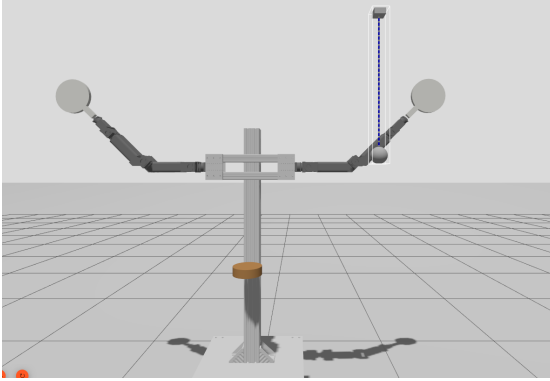


Figure 3.2: Gazebo simulation of the

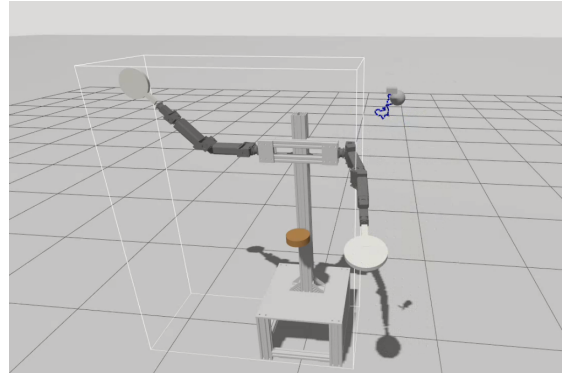


Figure 3.3: Gazebo simulation when the physics fail

The second option that was tried is to have an invisible box/sphere where the ball would be inside and that couldn't leave the box. However, the racket still should be able to go through that box as it wasn't there. The 2 different collisions constraint, in the gazebo, is hard to do, and even while trying to set different collision bitmasks between the racket and the box. Mujoco also seemed to do it quite easily.

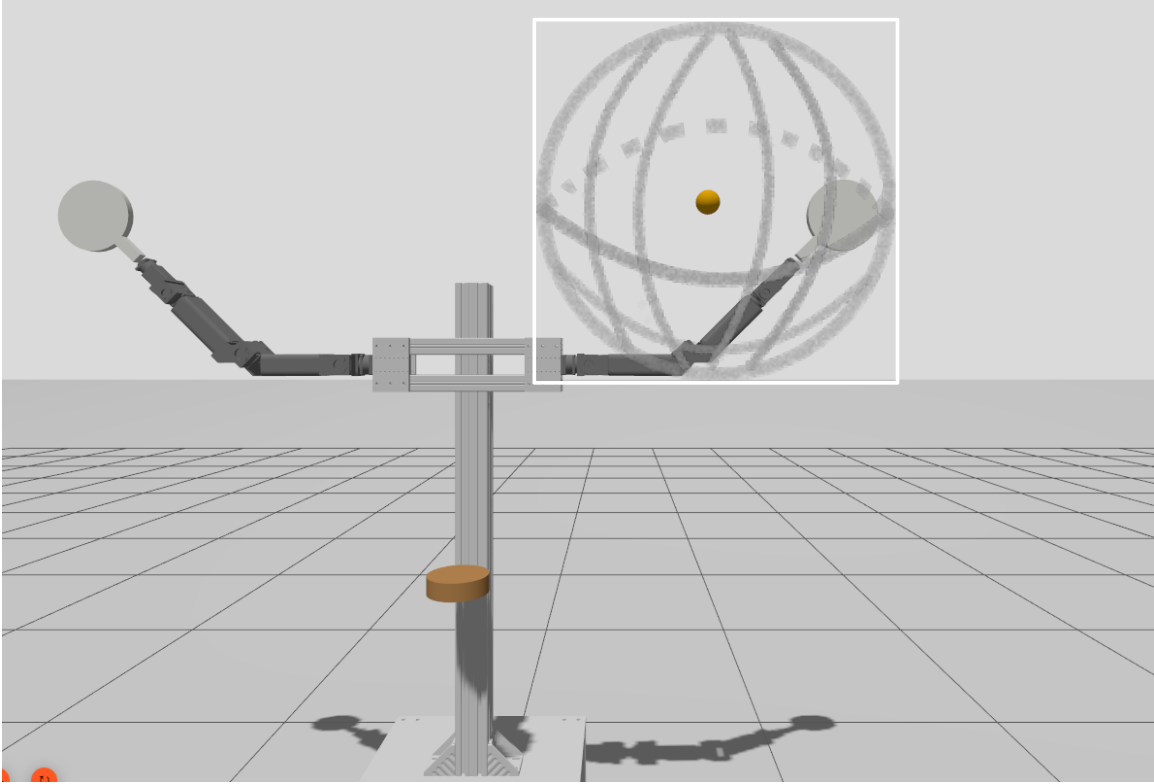


Figure 3.4: Gazebo simulation of the Sphere/box

The third option was to have a sphere around the ball spawn point and whenever the ball would touch the sphere with a certain speed, a small force would be applied to the ball in the other side to redirect it toward the center, like a rope would act. This was also hard to have the right force and have an algorithm that would read the pos of the ball fast enough. Additionnaly, I tried hidding a small plank to have the ball rest on, before hitting it, so it would be bouncing continuously against the bottom of the sphere, but that take us back to the second problem with the collision issue.

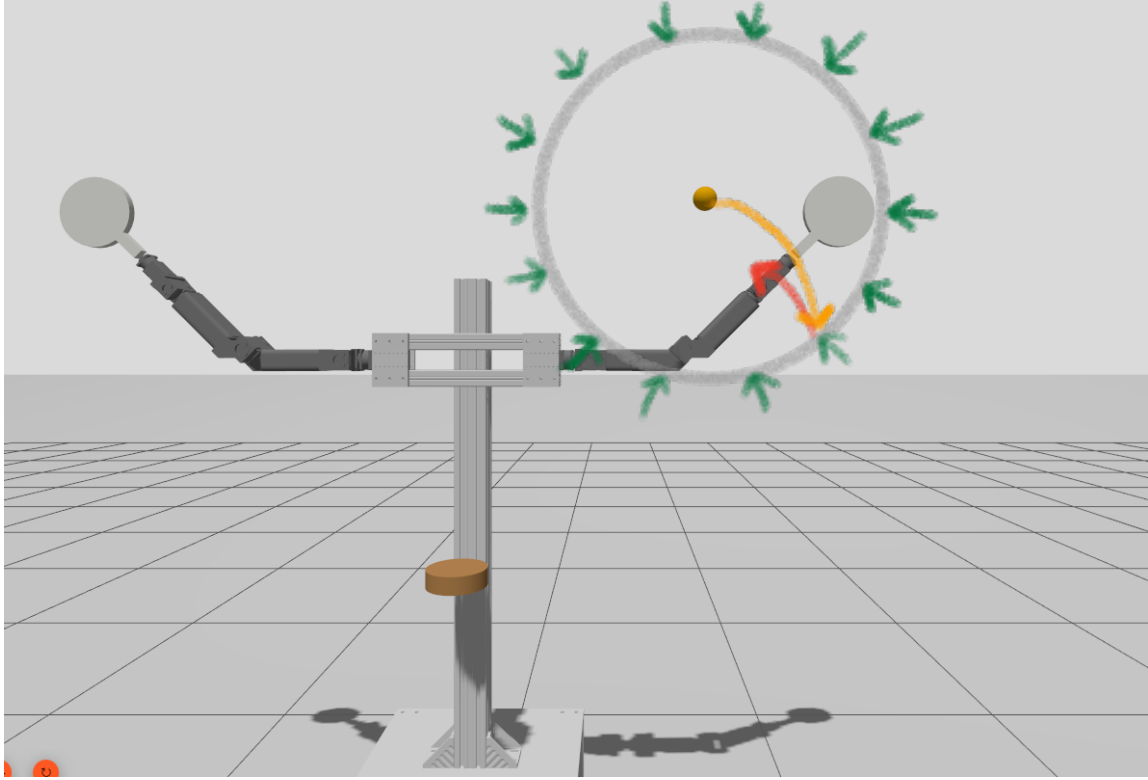


Figure 3.5: Gazebo simulation of force field

In the end, the simple rope with the ball was used for the experiment on the simulator.

However the physics, were still causing a problem, making the ball act unpredictably.

Second and Final Experiment

For this second experiment, a Tether Ball setup will be used. As explained previously, the rope physics make the ball follow an unpredictable path, in order to minimize it, the tether ball game solves it in an easy way. As we hit the ball and launch it around the pole, the rope is mainly stretched, with only starting to curve while it goes around the pole, making the links and joints move slowly.

