ACCELERATING ROBOT LEARNING VIA HUMAN-IN-THE-LOOP SHARED CONTROL

A Thesis

by

Deniz Yılmaz

Submitted to the

Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Computer Science

Özyeğin University
January 2024

ACCELERATING ROBOT LEARNING VIA HUMAN-IN-THE-LOOP SHARED CONTROL

Advisor: Assoc Prof. Barkan Uğurlu, Özyeğin University
Coadvisor: Prof. Erhan Öztop, Özyeğin University

Approved by:

Assoc Prof. Barkan Uğurlu, Advisor
Dept. of Mechanical Eng.
Özyeğin University

Asst. Prof. Reyhan Aydoğan
Dept. of Computer Science
Özyeğin University

Prof. Dr. Duygun Erol Barkana
Dept. of Electrical & Electronics Eng.
Yeditepe University

Date Approved: January 5, 2024

To my niece, Sara

ABSTRACT

This thesis presents a training procedure for reinforcement learning-based robot control to achieve skill acquisition with reduced training time and data requirements. It achieves this by actively incorporating the actions provided via human inputs in addition to actions generated by the learning algorithm. As the algorithm acquires the target skill, the contribution of the human actions is gradually decreased for autonomous task execution. To demonstrate the efficacy of the proposed approach, a ball-balancing task with manipulability index maximization was chosen. In this task, a 7-DoF robot arm achieved skill acquisition via reinforcement learning in which two distinct training approaches were employed: i)autonomous training, and ii) human-in-the-loop training approach. The task required the robot to bring the ball to the center of a tray attached to its end-effector, starting from arbitrary initial ball positions in the face of perturbations. Simulation experiments were carried out with 24 human participants where each participant guided robot learning in a human-in-the-loop shared control setting. Compared to autonomous training, the human-in-the-loop training approach showed superior performance in terms of reduced training time and data usage while exhibiting favorable ball-balancing skills.

ÖZETÇE

Bu tez, eğitim süresini düşürerek ve veri gereksinimlerini azaltarak insandan robota beceri kazanmayı amaçlayan öğrenmeye dayalı robot kontrolü için bir eğitim süreci sunmaktadır. Bu eğitim prosedürü, öğrenme algoritması tarafından üretilen komutlara ek olarak, insanın döngüye katılmasıyla sağlanan komutları etkin bir şekilde kullanarak eğitim sürecinin hızlanmasını sağlar. Algoritma hedef beceriyi kazandıkça insan eylemlerinin katkısı, görevin otonom gerçekleştirilebilmesi için kademeli olarak azaltılmıştır. Önerilen yaklaşımın etkisini gözlemlemek için, manipulabilite indeksinin iyileştirilmesi gözetilerek robotun uç işlevcisinde top dengeleme görevi seçilmiştir. Bu görevde, 7 serbestlik dereceli bir robot kolun öğrenme yoluyla beceri kazanmasını sağlamak için iki farklı eğitim yaklaşımı kullanılmıştır: i)otonom eğitim, ve ii) insanın eğitim döngüsüne dahil olduğu yaklaşım. Görev, robotun topu, rastgele belirlenmemiş pozisyonlarından başlayarak, robotun uç işlevcisine bağlı bir tepside dengelemesini gerektirmektedir. Otonom eğitime benzer şekilde, önerilen yaklaşım için 24 katılımcı ile benzetim çalışmaları gerçekleştirilmiştir. Her bir katılımcı robotu, insanın dahil olduğu paylaşımlı kontrol ortamında yönlendirmiştir. Otonom eğitimle karşılaştırıldığında, insanın dahil olduğu eğitim yaklaşımı, eğitim süresinin ve veri kullanımının düşürülmesini sağlayacak şekilde olumlu bir performans göstermiş ve top dengeleme becerisini başarılı bir şekilde sergilemiştir.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to all those who have contributed to the completion of this thesis.

First and foremost, I extend my sincere appreciation to my thesis advisors, Barkan Uğurlu and Erhan Öztop, for their invaluable guidance, unwavering support, and insightful feedback throughout the entire research process. Their expertise and encouragement played a pivotal role in shaping the direction and quality of this work.

I am grateful to my family and friends for their unwavering support, understanding, and encouragement. Their patience and belief in my abilities were constant sources of inspiration.

Special thanks go to my labmates Burak Özkaynak and Erim Can Özçınar for their camaraderie and collaborative spirit. The exchange of ideas and shared experiences greatly contributed to the overall learning and research environment.

Finally, I want to express my gratitude to Özyeğin University for providing educational support for this research. Their support made it possible for me to dedicate time and resources to the successful completion of this thesis.

This journey would not have been possible without the collective support and encouragement of these individuals and institutions. Thank you for being an integral part of this academic journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

List of Symbols

$\boldsymbol{q}$	Joint Angle Vector
$\boldsymbol{q_{max}}$	Maximum Joint Angle
$\boldsymbol{q_{min}}$	Minimum Joint Angle
$\boldsymbol{\dot{q}}$	Joint Velocity Vector
$\boldsymbol{\dot{q}_{max}}$	Maximum Joint Velocity
$\boldsymbol{\ddot{q}}$	Joint Acceleration Vector
$\boldsymbol{\ddot{q}_{max}}$	Maximum Joint Acceleration
$\boldsymbol{T}$	Transformation Matrix
$\boldsymbol{T_{OE}}$	End-Effector Transformation Matrix
$\boldsymbol{\tau}$	Torque
$\boldsymbol{\tau_{control}}$	PD Controller Torque
$\boldsymbol{\tau_g}$	Nonlineartiy Torque
$\boldsymbol{\tau_{ppo}}$	PPO Torques
$\boldsymbol{\tau_{cmd}}$	Command Torque
$\boldsymbol{\tau_{max}}$	Maximum Torque
$\boldsymbol{R}$	Rotation Matrix
$\boldsymbol{J}$	Jacobian Matrix

$\boldsymbol{J_r}$	Rotational Jacobian Matrix
$\boldsymbol{J_t}$	Translational Jacobian Matrix
$\boldsymbol{P_x}$	End Effector Position in x-axis
$\boldsymbol{P_y}$	End Effector Position in y-axis
$\boldsymbol{P_z}$	End Effector Position in z-axis
$\dot{\boldsymbol{x}}$	Linear Velocity in x-axis
$\dot{\boldsymbol{y}}$	Linear Velocity in y-axis
$\dot{\boldsymbol{z}}$	Linear Velocity in z-axis
$\boldsymbol{\omega_x}$	Angular Velocity in x-axis
$\boldsymbol{\omega_y}$	Angular Velocity in y-axis
$\boldsymbol{\omega_z}$	Angular Velocity in z-axis
$\dot{\boldsymbol{v}}$	Linear Acceleration Vector
$\dot{\boldsymbol{\omega}}$	Angular Acceleration Vector
$\boldsymbol{M(q)}$	Mass Matrix
$\boldsymbol{C(q,\dot{q})}$	Coriolis and Centrifugal Terms
$\boldsymbol{G(q)}$	Vector of Gravitational Forces
$\boldsymbol{F_e}$	External Forces
$\boldsymbol{M_e}$	External Moments
$\hat{\boldsymbol{z}}$	Unit Vector

$\boldsymbol{I}$	Inertia Tensor
$\boldsymbol{V_e}$	End Effector Twist
$\boldsymbol{f}$	Frequency of Cosine Wave
$\boldsymbol{A}$	Amplitude of Cosine Wave
$\boldsymbol{K_p}$	Proportional Control Gain Matrix
$\boldsymbol{K_d}$	Derivative Control Gain Matrix
$\boldsymbol{s}$	Current State
$\boldsymbol{s'}$	Expected Next State
$\boldsymbol{a}$	Action
γ	Discount Factor
$\boldsymbol{\pi}$	Policy
$\boldsymbol{V^{\pi}(s)}$	Value Function
$\boldsymbol{R_{\alpha}}$	Reward Penalty - Orientation
$\boldsymbol{\omega_{\alpha}}$	Weight Constant of Reward Penalty - Orientation
$\boldsymbol{\alpha}$	End Effector Pitch Orientation
$\boldsymbol{R_V}$	Reward Penalty - Velocity
$\boldsymbol{\omega_V}$	Weight Constant of Reward Penalty - Velocity
$\boldsymbol{e_{V_x}}$	End Effector Joint Velocity Error
$\boldsymbol{e_x}$	End Effector Joint Position Error