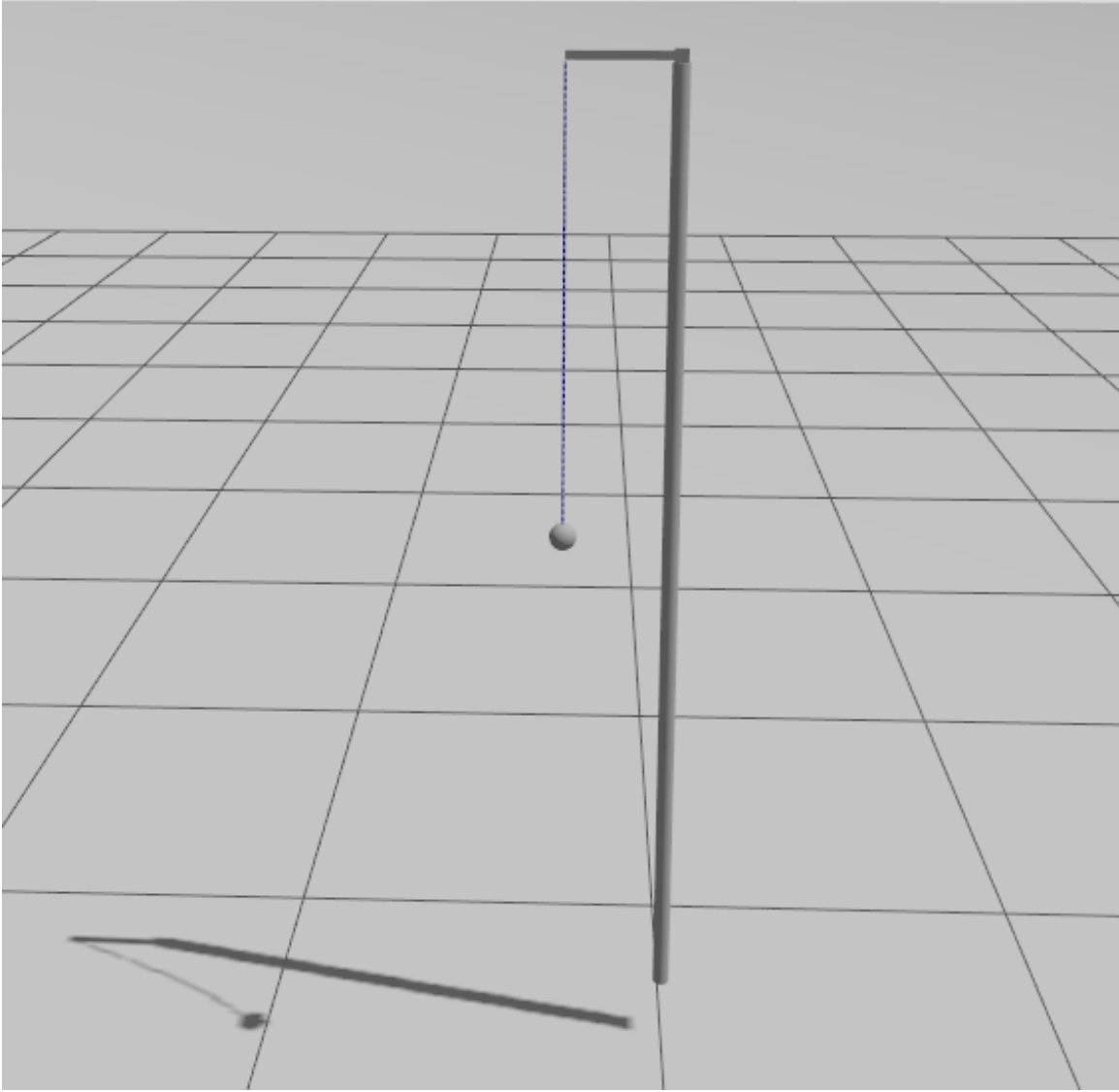


Figure 3.6: Gazebo Tether Ball Setup

This setup came with its own problem, as to be able to have the ball go around the ball in only one hit, the ball should not be straight below the pole, but should be already bent, somewhere in the air. The solution that came up for gazebo was to start it with sending a touch to the ball, making it go up around an axe and positioning itself to be hit.

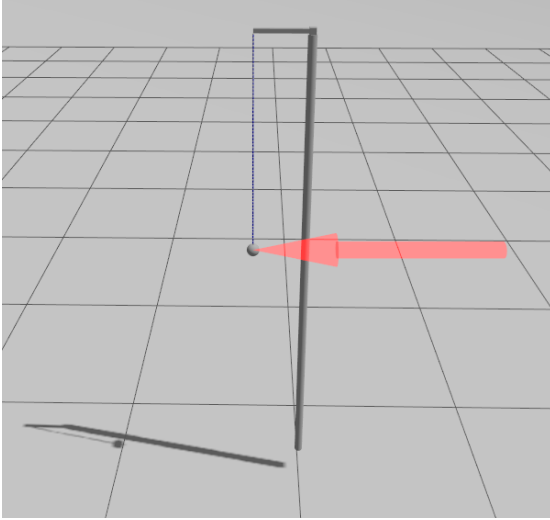


Figure 3.7: Force Applied to the ball

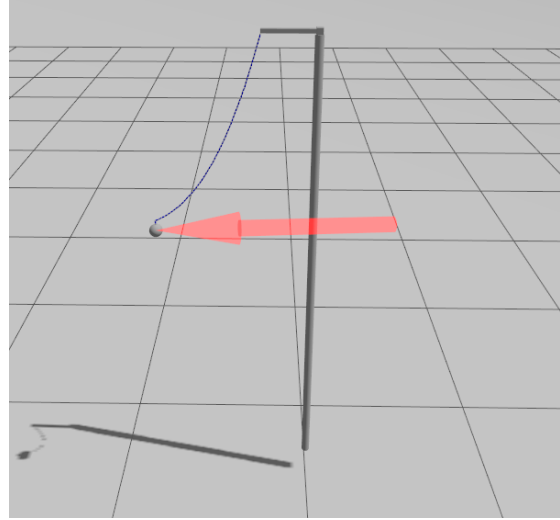


Figure 3.8: Hit-Ready Position

Once the ball positioned, the experiment can continue as the ball will be hit by the racket and wrap around the pole.

Chapter 4

METHODS

DMP

Dymanic Movement Primitives are a way to encode dynamical systems instead of raw trajectories. DMPs act like a spring-damper system, which means it behaves like point pulled toward the goal, while slowing it down, to ensure that the motion converges to the goal.

On top of this, Dmps are a way to mathematically encode the shape of a trajectory instead of using a list of points. With one demonstration, the DMPs learn a set of equations/weights that describe the curve and how its motion change over time. Those weights are used to recreate a smooth version of the same movement.

Mathematical Formulation

We employ discrete, goal-variant DMPs following the formulation of Ijspeert et al. [7]. The system consists of three coupled differential equations:

Transformation System

The core of the DMP is a second-order dynamical system that governs the trajectory evolution:

$$\tau^2 \ddot{y} = K(g - y) - D\tau \dot{y} + K \cdot f(s) \cdot (g - y_0) \quad (4.1)$$

where:

- $y \in \mathbb{R}^3$ is the current 3D position (x, y, z coordinates)
- \dot{y} and \ddot{y} are velocity and acceleration
- $y_0 \in \mathbb{R}^3$ is the start position
- $g \in \mathbb{R}^3$ is the goal position
- $\tau \in \mathbb{R}^+$ is the temporal scaling factor (controls duration)
- $K = 1000.0$ is the spring constant (stiffness)
- $D = 2\sqrt{K} \approx 63.25$ is the damping coefficient (near-critical damping)
- $f(s)$ is the nonlinear forcing function that shapes the trajectory
- $s \in [0, 1]$ is the phase variable from the canonical system

The term $K(g - y)$ acts as a spring pulling the system toward the goal, while $-D\tau\dot{y}$ provides damping to prevent oscillations. The critical innovation is the forcing term $K \cdot f(s) \cdot (g - y_0)$, which modulates the trajectory shape while maintaining goal-invariance through the $(g - y_0)$ scaling factor.