$\|\boldsymbol{e_x}\|$	Norm of the End Effector Joint Position Error
$\boldsymbol{marg_x}$	Joint Position Penalty Term
$\boldsymbol{\omega_{marg}}$	Weight Constant of Joint Position Penalty Term
$\boldsymbol{R_{kw}}$	Reward - Manipulability
$\boldsymbol{k_w}$	Manipulability Index
$\boldsymbol{\omega_{k_w}}$	Weight Constant of Reward - Manipulability
$\boldsymbol{R_P}$	Reward Penalty - Position
$\boldsymbol{\omega_P}$	Weight Constant of Reward Penalty - Position
$\boldsymbol{R_{P_e}}$	Reward - Position
$\boldsymbol{\omega_{P_e}}$	Weight Constant of Reward - Position
$\boldsymbol{r}$	Overall Reward Function
$\boldsymbol{M_y}$	Pitch Axis Cartesian Moment
$\boldsymbol{M_{yai}}$	Pitch Axis Cartesian Moment from Learning Agent
$\boldsymbol{M_{yh}}$	Pitch Axis Cartesian Moment from Human
$\boldsymbol{W_{tr}}$	Cartesian End Effector Wrench

CHAPTER I

INTRODUCTION

In the realm of contemporary technology, the integration of robotics and reinforcement learning emerges as a pivotal force, shaping the landscape of intelligent systems. Robotics, with its capacity for automated task execution and physical interaction, has transcended traditional industrial applications to permeate various facets of our daily lives. Simultaneously, reinforcement learning, a subset of machine learning, empowers systems to learn and make decisions through interaction with their environment. The amalgamation of these two domains not only propels automation to new heights but also heralds a paradigm shift in how machines acquire and apply knowledge. As we navigate this era of rapid technological advancement, understanding the symbiotic relationship between robotics and reinforcement learning becomes crucial for unlocking the full potential of intelligent and adaptive systems.

1.1 Literature Review and Problem Statement

The area of influence of robots in the world is growing as they become increasingly usable and efficient thanks to ongoing research on human-robot collaboration [2]. Many studies at the intersection of reinforcement learning and robotics focus on human-robot shared control, where humans and robots collaborate to perform tasks, ensuring both safety and efficiency [3]. Additionally, robots can learn complex tasks by using data collected from humans, such as commands and directions, through the use of imitation learning [4] and interactive learning [5].

A reinforcement learning agent aims to build a policy by learning and interacting with the environment, enabling it to successfully perform the target task upon the completion of learning process. During the learning phase, the agent explores

action possibilities while experiencing different states and rewards delivered by the environment. In robotic applications, states could be the joint variables whereas the reward function is defined by the designer to encode the task goals. To this end, time and data are of great importance as an efficiency aspect when implementing reinforcement learning. A major limitation of current reinforcement learning algorithms remains in its sample inefficiency where the required number of interactions with the environment may be impractically high [6].

In addition to sample inefficiency issues, reinforcement learning-based could control possess several challenges such as unguaranteed stability, reward shaping, data requirements [7]. To mitigate these challenges, researchers suggested the incorporation of human expertise into the teaching of complex tasks such that robots can effectively acquire advanced skills [8]. This approach has encouraged further research into human-in-the-loop learning, interactive learning, and imitation learning. Reinforcement learning alone, cannot replace the expertise of humans in specific domains, making the human-in-the-loop learning method increasingly popular. By leveraging human knowledge and experience, these methods aim to train prediction models at minimal cost [9].

In particular, the learning process can take thousands of episodes and hours of training time, depending on the characteristics of the environment and task [10]. In response to this matter, providing human guidance to the robot may reduce the success/failure attempts made by the agent during the training process and could facilitate favorable exploration [9]. This approach is known as human-centered reinforcement learning, interactive learning, or human-in-the-loop learning, as discussed in this thesis.

There are several studies on human-in-the-loop learning to make use of its advantages. For instance, an application of human guidance was demonstrated with

a humanoid Nao robot for a pick-and-place task [11]. In another study, human responses were used for policy-shaping, leading to successful outcomes [12]. Following a similar strategy, Arakawa et al. incorporated both human feedback and distant rewards which leveraged people's facial expressions to guide learning [13]. Recently, Luo et al. performed a study where they showed that the human-in-the-loop approach relatively increases learning speed and performance specifically in continuous action spaces [14].

Human-in-the-loop learning was also successfully implemented in physical human-robot interaction applications, for instance, Peternel et al. combined human-in-the-loop learning with the impedance control for haptic interfaces [15]. In another study, they explored transitioning from a tutor's action policy for robot learning to achieve full autonomy over time, with a focus on understanding the contributions of humans to learning processes [16]. These studies pave the way for more advanced collaborations between humans and robots, shaping a future where the teamwork between human skills and robotic abilities continues to push the boundaries of interactive technologies.

In the light of these studies on the involvement of humans in the training loop of reinforcement learning applied in the field of robotics. Each of these studies aims to generate solutions for challenges in reinforcement learning-based control. While useful in their application domains, most learning-based control approaches dictate the need to learn the whole robot dynamics from scratch, as the produced actions are usually passed as open-loop torque commands [17]. These findings highlight the ongoing work to improve how reinforcement learning is used in robotics, aiming for more effective solutions in the future.

1.2 Contributions

To remedy training period issues, in this thesis, we utilized a dynamics-incorporated learning-based control algorithm in which the actions were passed as feed-forward torques within a computed torque control scheme, leveraging skill acquisition during the training period. In doing so, the corrective balancing actions from both agents are passed as external moments applied to the end effector, thus, enabling dynamically consistent skill acquisition without having to learn the whole body dynamics of the system. As a result, it leads to much shorter training time and we demonstrated significant reductions in training time. Moreover, the manipulability index was maximized to exploit the redundancy for dexterous skill acquisition.

As a use case scenario, a fundamental ball balancing task was performed using Franka Emika's Panda robot model in a simulation setting [18]. The main motivation behind this task choice was to leverage complexity and training data requirements to explicitly observe whether the proposed training approach could provide significant reductions in training time. Two distinct training strategies were employed while using the PPO (Proximal Policy Optimization) algorithm: i) *Autonomous training* approach, where the robot learns the desired task without any intervention. ii) *Human-in-the-loop training* approach, where a human tutor's commands were incorporated into the environment using a keyboard. With the proposed human-in-the-loop approach the contributions of this thesis are as follows:

- Introducing a learning-based control algorithm that facilitates dynamically consistent skill acquisition without the requirement to learn the entire body dynamics of the system.
- Developed a control scheme that enhances the PPO algorithm through smoother exploration with human involvement, intervening with continuous actions in the training loop.

- Proposed human-in-the-loop approach and reduced training time by 97.6% compared to autonomous training within the scope of the specified task, thus achieving faster skill acquisition and learning of more robust policies.
- Designed a human-in-the-loop learning approach, resulting in the achievement of better regulation in the training process.

1.3 Organization

The remainder of this thesis is structured as follows:

In Chapter II, the derivation of the kinematic and dynamic model of Franka Emika's Panda is explained in detail. Panda's joint configuration is introduced, followed by the derivation of the forward kinematics and velocity kinematics that are used in the test procedure of the trained models. Chapter II also provides the methods used to derive the dynamics model of Panda, which is utilized for feedback linearization.

In Chapter III, the human-robot shared control method is introduced. The specified task is explained. Two different approaches are covered: autonomous training and the human-in-the-loop training approach.

In Chapter IV, the simulation results are introduced. The performance assessment of the training procedures of these two approaches is presented and the relevant results are discussed.

Finally, in Chapter V, the conclusions drawn from the findings are discussed, and potential directions for future research are outlined.

CHAPTER II

FRANKA EMIKA'S PANDA KINEMATICS AND DYNAMICS

2.1 Robot Model of the Franka Emika's Panda

The work proposed in this thesis was implemented using Franka Emika's Panda robot model [19]. It was simulated in Raisim[18], a dynamic simulation program for articulated systems. Panda is a 7-degree-of-freedom robot with torque-controllable joints. The joint configuration of the Panda is shown in Fig. 1. Panda has a 3 kg payload capacity. The repeatability of the Panda robot is 0.1 mm and the robot's weight is approximately 18 kg. The joint space limits of the Panda which are adjusted to the URDF (Unified Robotics Description Format), are tabulated in Table 1.

Since the specific task in this thesis is to balance the ball a the tray placed at the end effector, a rectangular prism with dimensions of 40 cm x 10 cm was designed and assembled into Panda see Fig. 2.