Forcing Function

The forcing function $f(s)$ is represented as a weighted sum of Gaussian basis functions:

$$f(s) = \frac{\sum_{i=1}^M \psi_i(s) w_i}{\sum_{j=1}^M \psi_j(s)} \cdot s \quad (4.2)$$

where:

- $M = 35$ is the number of radial basis functions (RBFs) per axis
- $w_i \in \mathbb{R}$ are the learnable weights (35 per axis, 105 total for 3D)

- $\psi_i(s)$ are Gaussian basis functions defined as:

$$\psi_i(s) = \exp(-h_i(s - c_i)^2) \quad (4.3)$$

where:

- $c_i \in [0, 1]$ are the centers of the basis functions, uniformly distributed
- $h_i > 0$ are the widths controlling the Gaussian spread
- The multiplication by s ensures the forcing term decays to zero as $s \rightarrow 0$

The design allows for a lot of flexibility. DMPs give us a baseline of the demonstration, with the start and end goal, and then the weights that permit to reconstruct the motion. The weights allow us to modify a lot of parameters of the demonstration, such as :

- Temporal scaling : allows us to change the speed of the movement.
- Spacial scaling : allows to change the start or end goal without having to re-learn everything.
- Shape changing : the weights allow to change the trajectory without changing the end goal or start goal

We record the wrist position from the smartwatch while doing the arm action (touching the ball hanging from the string). We run these recordings through the DMP, which gives us a baseline and the weights that we chose. To replay the demo, we just use the same baseline + weights and sample the 3D position at a fixed interval. Those 3D points go into inverse kinematics, and we stream the resulting joint values to the robot. This gives a near-perfect replay of the motion, with smoother curves and more stable behavior than the raw points.

For the first experiment setup, we were recording multiple hits through a single weight file. With DMPs giving us a similar, but smoother trajectories, with however the damping effect causing a slight mismatch by the end of the DMPs.

For the final experiment setup, we recorded a single motion, from start to the end goal, being the position of the racket one second after hitting the ball. To make it easier, in the baseline, the starting point was the average of all the starting points of the episodes and so was the end point done the same way, for the weights to be easily changed in the same context for all of them.

ICL

In-context learning (ICL) enables large language models (LLMs) to perform a task by learning from examples provided directly in the prompt, without any gradient-based training on that specific task. Recently, Prompted Policy Search (ProPS) [15] showed that LLMs can use ICL to do policy search by combining numerical rewards and language feedback. This thesis follows the same high-level idea and uses ProPS as its main inspiration.

LLMs have strong pattern recognition skills, which have been developed for their chat bots, and that is what ICL takes advantage of. Given successful and unsuccessful demonstrations, the LLM can identify the features that separate good hitting motions from poor ones and generate new trajectories accordingly, with DMP in our case.

The way to determine what is a success or a failure, other than a straight "failed binary situation", is by introducing rewards and state transitions. The LLM will match the rewards with certain repetitive patterns and generate, in our case, new trajectories that inherit the desired properties.

Unlike traditional imitation learning or Reinforcement learning approaches, that needs extensive training data and computation. Only a few demonstrations, around 20 is a good start, are sufficient for the model to be able to extract task-relevant patterns and generate new behaviors.

Data Collection and Processing

Demonstrations are collected through teleoperation, where the controller, a human wearing the smartwatch, will hit the suspended ball 3 to 4 times for each episode.

With every episode, a few set of data will be collected, such as :

- 3D wrist position : used for the DMP trajectories.
- 3D Ball position : The ball position with gazebo for the simulation, or with a machine vision algorithm for the real world experimentation. The ball position will be used as the sole reward function parameter.
- Contact events : timestamp, positions and the distance between the ball and the racket when contacts occurs. It's the most useful way to see when contact has been done and that we should see the effects of it on the ball position.

Reward function

First Experiment Setup

For the earlier ball-on-string experiment, we scored both each hit and the whole episode. Unlike a pure binary metric (e.g., PROP), our reward had four small parts:

- **Height gain:** vertical rise after contact; success needs at least 10 cm.
- **Centering:** penalty for drifting away from the rope center line.
- **Descent:** bonus for settling instead of spiraling (a Gazebo rope-physics issue).
- **Speed:** lower speed with enough height is preferred (rope would cap travel anyway).

A hit is **successful** only if it passes both checks: height ≥ 10 cm and drift \leq max allowed. The **episode reward** mixes how many successful hits we get with their average quality, so we favor multiple good hits rather than a single perfect one.

Final Experiment Setup

The reward function we used for the tether ball is a more direct one. to reduce the uncertainty of the gazebo world, the reward function consist of only one element :

- **Checkpoints:** The checkpoints will be working as quadrants, every time a ball enters a quadrant around the pole, such as North-West, North-East, etc..., while making sure that it follows a counter-clock rotation, the reward will count as +1.
- A full wrap will count as +10 to better simulate the "tether ball" game; however, as it is the only reward, it can not be used and only rely on the checkpoints passed.

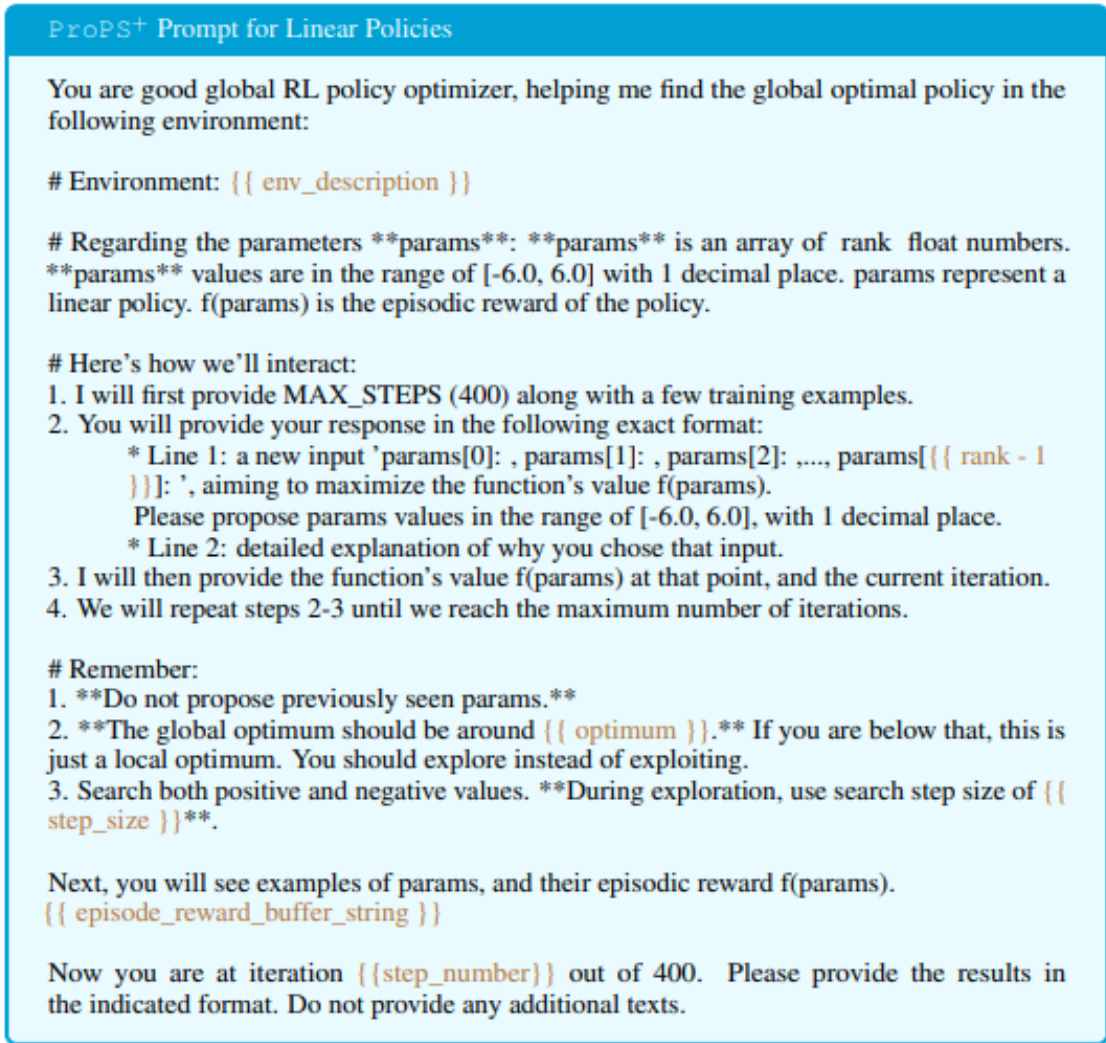


Figure 4.2: Prompt Example from ProPS paper

This will be used as an inspiration format for the Prompt we use in both our experiments. There will be some small changes in the prompt.