Table 1: Panda's Joint Space Limits

$Name$	$Joint-1$	$Joint-2$	$Joint-3$	$Joint-4$	$Joint-5$	$Joint-6$	$Joint-7$	$Unit$
q_{max}	2.8973	1.7628	2.8973	-0.0698	2.8973	3.7525	2.8973	rad
q_{min}	-2.8973	-1.7628	-2.8973	-3.0718	-2.8973	-0.0175	-2.8973	rad
$\dot{q}_{max}$	2.1750	2.1750	2.1750	2.1750	2.6100	2.6100	2.6100	rad/s
$\ddot{q}_{max}$	15	7.5	10	12.5	15	20	20	rad/s^2
$T_{j_{max}}$	87	87	87	87	12	12	12	Nm

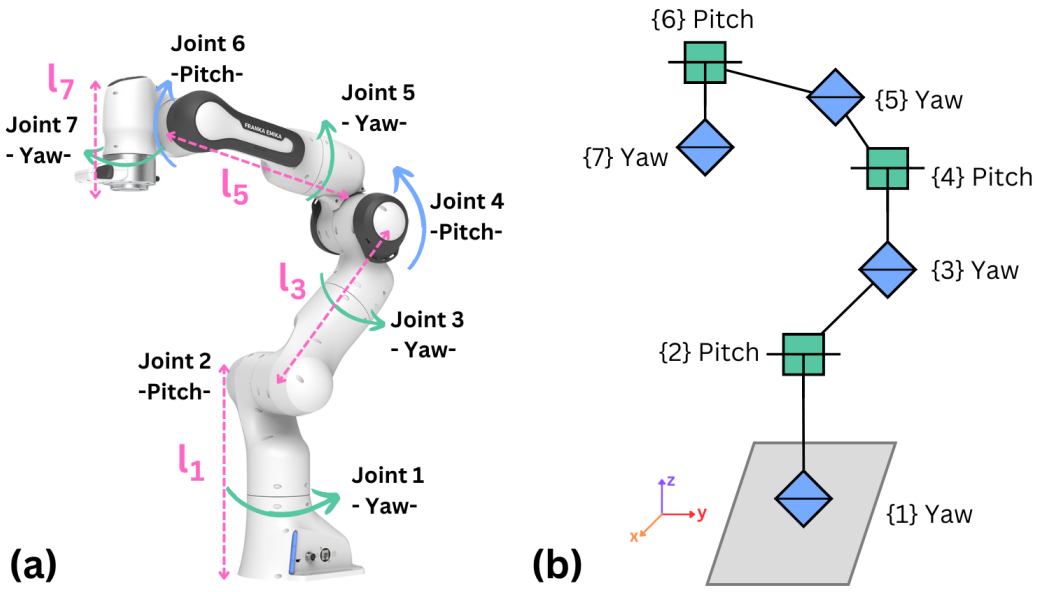

Figure 1: Joint configuration of the Franka Emika's Panda [1]

Figure 2: Franka Emika's Panda in Raisim simulation environment

2.2 Panda Kinematics

The robot kinematics demonstrate an effective mapping between Cartesian coordinates, representing the position and orientation in space, and joint space coordinates, which describe the configuration of the robot's joints. Studies were carried out to derive the kinematic models of Panda. For this task, we utilized the methodology outlined in [20].

2.2.1 Forward Kinematics

To derive Panda's forward kinematics using the joint variables in Table 2, the transformation matrix for each joint was found as shown below. In the transformation matrices, R is the rotation matrix and P_x, P_y, P_z is the joint positions.

$$T_1^0 = \begin{bmatrix} \cos(q_1) & -\sin(q_1) & 0 & 0 \\ \sin(q_1) & \cos(q_1) & 0 & 0 \\ 0 & 0 & 1 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$T_2^1 = \begin{bmatrix} \cos(q_2) & 0 & \sin(q_2) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(q_2) & 0 & \cos(q_2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$T_3^2 = \begin{bmatrix} \cos(q_3) & -\sin(q_3) & 0 & 0 \\ \sin(q_3) & \cos(q_3) & 0 & 0 \\ 0 & 0 & 1 & l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$T_4^3 = \begin{bmatrix} \cos(q_4) & 0 & \sin(q_4) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(q_4) & 0 & \cos(q_4) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$T_5^4 = \begin{bmatrix} \cos(q_5) & -\sin(q_5) & 0 & 0 \\ \sin(q_5) & \cos(q_5) & 0 & 0 \\ 0 & 0 & 1 & l_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$T_6^5 = \begin{bmatrix} \cos(q_6) & 0 & \sin(q_6) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(q_6) & 0 & \cos(q_6) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

$$T_7^6 = \begin{bmatrix} \cos(q_7) & -\sin(q_7) & 0 & 0 \\ \sin(q_7) & \cos(q_7) & 0 & 0 \\ 0 & 0 & 1 & l_7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

Table 2: Panda's Frames and Link Lengths

$Frame$	$JointRotationAxis$	$LinkLengths(m)$
1	Yaw	$l_1 = 0.333$
2	$Pitch$	$l_2 = 0$
3	Yaw	$l_3 = 0.316$
4	$Pitch$	$l_4 = 0$
5	Yaw	$l_5 = 0.384$
6	$Pitch$	$l_6 = 0$
7	Yaw	$l_7 = 0.107$

The transformation matrix of the end effector T_{OE} derived as in (8). The rotation matrix and position vector of the end effector were obtained from the resulting transformation matrix T_{OE} as in (9) where $T_{OE} \in \mathbb{R}^{4\times4}$.

$$T_{OE} = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 T_7^6 \tag{8}$$

$$T_{OE} = \begin{bmatrix} Re_{11} & Re_{12} & Re_{13} & Pe_x \\ Re_{21} & Re_{22} & Re_{23} & Pe_y \\ Re_{31} & Re_{32} & Re_{33} & Pe_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

2.2.2 Velocity Kinematics

The Jacobian provides the relation between joint velocities and end effector velocities of a robot manipulator as shown below.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} & \cdots & J_{17} \\ J_{21} & J_{22} & \cdots & J_{27} \\ \vdots & \vdots & \ddots & \vdots \\ J_{61} & J_{62} & \cdots & J_{67} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_7 \end{bmatrix} \tag{10}$$

where $\dot{x}, \dot{y}, \dot{z}$ represents linear velocities of the end effector, $\omega_x, \omega_y, \omega_z$ represents angular velocities of the end effector frame, $\dot{q}$ represents the joint velocities and J is the Jacobian.

End effector Jacobian is divided into two components: the translational Jacobian $J_t(q)$ and the rotational Jacobian $J_r(q)$. The translational Jacobian $J_t(q)$, establishes the relationship between the derivatives of the end-effector's cartesian coordinates x, y, z and the joint positions $q_1, q_2, \ldots, q_7$. The rotational Jacobian $J_r(q)$, illustrates

the impact of joint velocities $\dot{q_1}, \dot{q_2}, \ldots, \dot{q_7}$ on the end-effector's orientation through partial derivatives of the angular velocity vector ω with respect to the joint variables as shown in below.

$$J_t(q) = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \cdots & \frac{\partial x}{\partial q_7} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \cdots & \frac{\partial y}{\partial q_7} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \cdots & \frac{\partial z}{\partial q_7} \end{bmatrix} \tag{11}$$

$$J_r(q) = \begin{bmatrix} \frac{\partial \omega_1}{\partial \dot{q}_1} & \frac{\partial \omega_1}{\partial \dot{q}_2} & \cdots & \frac{\partial \omega_1}{\partial \dot{q}_7} \\ \frac{\partial \omega_2}{\partial \dot{q}_1} & \frac{\partial \omega_2}{\partial \dot{q}_2} & \cdots & \frac{\partial \omega_2}{\partial \dot{q}_7} \\ \frac{\partial \omega_3}{\partial \dot{q}_1} & \frac{\partial \omega_3}{\partial \dot{q}_2} & \cdots & \frac{\partial \omega_3}{\partial \dot{q}_7} \end{bmatrix} \tag{12}$$

$$\omega = \frac{dR}{dt} R^T \tag{13}$$

To obtain the translational Jacobian $J_t(q)$ and rotational Jacobian $J_r(q)$, as represented in (11) and (12), the end-effector position and rotation matrix elements of Panda are utilized. These elements are derived from the transformation matrix T_{OE} obtained in the (9). The overall Jacobian matrix $J(q)$ is shown in (14) where $J(q) \in \mathbb{R}^{6\times7}$.

$$J(q) = \begin{bmatrix} J_t(q) \\ J_r(q) \end{bmatrix} \tag{14}$$

2.3 Panda Dynamics

The dynamic model of Panda was studied to be used in calculating the torque value transmitted as an action command to create the desired movement in the specified task. Franka's inverse and forward dynamics model was examined in a MATLAB environment. The recursive Newton-Euler algorithm (RNEA) used by Raisim was examined. Using the RNEA, Panda's Coriolis and the gravitational term were obtained

and added to the overall torque command sent to Panda throughout the simulation as non-linearities.

2.3.1 Equation of Motion

The equation of motion in robot dynamics describes the relationship between the forces and torques acting on a robot. The equation of motion is expressed as shown below.

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) - J^{\intercal}\begin{pmatrix} F_e \\ M_e \end{pmatrix} \tag{15}$$

where $M(q)$ is the mass matrix, $C(q,\dot{q})$ is matrix of Coriolis and centrifugal terms, $G(q)$ is the vector of gravitational forces and F_e and M_e are the external force and moment.