First Experiment Setup

- Objective: Achieve 3-4 consecutive successful ball bounces
- Success criteria: Height ≥ 10 cm, centering ≤ 50 cm, ball returns downward
- Robot embodiment: 6-DOF bimanual arm with ping pong racket end-effector

- Environment constraints: Ball constrained by rope attached at (0.78, 0.55, 2.1) m
- Physics context: Rope introduces pendulum dynamics and limits ball motion

PROPS shows the importance on grounding the task in the robot’s physical capabilities and workspace constraints. 2. Demonstration : Out of the 20 demonstrations, we will keep around 5 of the best ones and 3 of the worse ones. For reason that is has to fit within the LLM’s context window. The better the demonstrations, the faster the LLM will learn successful patterns from it.

It would then create a Json with the generated weights and expected rewards. After running the generated weights through the reward function, they will both be inputted into the prompt with feedback to generate new weights and so on.

Final Experiment Setup

For the tether experiment, PROPS’ will still be used but with a different environment task. The Prompt format is :

- **Task:** Tetherball Around a Pole - Quadrant Wrap Control
- **Objective:** Generate Dynamic Movement Primitive (DMP) weights for the robot wrist so that, when executed repeatedly, the racket hits the ball and makes it go around the pole as many times as possible.

The ball is attached to a rope on a vertical pole. In the xy-plane, we track its position relative to the pole and assign a quadrant index: - SE (0) - NE (1) - NW (2) - SW (3) - SE (0) - ...

Each ****forward quadrant transition**** (0-1-2-3-0) is a checkpoint.

Every 4 checkpoints = 1 full wrap around the pole.

- **Reward Definition (per episode)** - +1 reward for each ***forward*** quadrant change (checkpoint) - +10 extra reward each time 4 checkpoints complete a full wrap - Total episode reward = sum of these step-wise rewards

Therefore: - More wraps and more correct quadrant transitions \Rightarrow higher episode-reward -
We want DMP weights that maximize episode-reward (wraps).

- **Action Space:** DMP Weights for the Wrist

You will generate weights that define the 3D wrist trajectory: - We use a goal-normalized DMP formulation - All episodes share common baseline: - Start position y_0^* (baseline start) - Goal position g^* (baseline end) - Weights shape motion in this normalized space - The same weights are executed repeatedly in the episode.

- **Weights Examples:** [This parts requires get the baseline and weights from the files in python] starts with baseline information such as the start position and goal position. For the dmp parameters: Basis function, Weight dimensions, Motion duration.

```

# GOOD EPISODES
for i, episode in enumerate(good_episodes):
    prompt += f"### Episode {i+1}: GOOD EXAMPLE\n"
    prompt += f"- episode_id: {episode['episode_id']}\n"
    prompt += f"- episode_reward: {episode['episode_reward']:.3f}\n"
    prompt += f"- wraps_completed: {episode['wraps_completed']}\n"
    prompt += f"- checkpoints_passed: {episode['checkpoints_passed']}\n"
    prompt += f"- num_steps: {episode['num_steps']}\n\n"

    dmp_data = load_dmp_weights(episode['episode_id'], logs_dir)
    if dmp_data:
        prompt += "**DMP weights used in this episode (preview):**\n"
        prompt += "```\n"
        prompt += format_dmp_weights_compact(dmp_data["weights"], n_preview=8) + "\n"
        prompt += "```\n\n"

    prompt += "**Why this is good:**\n"
    prompt += "- High reward and multiple wraps around the pole.\n"
    prompt += "- Quadrant transitions follow the correct order SE->NE->NW->SW->SE many times.\n"
    prompt += "- The same DMP weights, when repeated, keep producing forward wraps.\n\n"
    prompt += "...\n\n"

# BAD EPISODES
if bad_episodes:
    for i, episode in enumerate(bad_episodes):
        prompt += f"### Counter-Example {i+1}: POOR EXAMPLE\n"
        prompt += f"- episode_id: {episode['episode_id']}\n"
        prompt += f"- episode_reward: {episode['episode_reward']:.3f}\n"
        prompt += f"- wraps_completed: {episode['wraps_completed']}\n"
        prompt += f"- checkpoints_passed: {episode['checkpoints_passed']}\n"
        prompt += f"- num_steps: {episode['num_steps']}\n\n"

        dmp_data = load_dmp_weights(episode['episode_id'], logs_dir)
        if dmp_data:
            prompt += "**DMP weights (preview):**\n"
            prompt += "```\n"
            prompt += format_dmp_weights_compact(dmp_data["weights"], n_preview=8) + "\n"
            prompt += "```\n\n"

        prompt += "**Why this is bad:**\n"
        if episode['wraps_completed'] == 0:
            prompt += "- No full wraps completed → motion fails to move the ball around the pole.\n"
        else:
            prompt += "- Very few wraps compared to good examples → reward is much lower.\n"
        prompt += "- The same DMP weights do not produce many forward quadrant transitions.\n\n"
        prompt += "...\n\n"

```

Figure 4.3: Good and Bad Episodes Prompt

- **Your Task:**

Using the patterns above, generate NEW DMP weights for the wrist that will: - Produce strong, consistent hits on the ball - Make the ball travel around the pole in the correct order of quadrants - Maximize the number of wraps and thus the quadrant-based reward

Focus only on: - Increasing ****wraps-completed**** - Increasing ****episode-reward**** (sum of checkpoint and wrap bonuses)

You can assume: - The same baseline y_0^* and g^* will be used. - The DMP system, basis

functions, and time scaling are unchanged.

- **Required Output Format**

```
'''json
{{
  "dmp_weights": [
    [w_x1, w_x2, ..., w_x{baseline['M']}], // X-axis weights ({baseline['M']} values)
    [w_y1, w_y2, ..., w_y{baseline['M']}], // Y-axis weights ({baseline['M']} values)
    [w_z1, w_z2, ..., w_z{baseline['M']}]  // Z-axis weights ({baseline['M']} values)
  ],
  "reasoning": "Short explanation of why this weight pattern should increase wraps.",
  "expected_outcome": {{
    "predicted_wraps": <int>,           // expected number of wraps in one episode
    "predicted_checkpoints": <int>,     // expected total forward quadrant transitions
    "confidence": <float between 0 and 1> // how confident you are in this prediction
  }}
}}
```

- **Guidelines:**

1. Keep weight magnitudes in a similar range as the good examples above.
2. Preserve the overall movement direction and rhythm visible in the good DMPs.
3. Avoid extremely large weights that would create unstable or chaotic motion.
4. Concentrate on shaping a repeatable swing that continuously advances quadrants.

Respond with ****JSON only****, with no extra text and no markdown fences.

Chapter 5

RESULTS

DMPs Implementation and Trajectory Generation

The dynamic Movement Primitive framework was successfully implemented to encode and reproduce the wrist positions obtained from the watch. Using goal-variant DMPs helped facilitate the results and enabled an accurate reproduction of the movement.

The wrist positions recorded directly from the smartwatch appeared slightly more scattered and shaky. This is expected, because of the sensor noise coming from the smartwatch mocap software. When applying the DMPs, the resulting 3D curve is much smoother and continuous, while still following the general path, except for one line, that is the ending line and how fast the motion was, giving less points for the DMP to have to follow. This demonstrates that the DMPs are able to clean up the signal, reduce noise, and produce stable and reusable trajectory. However, as seen in the 2 plots below, by the end, some damping effect can be seen for the final trajectory toward the end goal.

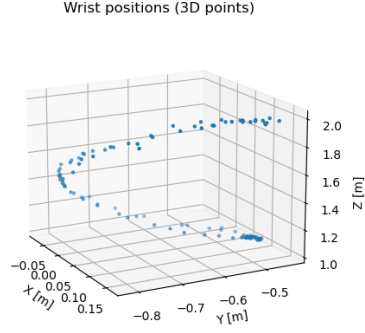


Figure 5.1: Wrist Positions (3D points)

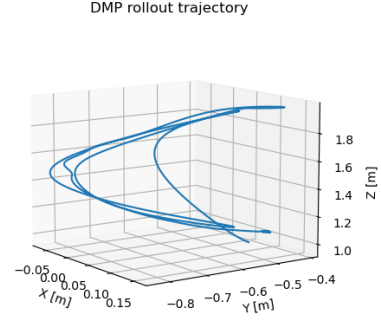


Figure 5.2: DMP rollout trajectory

For the final experiment setup, with only a single movement, the damping effect is close to non-existent.