The mass matrix $M(q)$ represents the mass matrix at a particular configuration q. It characterizes how the robot joints respond to variations in joint acceleration. The matrix of Coriolis and centrifugal terms $C(q,\dot{q})$, represents the effects of joint velocities $\dot{q}$ on the dynamics and the vector of gravitational forces $G(q)$ representing the effects of gravity on the robot.

2.3.2 Recursive Newton-Euler Algorithm (RNEA)

The Raisim simulation environment used in the studies uses the recursive Newton-Euler algorithm to derive a robot dynamics model. The Recursive Newton-Euler algorithm is an approach used to compute the forward dynamics of a robot. This method applies Newtonian mechanics and Euler's rotational equations in a recursive manner to determine the dynamic characteristics of a robot throughout its kinematic chain.[21]

The algorithm consists of two recursive computations Forward Recursion (Bottom-Up Pass) and Backward Recursion (Top-Down Pass). The forward recursion, from the

first link to the nth link, calculates the velocities and accelerations of each link, leading to the determination of the dynamic forces applied to each link. Conversely, the backward recursion, starting from the nth link and returning to the base, establishes the reactive forces on the links and subsequently the torques at the joints [22]. This approach expresses the joint torques based on the joint positions, velocities, and accelerations, without the explicit computation of matrices in the equation of motion. The relevant algorithm can be simply expressed as a function shown in (16).

$$\tau = NewtonEuler(q, \dot{q}, \ddot{q}, f_e, m_e) \tag{16}$$

When these variables are provided as input to the RNEA algorithm, the required joint torques for the given trajectories can be computed.

In the forward recursion part of the algorithm, starting from the end effector and progressing towards the base, the algorithm computes the linear $\dot{v}_i$ and angular $\dot{\omega}_i$ accelerations for each link within the robot's kinematic chain. Linear acceleration and angular acceleration recursive formulae are shown below.

$$\dot{v}_i = \dot{v}_{i-1} + \ddot{q}_i \times r_{i-1,i} + \omega_{i-1} \times (\omega_{i-1} \times r_{i-1,i}) \tag{17}$$

where $r_{i-1,i}$ is the distance vector from the center of mass of link i to the joint axis, and

$$\dot{\omega}_i = \dot{\omega}_{i-1} + \ddot{q}_i \times \hat{z}_{i-1} + \omega_{i-1} \times \dot{q}_i \times \hat{z}_{i-1} \tag{18}$$

where $\hat{z}_{i-1}$ is the unit vector along the joint axis.

In the backward recursion part of the algorithm, the algorithm computes the joint forces $f_{i,\text{coriolis}}, f_{i,\text{gravity}}$ and torques τ_i for each joint.

$$f_{i,\text{gravity}} = m_i g - m_i \dot{v}_i \tag{19}$$

where g is the acceleration due to gravity.

$$f_{i,\text{coriolis}} = -\omega_{i-1} \times (\omega_{i-1} \times m_i c_{i-1,i}) - 2\omega_{i-1} \times m_i \dot{c}_{i-1,i} \tag{20}$$

where $c_{i-1,i}$ is the center of mass location of link i relative to joint $i-1$.

Following the computation of forces and torques acting upon individual joints, joint torques are determined using inward iterations. The torque acting on each joint τ_i can be derived as follows:

$$\tau_i = f_{i,\text{gravity}} \hat{z}_{i-1} + f_{i,\text{coriolis}} \hat{z}_{i-1} + I_i \ddot{q}_i + \omega_i \times I_i \omega_i + \dot{q}_i \times (I_i \omega_i) \tag{21}$$

where I_i is the inertia tensor of the i^{th} link.

2.4 Kinematics and Dynamics Validation

To validate the kinematic and dynamic models of the Panda robot, the relevant derivations were tested in the Raisim simulation environment and the obtained results are discussed.

In order to validate the velocity kinematics, the joint velocity reference, calculated using the derived Jacobian and twist vector, was given as a reference with the torque command and subsequently compared with the measured velocity value of the end effector.

The end-effector twist vector V_e, is given by the product of the Jacobian matrix $J(q)$, and the joint velocity vector $\dot{q}$. To obtain the joint velocity reference $\dot{q}$, to be given to Panda's all seven joints, the twist vector V_e was multiplied by the inverse of the Jacobian matrix $J^{-1}(q)$ as in (23).

$$V_e = J(q)\dot{q} \tag{22}$$

$$\dot{q} = J^{+}(q)V_e \tag{23}$$

where,

$$V_e = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & \omega_x & \omega_y & \omega_z \end{bmatrix}^T \tag{24}$$

As a reference, a cosine wave with three different frequencies of 0.25 Hz, 0.50 Hz, and 1 Hz was given to linear velocity along the x-axis $\dot{x}$, in the twist vector of the end effector. The reference signal for the linear velocity along the x-axis is given by:

$$\dot{x}_{\text{ref}}(t) = A\omega_x \cos(\omega_x t) \tag{25}$$

where, A is the amplitude of the wave, ω_x is the angular frequency, and $\cos(\omega_x t)$ represents the oscillating behaviour over time. In the specific implementation, the amplitude A is set to 0.05, and the angular frequency (ω_x) is calculated as $2\pi \times f$ where f is the frequency. These reference signals are used to modulate the linear velocity along the x-axis in the time domain.

$$V_e = \begin{bmatrix} \dot{x}_{\text{ref}}(t) & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{26}$$

The reference joint velocity $\dot{q}$ was obtained by multiplying the reference end effector twist vector V_e and the inverse of the Jacobian matrix as in (25).

Using the relevant joint velocity and joint position reference, the torque command was calculated and sent to Panda in the Raisim environment. The end effector's linear velocity in the x-axis was measured and compared with the given reference values to check the correction of the Jacobian matrix derived in the velocity kinematics section.

$$\tau_{\text{control}} = K_p(q_{ref} - q) + K_d(\dot{q}_{ref} - \dot{q}) \tag{27}$$

where q is joint position, $\dot{q}$ is joint velocity and K_d,K_d are the control gains.

In addition to velocity kinematics, in terms of Panda's dynamics model, non-linearity term τ_{g} obtained from Raisim were added to the torque value τ_{pd} calculated

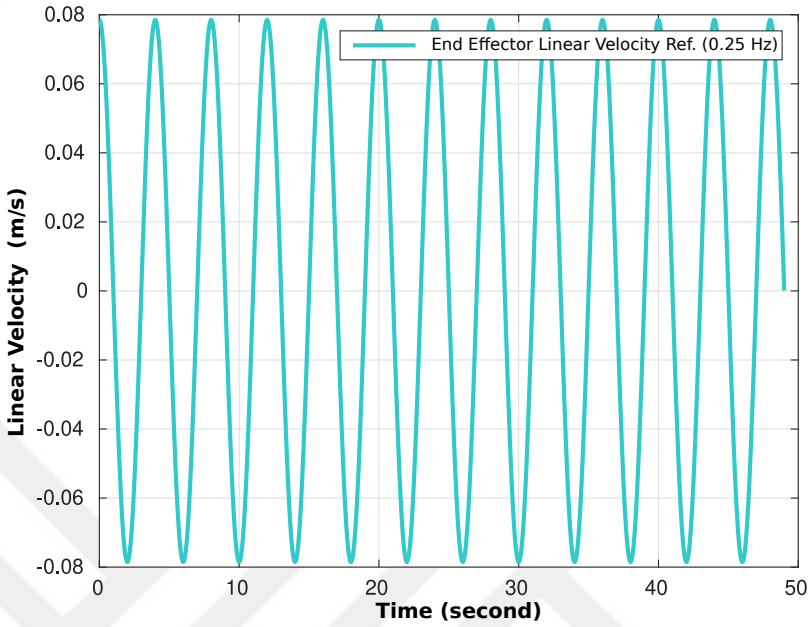

Figure 3: Linear velocity reference - 0.25 Hz

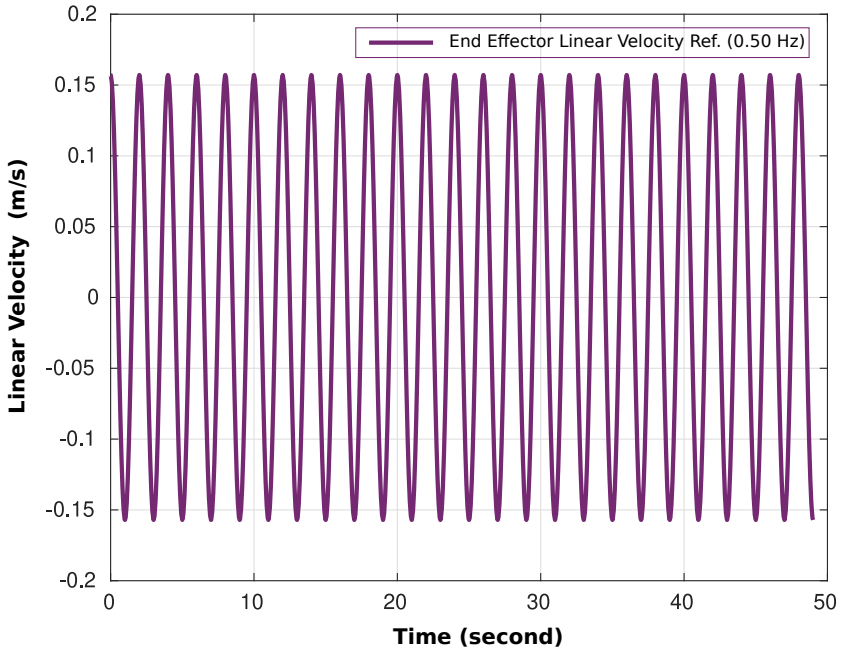

Figure 4: Linear velocity reference - 0.50 Hz

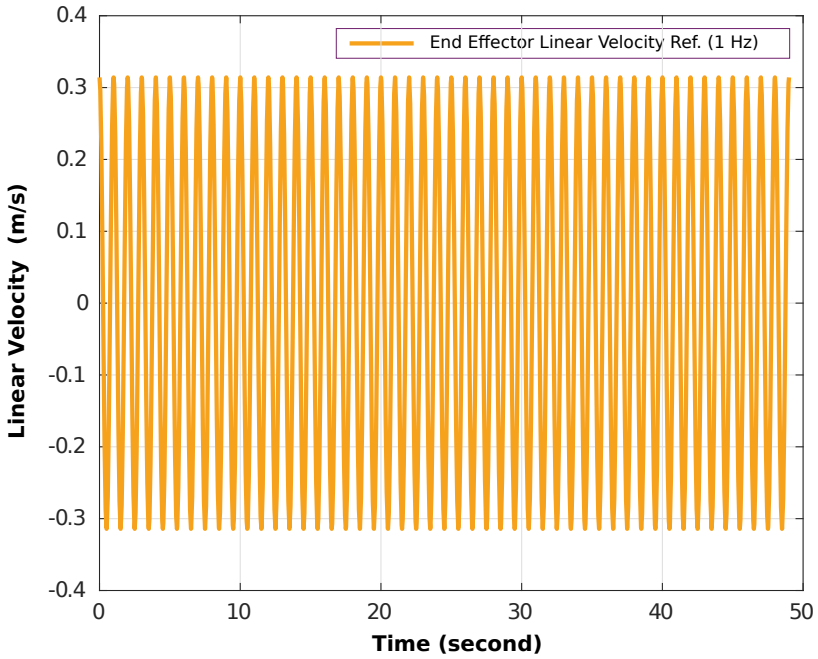

Figure 5: Linear velocity reference - 1 Hz

in (27) to validate and see the effect of the Coriolis and the gravitational term derived with the Recursive Newton-Euler algorithm.

$$\tau_{\mathrm{g}} = C(\mathbf{q}, \dot{\mathbf{q}}) + G(\mathbf{q}) \tag{28}$$

$$\tau_{\mathrm{cmd}} = \tau_{\mathrm{pd}} + \tau_{\mathrm{g}} \tag{29}$$

Velocity tracking tests were performed with and without the addition of the non-linearity term to the commanded torque value as scenario-1 and scenario-2. The tests were performed with three different frequency values as illustrated in Figs. 3, 4, 5 for both scenarios.

Tracking test was performed using the high control gains shown in Table - 3. For all three frequency values, in both scenarios, the reference linear velocity given to the end effector was tracked with the high gains as shown in Figs. 6, 7, 8. Purple lines show the measured linear velocity and green dashed lines show reference value. The

Table 3: Proportional - Derivative Controller Gains

Joint	$Kp_{\mathbf{high}}$	$Kd_{\mathbf{high}}$	$Kp_{\mathbf{low}}$	$Kd_{\mathbf{low}}$
Joint - 1	200	450	35	85
Joint - 2	1400	3150	245	595
Joint - 3	200	450	35	85
Joint - 4	1400	3150	245	595
Joint - 5	200	450	35	85
Joint - 6	200	450	35	85
Joint - 7	200	450	35	85

relative matching of the lines indicates the correct derivation of the Jacobian matrix.

Although both scenarios were successful in the velocity tracking with high control gains, the scenarios were re-run using low control gains shown in Table - 3, in order to observe the effect of non-linearity terms derived from Panda's dynamic model. As seen in Figs. 9, 10, 11, scenario-1 showed a better tracking performance than scenario-2 at all frequency conditions. Red lines show the measured linear velocity and green dashed lines show reference value. In light of the results obtained, the non-linearity term was included when calculating the torque in the remainder of the studies.