Reward Function and Episode Assessment

First Experiment Setup

The reward function successfully quantified episode quality based on three primary metrics that we set : Height gain (10cm minimum), centering (maximum drift of 50cm from rope anchor), and descent validation (minimum 8cm). Episode rewards were computed as the weighted combinations of these components, with height weighted at 2.0, centering at 1.5 and descent at 0.5. reflecting the different importance for task success.

Final Experiment Setup

The reward function also successfully quantified the episode quality of every episode with the checkpoint metric that we set. Using the ball position recorded after the contact with the ball, it was able to correctly calculate how many quadrants the ball went through, while keeping in mind that it has to be in a counter clock way.

A few of these examples can be seen below, with a color for every quadrants and also at the end how many checkpoints were passed and wraps were done.

The first example was of a normal, good, hit. The results were : 7 checkpoints passed and one full wrap. Rewards : 17.

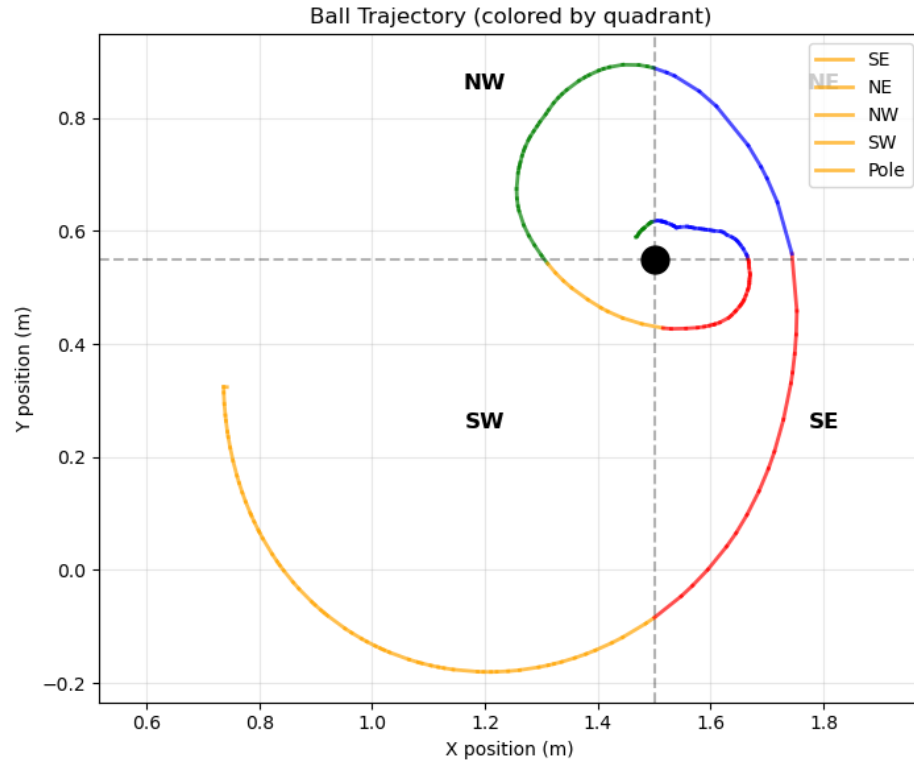


Figure 5.3: Basic episode reward plot

The second example was of a good, close to perfect, hit. The results were : 13 checkpoints passed and 3 full wrap. Rewards : 43.

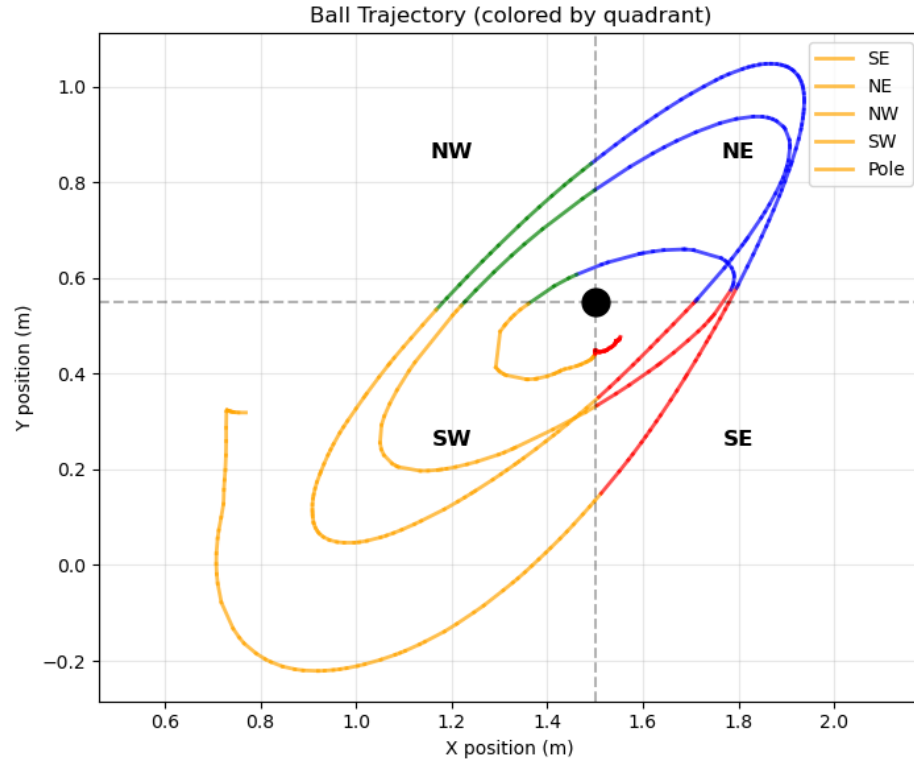


Figure 5.4: Good episode reward plot

The first example was of a bad hit. The results were : 0 checkpoints passed and no full wrap. Rewards : 0. Even tho the ball wrapped around the pole, it was not on the right side of it, making it a fail.

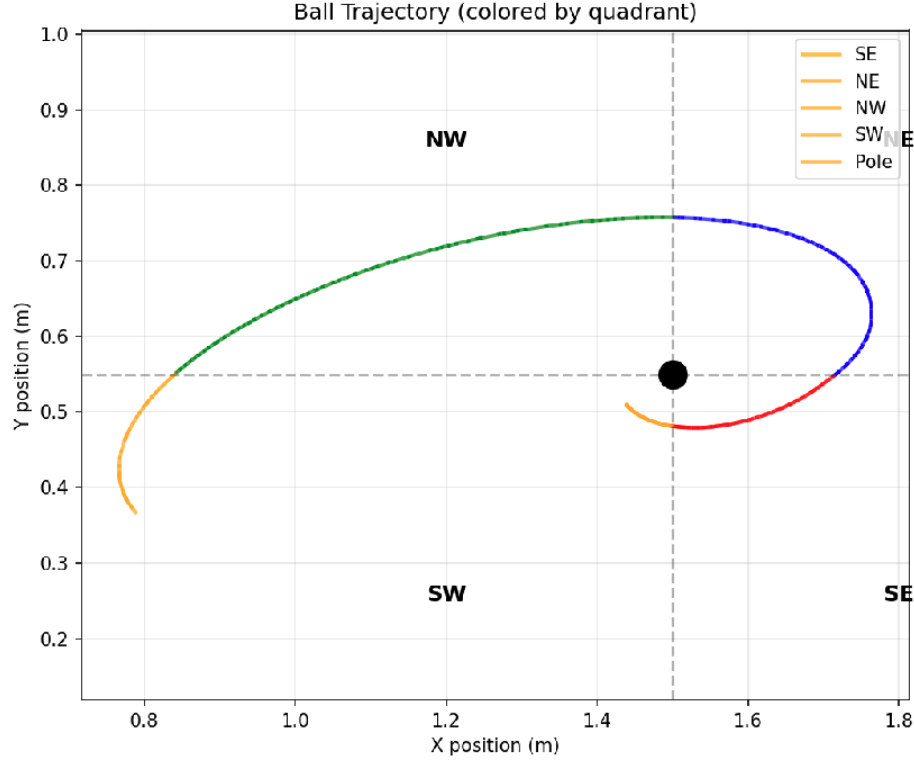


Figure 5.5: Bad episode reward plot

Analysis of Ball bouncing Failure Expiement

Quantitative evaluation of ball bouncing performance

We evaluated a total of 18 recorded episodes. The average consistency score across all episodes was 0.762.

Table 5.1 shows the three best episodes based on their reward. In these trials, most hits were successful: the ball reached a reasonable height, and it did not move too far away from the center. Their consistency scores are all above 0.75, which means the behavior was quite stable.