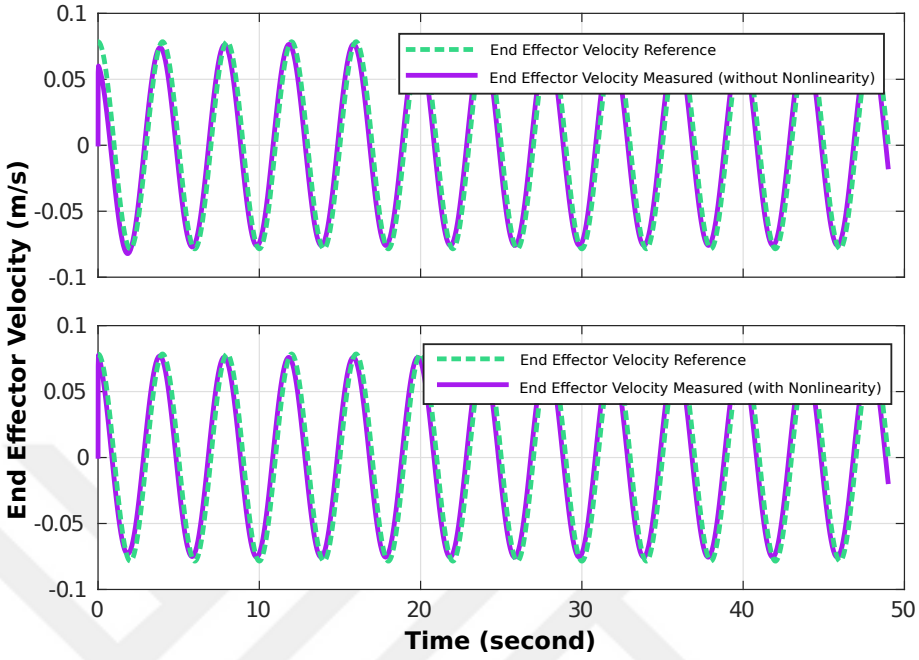

Figure 6: Joint velocity tracking error with high control gains - 0.25 Hz

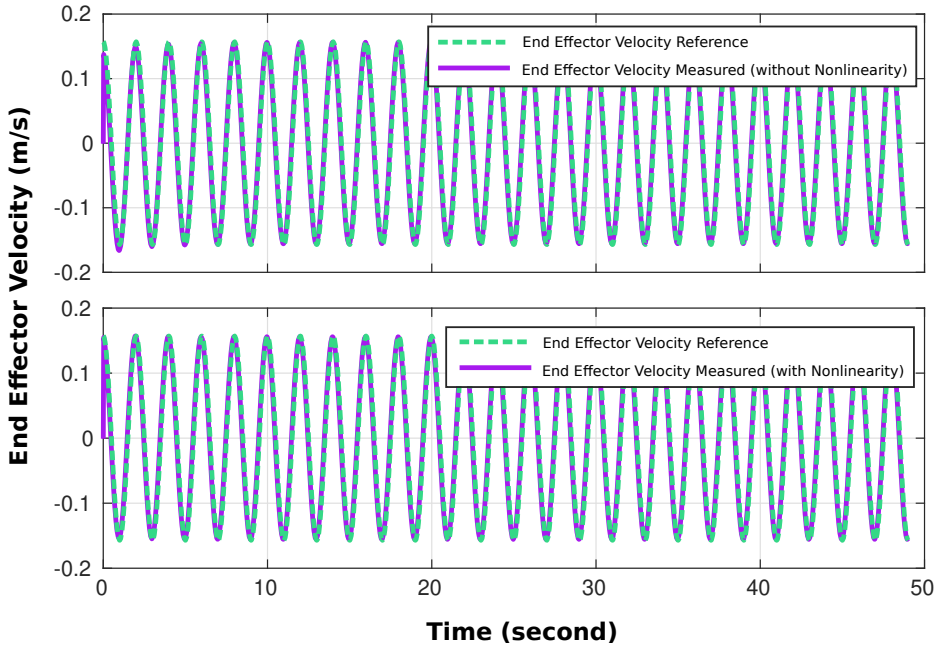

Figure 7: Joint velocity tracking error with high control gains - 0.50 Hz

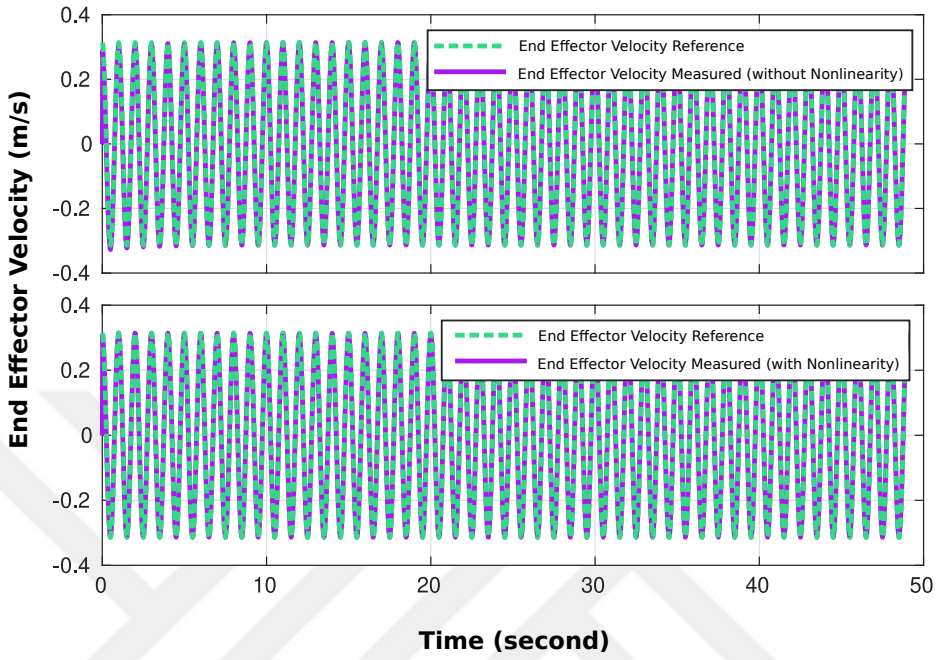

Figure 8: Joint velocity tracking error with high control gains - 1 Hz

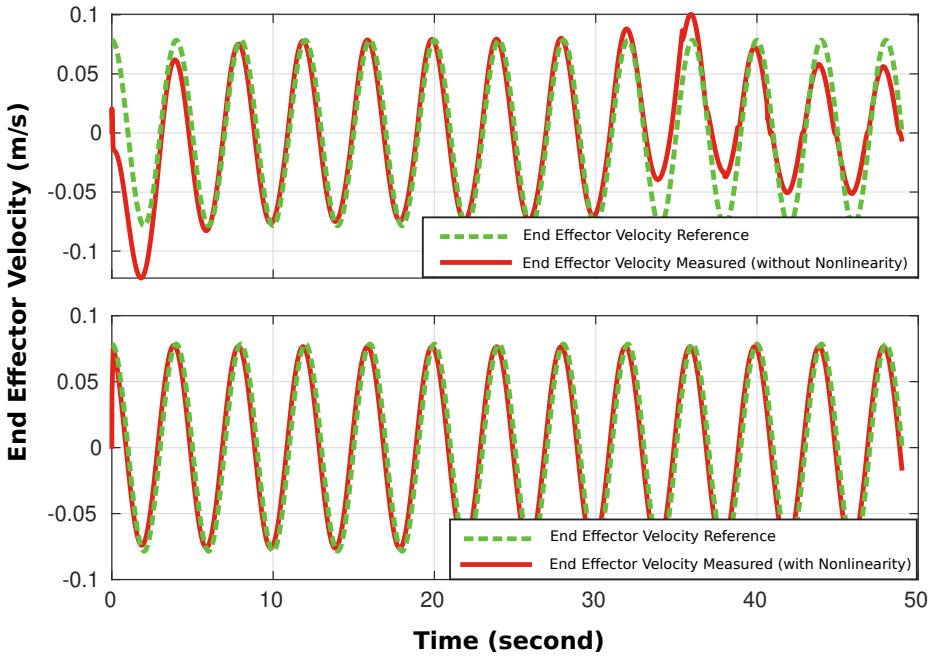

Figure 9: Joint velocity tracking error with low control gains - 0.25 Hz

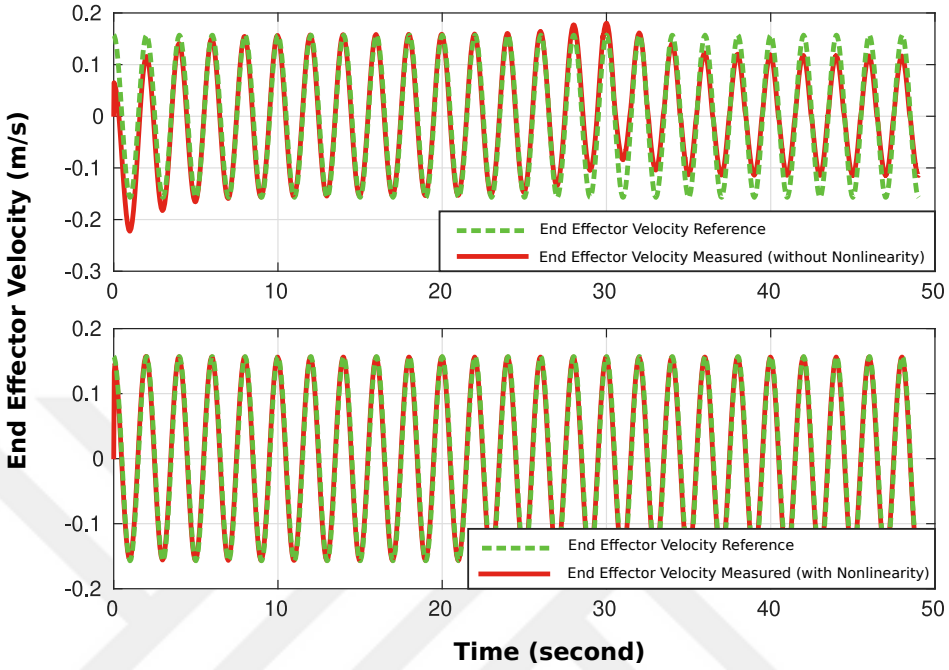

Figure 10: Joint velocity tracking error with low control gains - 0.50 Hz

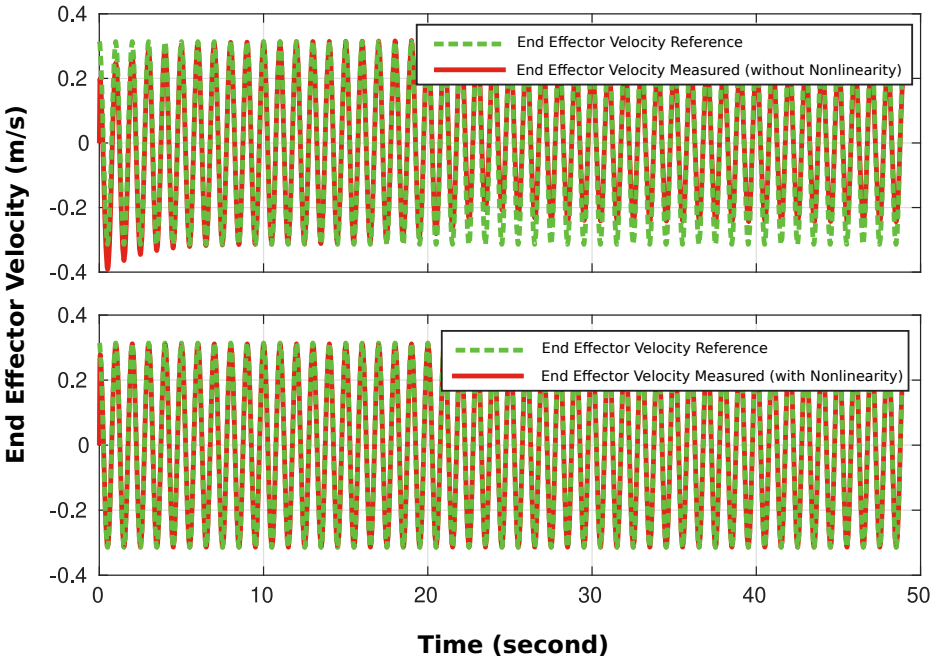

Figure 11: Joint velocity tracking error with low control gains - 1 Hz

CHAPTER III

HUMAN ROBOT SHARED CONTROL

This thesis presents a training procedure for reinforcement learning-based robot control to achieve skill acquisition with reduced training time and data requirements. In this section, two different approaches used for the ball balancing task namely learning-based control and human-in-the-loop learning, are introduced. Learning-based control approach, which can also be called autonomous training, was implemented. In this approach, only the proximal policy optimization algorithm's agent was used to enable Panda to learn the robot ball balancing task. In the other approach, human-in-the-loop learning, in addition to proximal policy optimization, the human is included in the training loop with the action command it sends.