In the best episode, all three hits were successful, the ball stayed close to the center line, and the motion was very similar across hits. For the 2 others, there is a problem with the way the ball position and the collision is being detected. The Simulation using already a high part of the cpu use FollowJointTrajectory, making multiple requests every second to get the ball position on top of it, makes the whole simulation less stable and sometimes has built up lag. Calculating the ball position through a camera would solve a lot of the problems we currently have.

In contrast, the three lowest-ranked episodes showed almost no successful hits, very low bounce heights, and higher variation in drift over time. Their consistency scores are much lower (around 0.45 on average), reflecting unstable control and poor repeatability. All the demonstrations were made by hitting the ball so it just means that it was either a problem with the sensors or with the rope physics making the ball bounce in an unusual way.

Table 5.1: Top 3 episodes by reward.

Episode ID	Reward	Successful hits	Avg height [m]	Consistency
20251108_001459	0.919	3/3	0.158	0.914
20251109_004841	0.759	2/3	0.104	0.901
20251108_002123	0.723	2/5	0.057	0.783

Generated Weight Characteristics of Ball Bouncing

With the successful and failed demonstration episodes, the model consistently generated DMP weights of too small magnitude. The generated weight ranged from -0.02 to +0.02, with total weight energy averaging 0.8-1.12. In contrast, the actual weights ranged from -0.35 to +0.35 with total energies of 12-18.

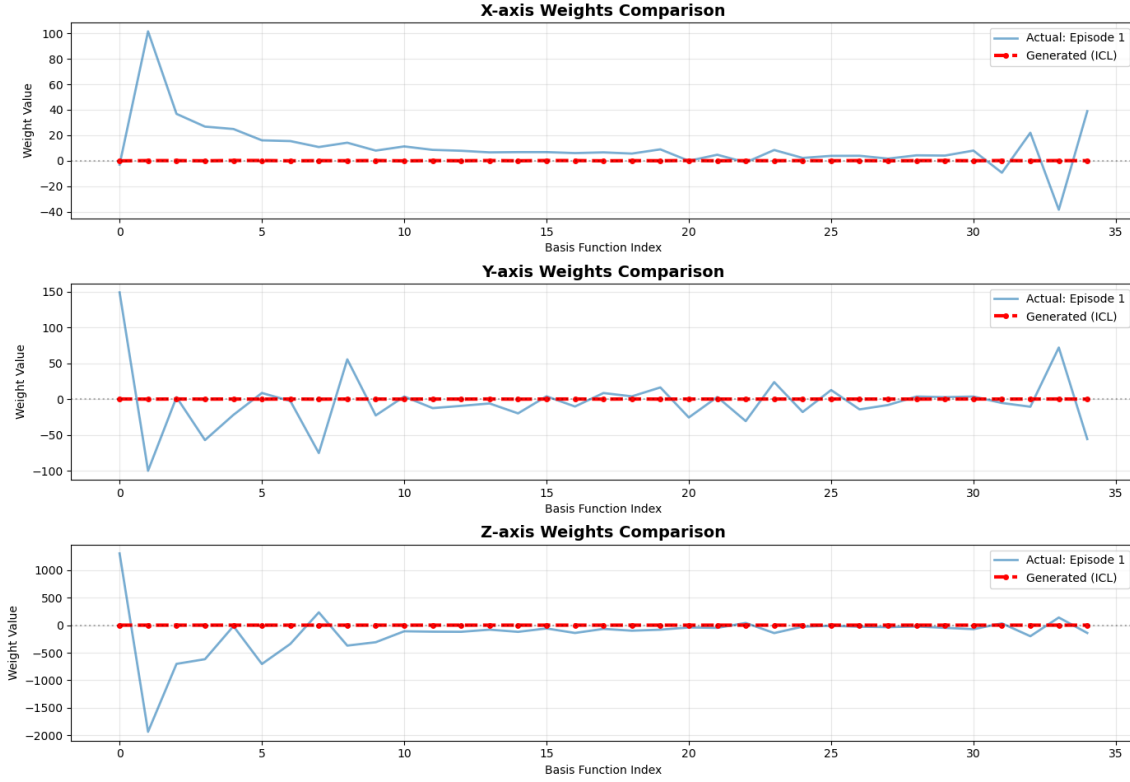


Figure 5.6: Weight comparison with generated weights

Executing those weights on the simulation produce minimal motion. The robot arm didn't move more than 5cm up and down, not even near enough the ball position, that is approximately 50 cm away. Even when giving feedback to the ICL by doing the generated weight rewards and some written feedback, the AI wasn't able to get a motion close to a full arm movement.

Root Cause Analysis

Further investigation into the limitations of In-Context learning and its capabilities, it revealed a fundamental architectural mismatch between the DMP structure and the ICL learning paradigm.

Temporal Bundling Problem:

The teleoperated demonstrations encoded for an average of 20 seconds with 3

to 4 hitting cycles within a single DMP weight sets. Each weight influenced the motion accros the full duration, for those multiple hits, with different basis functions activating during different hits.

The Language received the weights as a .npz file with flat arrays of 105 numbers (parameters of the DMP) with the associated outcomes, but those different situations made it hard for the AI to find patterns to understand :

- Which weights controlled hit 1 versus hit 2 versus hit 3
- how basis function temporal activation related to hit timing
- why changing the weights at weight number 10 would affect hit outcomes differently than if we changed at weight number 25

While we had the contacts recorded in a second file, coupling the DMPs with the contacts was still a task in itself to make ICL understand it.

Insufficient demonstration. With such a complex task, there would be a need of using more than a hundred demonstrations for the AI to start discerning some patterns, which is not possible with our 17 tests.

Conclusion of The Experiment

This experiment ended up in a **Failure**, due to the reasons explained, and the second experiment was then thoroughly thought about, with the gazebo physics constraints faced and also the things learned with the usage of DMPS.

Analysis of Successful Tetherball Experiment

After the first In-Context Learning iteration, the generated weights were already close to perfect, considering the successful examples. It is a possibility that it made

the task too easy as the first hit was an almost exelet hit, losing the possibility for the LLM to learn by itself how to maximize results. However it still managed to slightly maximize it.

The 5 successful and 3 unsuccessful examples used in the prompt were :

```
=====
SUMMARY (Quadrant-only reward)
=====
Total episodes processed: 21
Total wraps: 8
Total checkpoints: 50

Top 5 episodes:
1. 20251118_212504: reward=43.000, wraps=3, checkpoints=13
2. 20251118_210616: reward=29.000, wraps=2, checkpoints=9
3. 20251118_213150: reward=28.000, wraps=2, checkpoints=8
4. 20251118_215824: reward=17.000, wraps=1, checkpoints=7
5. 20251118_214914: reward=3.000, wraps=0, checkpoints=3

Worst 3 episodes:
19. 20251118_214646: reward=0.000, wraps=0, checkpoints=0
20. 20251118_215118: reward=0.000, wraps=0, checkpoints=0
21. 20251118_215452: reward=0.000, wraps=0, checkpoints=0
```

Figure 5.7: Prompted Examples

The result of the first iteration came out as json with :

- **Weights:** A list of the 105 weights
- **Reasoning:** "The new weight pattern slightly adjusts the successful patterns of good episodes by enhancing the wrist movement rhythm and consistency, aiming to increase the strength and predictability of the racket-ball interaction. These adjustments are based on maintaining the overall movement direction while ensuring higher energy transfer for consistent quadrant transitions."
- **Expected outcome:** "predicted wraps": 4, "predicted checkpoints": 17, "confidence": 0.85

First Iteration

After running those generated weights throught the robot, the results came as : 16 checkpoints passed and 4 full wrap. Rewards : 56