3.1 Task Specification

In the specified task, the learning agent's goal is to bring a 1.5 cm diameter, 0.17 kg ball to the center of a tray attached to Panda's end effector. The objective is to move the ball to the target position with maximum manipulability [23]. To this end, the pitch axis Cartesian moment was assigned as the corrective action in the learning algorithm as if the robot was trained through physical interaction. This requires the use of all 7 joints of the robot. To prevent the ball from falling, the tray is enclosed by walls. During training, the ball is initialized with random positions, and each training episode lasts 15 seconds. The criterion for completing the task is to keep the ball stationary in the center of the tray. The agent aims to maintain the ball's position at the center along the x-axis.

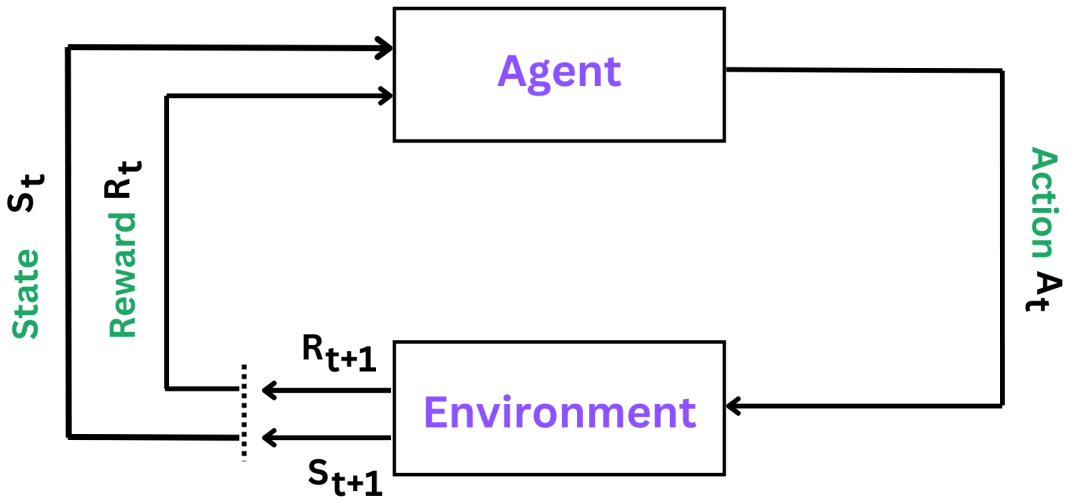

Figure 12: Reinforcement learning training loop

3.2 Reinforcement Learning: Proximal Policy Optimization Algorithm

Reinforcement Learning (RL) is a kind of machine learning that works with agents learning to make decisions by interacting with an environment. The agent takes actions in the environment, receives feedback as rewards or punishments, and adjusts its policy over time to maximize the cumulative reward. The goal is for the agent to learn a policy that maps observations to actions, optimizing its decision-making process through trial and error as shown in Fig. 12. RL algorithms are divided into two: model-free and model-based.

Model-based RL involves the agent building an internal model or representation of the environment. This model is used for planning and decision-making, allowing the agent to simulate different scenarios and predict the outcomes of its actions without directly interacting with the environment. By using this learned model, the agent can plan ahead and make more informed decisions. However, the effectiveness of model-based RL depends heavily on the accuracy of the learned model.

Model-free methods include approaches, where the agent learns to associate actions with the expected cumulative rewards. Model-free RL is often preferred in situations where the environment is complex or poorly understood, making it challenging to create an accurate model [24]. One of the model-free algorithms in the context of RL is PPO. PPO is an iterative optimization algorithm designed to enhance the policy of an RL agent. It aims to strike a balance between exploring new strategies and exploiting known ones by iteratively collecting experiences, optimizing the policy, and updating the agent.

In this thesis, PPO was used as the learning algorithm. The main difference between the PPO from the other policy gradient methods is that the policy updates are not excessively large, meaning that the new policy does not deviate significantly from the prior policy. Due to this property, PPO outperforms other online policy gradient methods and achieves a favorable balance between sample complexity and wall time[25]. This thesis utilizes PPO as the learning algorithm. In the context of reinforcement learning, a policy is a mapping from the action space to the state space. It can be explained as a set of instructions for the learning agent, specifying the actions it should take based on the current state of the environment. PPO updates the existing policy in each step.

The value function $V^{\pi}(s)$ describes the quality of the applied action in the current state s, under the given policy. It calculates the expected sum of rewards using the policy π, the state transition probabilities $p(s', r|s, a)$, the immediate reward r, and the discounted value function concerning the expected next state $V^{\pi}(s')$, as shown below.

$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V^{\pi}(s')] \tag{30}$$

In (30), s, s', a, and γ denote the current state, the expected next state, the action, and the discount factor, respectively. The actor-critic neural network within the PPO

algorithm generates the next action through the assessment of policy π which makes use of s, s', a, r, and $V^{\pi}(s)$.

3.3 Learning Based Control

A custom reward function has been created for the r value and states defined for the s value in the (30), taking into account the specified task. States are determined as errors in ball position, velocity, and also the pitch rotation of the tray. As for the reward function r, task-specific parameters are included to balance the ball. When the pitch orientation of the tray exceeds the designated threshold value of 25 degrees, it introduces challenges in controlling the ball's velocity on the tray. Consequently, the term R_{α} is introduced to penalize excess orientation, where ω_{α} represents the related weight constant as shown below.

$$R_{\alpha} = -\omega_{\alpha}\alpha^2 \tag{31}$$

$$R_V = -\omega_V \left((e_{V_x})^2 + marg_x\right) \tag{32}$$

$$marg_x = \omega_{marg}\|e_x\| \tag{33}$$

Concerning the specified task, the ball needs to remain stationary at the center with a velocity close to zero. To accomplish this, the reward function was extended by including a velocity term R_V, which penalizes the ball's velocity. ω_V represents the weight of the velocity term,α represent the pitch angle, and e_{V_x} signifies the velocity error of the ball on the x-axis. That being said, it introduces the possibility of getting stuck in a suboptimal outcome where the ball remains balanced with zero velocity at the edges of the tray. To address this issue, an additional term denoted as $marg_x$ was incorporated into the velocity penalty term where $\|e_x\|$ is the norm error of the ball position along the x-axis and ω_{marg} is the weight of this margin term. This term

provides a negative reward based on the ball position error if it comes to a stop at the endpoints.

$$k_w = \sqrt{\det\left(JJ^T\right)} \tag{34}$$

$$R_{kw} = \omega_{k_w}\left(k_w\right)^2 \tag{35}$$

One central aspect of the specified task is to maximize the manipulability index k_w such that the learning algorithm may not force the robot to near-singular configurations [23]. This is achieved via the reward term as in (34). In (34) and (35), J is the end-effector Jacobian matrix, and ω_{k_w} is the weight constant associated with the reward R_{kw}.

As the ball position error increases, the agent should receive a negative reward. This is provided via the R_P term that is proportional to the norm of the position error, and its weight is adjusted via the constant ω_P. While positioning error is penalized, the agent is rewarded via the R_{P_e} term that exponentially increases the positive reward while the ball is centered.

$$R_P = -\omega_P \|e_x\| \tag{36}$$

$$R_{P_e} = e^{\omega_{P_e}\|e_x\|} \tag{37}$$

In (37), the weight ω_{P_e} adjusts the convergence rate of exponential reward increase. Before simulation experiments, the weights were empirically tuned in a way that the unmodeled robot dynamics were not excited and the robot behavior was in line with the task requirements. Finally, the overall reward function is defined as the superposition of all reward sub-functions:

$$r = R_\alpha + R_V + R_{kw} + R_P + R_{P_e} \tag{38}$$

Note that penalizing rewards has minus signs. For the target ball balancing task, the states (s) are designated as ball position, ball velocity, and tray orientation. As previously mentioned, the generated PPO action (a) is assigned as the pitch axis Cartesian moment M_{yai}, since the ball balancing task does not require forces for other axes. For the case of human-in-the-loop training, the human agent also produces the action M_{yh}. Finally, the overall action is the superposition of both terms: $M_y = M_{yai} + M_{yh}$. Note that if the training is performed autonomously, $M_{yh} = 0$.

To incorporate dynamics into our learning-based control scheme, we implemented computed torque control; see Fig. 13. Reference joint positions are given as constants and they were set to 0, -90, 0, -90, 0, 179, and 90 degrees, respectively, from bottom to top. The command torque τ_{cmd} is computed as follows:

$$\tau_{cmd} = \tau_{pd} + \tau_{ppo} + \tau_g \tag{39}$$

where τ_{pd} is the output of PD controller, τ_g is feedforward torques that account for inverse dynamics. As the ball position is initialized randomly in each episode, the action M_y disturbs the set position in a way to balances the ball. To this end, it is inserted into the controller as the feedforward torque term τ_{ppo}, which is computed as follows:

$$\tau_{ppo} = J^\intercal W_{tr} \tag{40}$$

In (40), W_{tr} the Cartesian end-effector wrench; $W_{tr} = [0\ 0\ 0\ 0\ M_y\ 0]^\intercal$. Through the integration of robot dynamics into our learning-based control scheme, the controller gains were set to relatively low values; therefore, the action M_y could disturb the system to provide corrective actions for the ball balancing task.