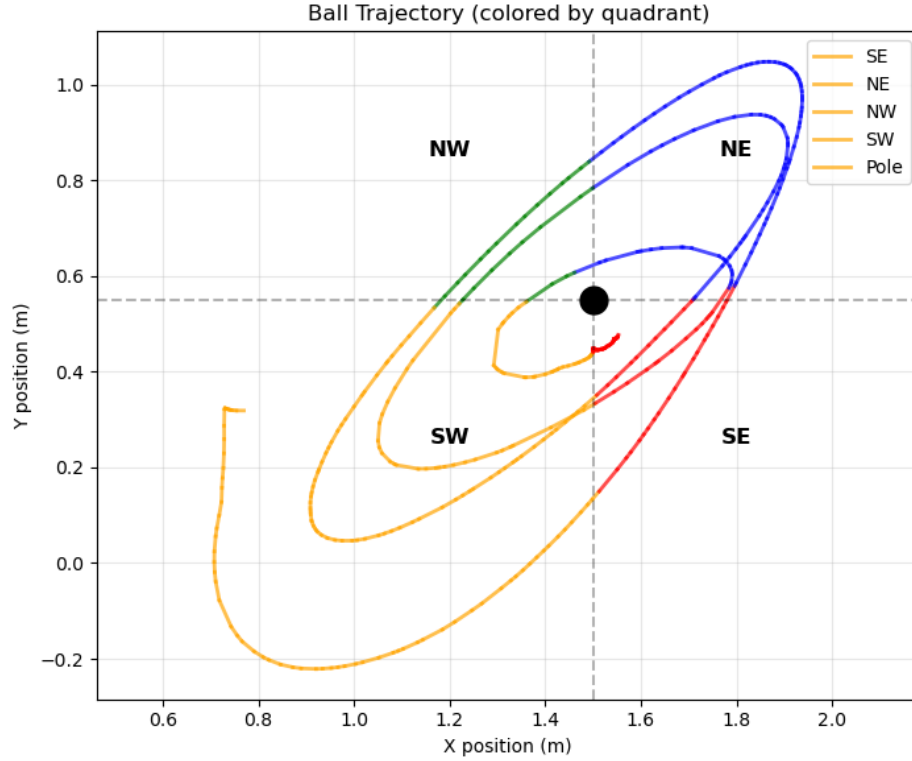


Figure 5.8: Good episode reward plot

Once again Gazebo make it difficult to run the same simulation, because of subscriber delay for the ball recording, robot moving and other internal software. However, when the setup was perfect, the results came out as better than what the LLM expected.

Those simulation problems make it so it's not possible to have the future iterations run repetitively by itself and thus making it harder to try 10 to 20 iterations.

Third Iteration

The LLM-generated third weights and the reasoning were :

- **Reasoning:** The updated DMP weights integrate successful patterns from the good reference examples by maintaining similar weight ranges and behaviors, particularly the high variability and range seen in successful episodes. The weights were adjusted to maintain stability while increasing the variability to enhance forward quadrant transitions and wraps.
- **Expected Improvement:** " The adjustments should result in smoother and more consistent transitions across quadrants, thereby increasing the number of wraps and overall reward.
- **Expected outcome:** "predicted wraps": 5, "predicted checkpoints": 20, "confidence": 0.8

However, the results we got while running it on the robot and getting the rewards for that episode were not as good as predicted, showing a slight downgrade from the first iteration. This can be explained by the fact that the LLM tried another way to get those weights, maybe learning from the less successful examples, to not be bound to the good first one.

The third iteration gave us : Checkpoints passed: 7, Wraps completed: 1, Total reward: 17.0

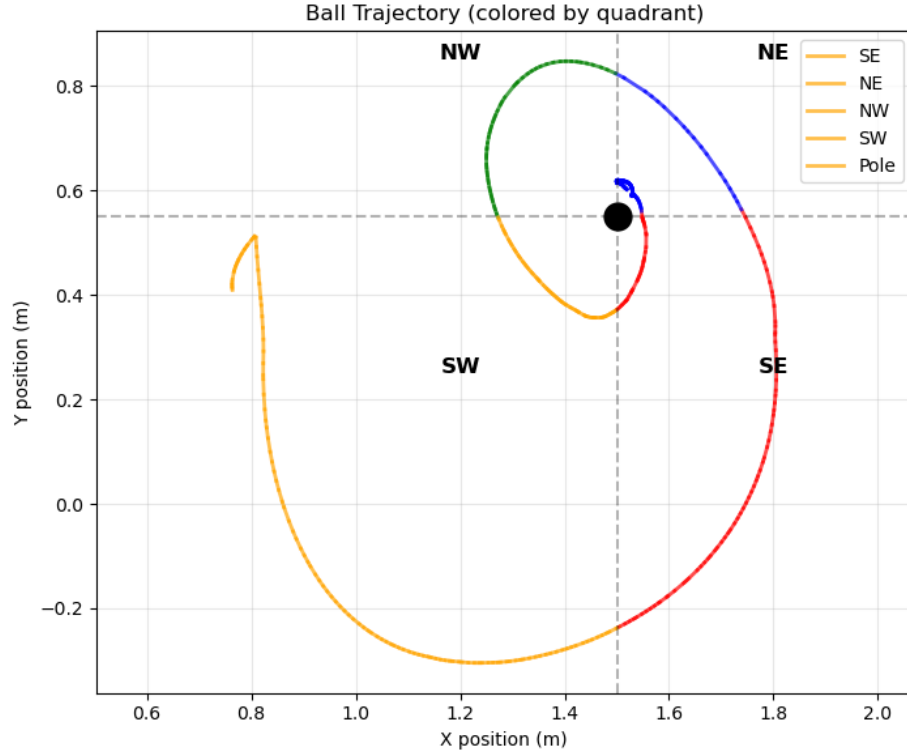


Figure 5.9: Third Iteration plot

Fifth Iteration

The LLM-generated third weights and the reasoning were :

- **Reasoning:** he weights were adjusted to have a more balanced distribution across the three axes, similar to high-performing reference episodes. The X-axis weights were slightly reduced to avoid overly aggressive motion while maintaining high values for strong movement. The Y and Z axes were adjusted to resemble successful patterns, increasing mid-range values to ensure smooth quadrant transitions. This approach aims to increase force and accuracy, addressing the feedback about needing more force.
- **Expected Improvement:** ” With these changes, we expect smoother and more forceful transitions between quadrants, resulting in increased wraps and

episode reward.

- **Expected outcome:** "predicted wraps": 3, "predicted_checkpoints" : 12, "confidence" : 0.8

After the fifth iteration, the LLM tried to work from the last 4 ones, learning to maximize it with that reasoning, not using the first one. We can see it in the prediction as it predicts way lower values than the first iteration but much more reasonable.

The fifth iteration gave us : Checkpoints passed: 9, Wraps completed: 2, Total reward: 29.0.

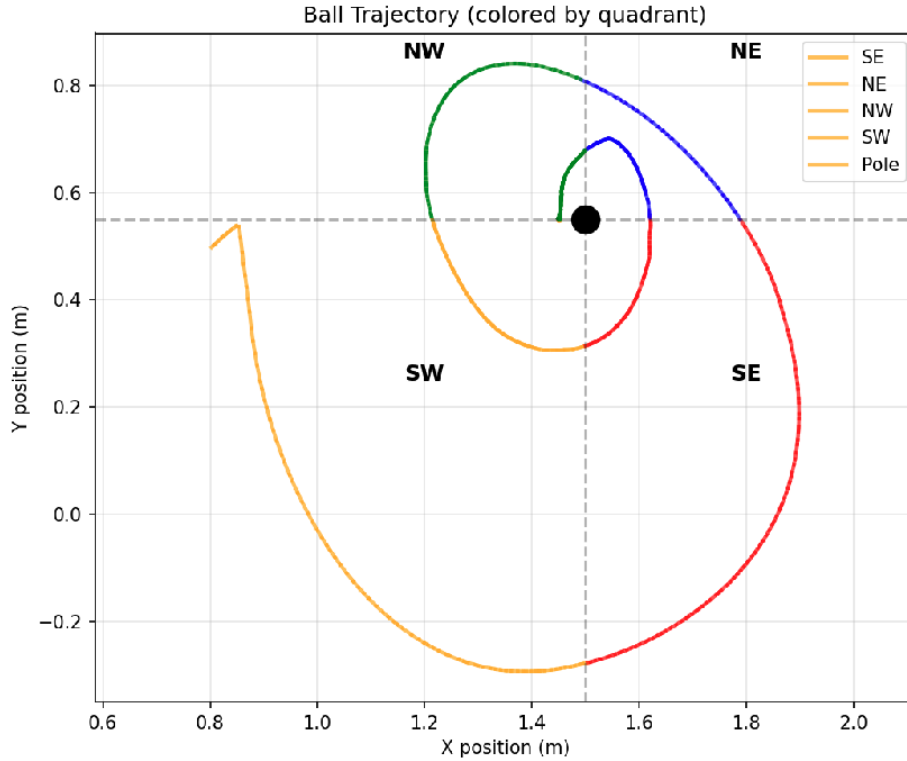


Figure 5.10: Fifth Iteration plot

After the fifth iteration, we can conclude that the llm totally forgot about the first example, and tries to build up from the other ones, which shows an improvement after every iteration. Those results are better than just slowly following the first example as it shows clear learning done by the LLM.

Simulation Environment and Ball Physics

The Gazebo simulation environment successfully gave visual dynamics for the ball-rope model. However it came with a lot of limitations.

- Ball State Estimation : The position Data was the most reliable data of those demonstrations. However there were still some latency introduced by Gazebo topic subscriptions, which was fighting with other ROS2 and Gazebo function using the same subscription path.
- Contact physics: The Contact was calculated by finding the distance between the center of the racket and the ball. However, while it was an easy way to do it, it introduced some errors if the ball was hit by the edge of the racket, the ball position estimator built up some latency and wouldn't catch the collision in time. With a stronger model using forward kinematics to accurately find the full racket surface position and continuous real world ball recording, collisions would be much more reliable. However, it is less prelevant in the tetherball experiment as only one contact is expected.
- Rope physics. The biggest problem of the environment is the rope dynamics. Touching the ball from the same position with the same velocity and same angle would never produce close to the same ball motion. This uncertainty makes ICL already failing from the start, as it would take a lot of demonstration to even understand how the rope-ball physics work. If it cannot understand how the ball will react, it is even less unlikely that it will understand how to hit the ball in a correct way. This proves once again that a correct set up would probably make the project much more accurate and close to a valid solution.

Chapter 6

DISCUSSION

DMP Framework Validation

The DMP framework used in this work was able to encode 3D end-effector motions into trajectories that can be replayed. The goal-invariant form helped combine different demonstrations and made it possible to reuse similar motions in slightly different conditions.

First Experiment Setup

However, for the first experiment, we had some slight limitation that required more in-depth understanding of how DMPs work and how it can be used for ICL. Several hits (3-4) were grouped into a single set of weights for each episode. This means one weight vector controls multiple hits with different timings. As a result, it becomes very hard to see how a change in one part of the weights relates to what happens to a specific hit.

Final Experiment Setup

For easy, one action tasks, the DMPs gave us perfect results, for the position or even the speed of the presentation, everything was the same, making it possible to run the episodes multiple times.

Limitation of In-Context Learning For the First Experiment

The first experiment started to break down during the In-Context Learning part and it gave us valuable insights into the limitations of episode-level DMPs encoding and for ICL. The fundamental issue faced was the **temporal correspondence problem**: using a single weight vector to encode multiple hits (3 to 4) with varying timing, making it impossible for the LLM to understand which weights index corresponded to which motion phases. When the same weight index controls different motion phases across the episode, the LLM cannot find a pattern to be used for the generated weights.

The key takeaway. apart from the urgent need to move the simulation to Mujoco, because of the physics problems in gazebo, is the need for temporal consistency in DMPs demonstrations. For ICL to work, each weight index must consistently control the same motion phase across all demonstrations. This led to two proposed solutions: either segment episodes into individual hits with fixed timing (Hit 1: [weights] - [outcome], Hit 2: [weights] - [outcome]), or normalize all hits within an episode to occur at identical time intervals. The principle is clear—LLMs require explicit, consistent structure to identify patterns and generate improvements.

Tetherball experiment Conclusion

This second experiment ended up pretty successful as showed in the results. However there were still some limitations faced. The most urgent solution would be to migrate from Gazebo to Mujoco, as the pros of using gazebo, which are how easy it is to go from the simulation to the robot, quickly were outweighed by the cons. Mujoco would allow the In-Context Learning iteration to be done automatically and thus

having up to a hundred iterations. In addition to the iteration, the rope physics are still a huge problem as it randomize slightly the ball trajectory, making it harder for the LLM to have successful hits. As liquids are possible to be generated in Mujoco, a simple rope is expected to work way better than in gazebo.

In addition to this quick fixes, implementing a way to record the ball and track it, in real life, would push the experiment to another dimension in term of results. It would allow for the LLM to have close to 0 environment error and making it focus on the policies only. This can be easily done by using a ball tracking software used on a camera, with the ball being attached to the robot, if we want to try again the Ball bouncing experiment, or to make an actual tetherball setup for the second experiment.

Conclusion

This thesis explored the usage of Dynamic Movement Primitives with In-Context Learning for robotic manipulation tasks, showing valuable insights into both the possibilities and limitations of this approach. The ball bouncing experiment revealed critical architectural constraints with LLMs requiring temporal consistency in DMP encoding to perform effective pattern matching. The tetherball experiment addressed this limitation by changing the setup and physics constraints while also addressing the limitation of DMPs by providing better temporal consistency a single weight vector executed repeatedly with predictable ball returns—enabling successful ICL-based weight refinement. The iterative feedback mechanism, proved, on a small scale, to improved the results in a way that would make it possible to maximize it with future iterations.

Despite the promising results, several limitations remain. The most pressing one is to migrate from Gazebo to Mujoco to address the physics limitations, particularly the rope dynamics, which introduced randomness that had an effect on the policy learning. Mujoco would also be able to automate 100+ iterations rather than relying on manual testing.

The key insight from this research is that In-Context Learning can be viable for DMP weight generation in robotics manipulation, with precise weight consistency to maximize the LLM capabilities. Future work should prioritize simulation improvements, automated iteration frameworks, and tasks with inherent temporal structure that naturally align with ICL’s strengths.

REFERENCES

- [1] “optitrack”, URL <https://optitrack.com/> (2025).
- [2] “Vicon”, URL <https://www.vicon.com/> (2025).
- [3] Brohan, A., Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian *et al.*, “Can foundation models perform zero-shot task specification for robot manipulation?”, arXiv preprint arXiv:2304.01907 (2023).
- [4] Brown, T., B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners”, Advances in neural information processing systems **33**, 1877–1901 (2020).
- [5] Chromy, A., P. Sopak and H. Cigler, “Validated low-cost standardized VICON configuration as a practical approach to estimating the minimal accuracy of a specific setup”, Scientific Reports **15**, 23351, URL <https://doi.org/10.1038/s41598-025-06111-9> (2025).
- [6] Huang, J., J. Gu, S. Garg *et al.*, “Understanding in-context learning from repetitions”, arXiv preprint arXiv:2402.08596 (2024).
- [7] Ijspeert, A. J., J. Nakanishi, H. Hoffmann, P. Pastor and S. Schaal, “Dynamical movement primitives: learning attractor models for motor behaviors”, Neural computation **25**, 2, 328–373 (2013).
- [8] Ijspeert, A. J., J. Nakanishi and S. Schaal, “Movement imitation with nonlinear dynamical systems in humanoid robots”, in “Proceedings 2002 IEEE International Conference on Robotics and Automation”, vol. 2, pp. 1398–1403 (IEEE, 2002).
- [9] Kaup, M., C. Wolff, H. Hwang, J. Mayer and E. Bruni, “A review of nine physics engines for reinforcement learning research”, arXiv preprint arXiv:2407.08590 URL <https://arxiv.org/abs/2407.08590> (2024).
- [10] Paraschos, A., C. Daniel, J. R. Peters and G. Neumann, “Probabilistic movement primitives”, Advances in neural information processing systems **26** (2013).
- [11] Schaal, S., A. Ijspeert and A. Billard, “Computational approaches to motor learning by imitation”, in “Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences”, vol. 358, pp. 537–547 (2003).
- [12] Schroeder, S., S. Jaeger, J. Schwer, A. M. Seitz, I. Hamann, M. Werner, C. Thorwächter, I. Santos, T. Wendler, D. Nebel and B. Welke, “Accuracy measurement of different marker based motion analysis systems for biomechanical applications: A round robin study”, PLOS ONE **17**, 7, e0271349 (2022).

- [13] Wake, N., A. Kanehira, K. Sasabuchi, J. Takamatsu and K. Ikeuchi, “Gpt-4v (ision) for robotics: Multimodal task planning from human demonstration”, arXiv preprint arXiv:2311.12015 (2023).
- [14] Windolf, M., N. Götzen and M. Morlock, “Systematic accuracy and precision analysis of video motion capturing systems—exemplified on the Vicon-460 system”, *Journal of Biomechanics* **41**, 12, 2776–2780, pMID: 18672241 (2008).
- [15] Zhou, Y., S. Grover, M. El Mistiri, K. Kalirathinam, P. Kerhalkar, S. Mishra, N. Kumar, S. Gaurav, O. Aran and H. Ben Amor, “Prompted policy search: Reinforcement learning through linguistic and numerical reasoning in LLMs”, in “Proceedings of the Thirty-Ninth Annual Conference on Neural Information Processing Systems (NeurIPS 2025)”, (2025), URL <https://openreview.net/forum?id=95plu1Mo20>, openReview preprint.