In the training process, each episode has a duration of 15 seconds. An episode

terminates when the ball position along the vertical axis falls below a certain threshold value. The episode is also terminated when the maximum episode time step is reached. Upon the episode reset, the robot returns to its home position. To promote exploration, the ball position is initialized randomly in each episode.

3.4 Human-in-the-loop Learning

To accommodate human action, three keyboard keys were defined: 'right,' 'left,' and 'space.' These keys transmit human action command M_{yh} as 'right' for negative magnitude and 'left' for positive magnitude. Each keystroke results in a change of ± 0.05 Nm. The 'space' key serves as a reset mechanism, enabling human agents to restore their action value to zero.

At the beginning of each simulation experiment, the environment was introduced to the participants, and the necessary keyboard commands were explained. The experiment was described and they were given extra trials until they were familiarized with the controls. Each simulation experiment was concluded when a human agent decided with her/his own will such that the robot acquired the target skill from her/his perspective. Each episode had a fixed duration of 15 seconds, and participants were allotted a maximum of 45 minutes to complete the simulation experiment.

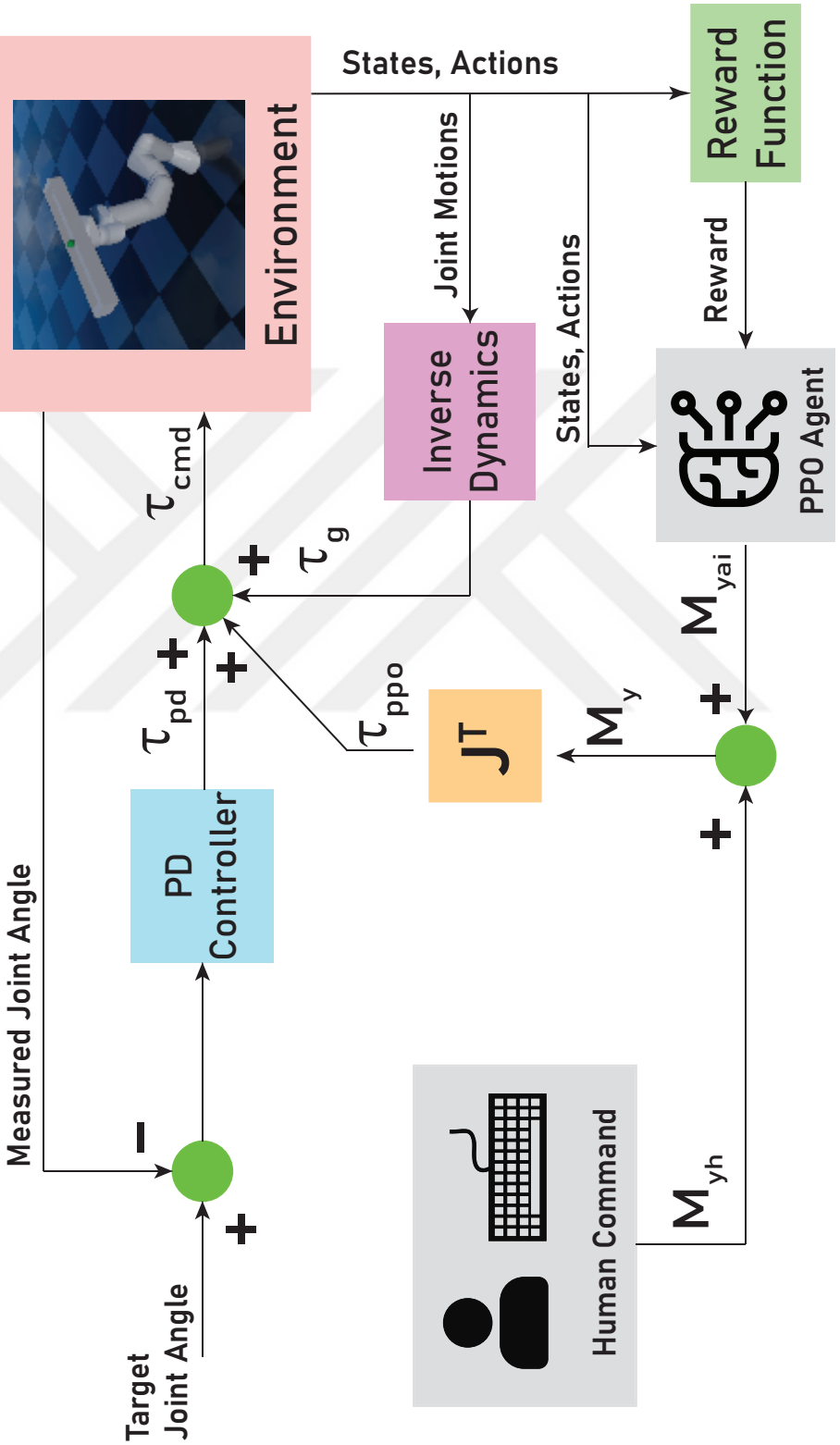

Figure 13: The proposed learning-based control structure

CHAPTER IV

SIMULATION EXPERIMENTS

Simulation experiments were conducted with 24 volunteer subjects including 16 males and 8 females, aged between 18-52. The subjects were selected from individuals who had basic computer skills. Upon the completion of all human-in-the-loop training sessions, the trained models were tested and only those with ball-balancing skills were labeled as successful. Out of 24 human-trained models, 22 of them were successful and considered in our study.

4.1 Performance Assessment: Training Time

To compare baseline and human-in-the-loop training approaches, the best-performing five subjects/agents were selected. These models were chosen based on their performance in a 30-second test episode, with rankings determined via their cumulative rewards. When using the proposed human-in-the-loop training approach, skill acquisition was attained with an average of 17 episodes. The average training time was approximately 10 minutes. In contrast, the same objective was achieved with an average of 788 episodes when using the baseline training approach. The average training time was approximately 10 hours. This result indicates that the proposed training approach provides a significant reduction in training time and data requirements as illustrated in Fig. 15.

Furthermore, we plotted the cumulative reward variation during training sessions of best performers; see Fig. 14. The solid cyan and purple lines show the cumulative reward of the best-performing subject/agent concerning the proposed and baseline training approaches, respectively. For the case of human-in-the-loop training, the skill acquisition was completed after 45 episodes, whereas this number was around

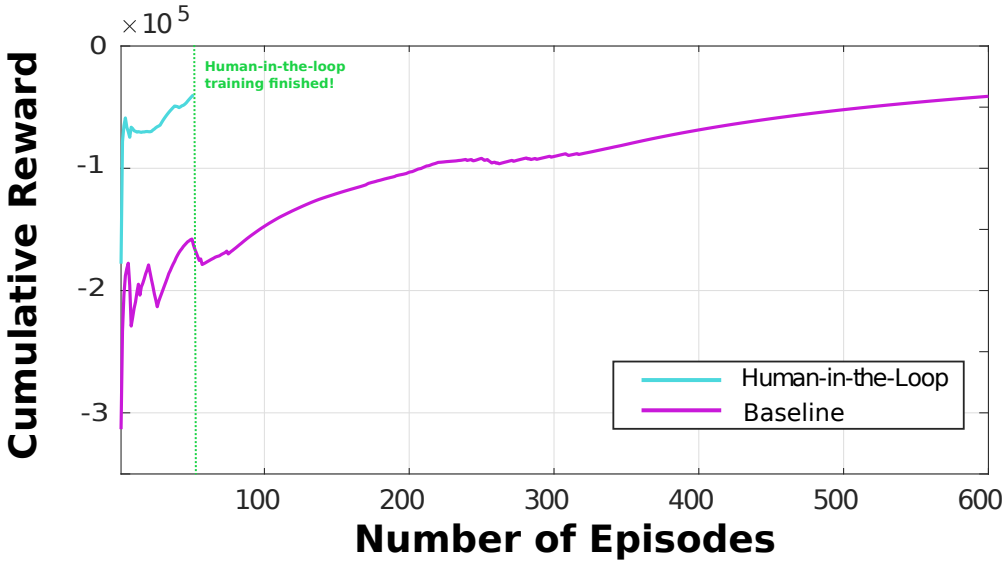

Figure 14: Cumulative reward achieved during training

600 for the case of baseline training.

4.2 Testing Trained Models: A Randomized Procedure

In order to test the trained models, we ran simulation experiments in which the ball was initialized randomly and the learning-based controller retrieved its position to the center of the tray. Each trained model was tested for a total of 20 episodes and each episode lasted 10 seconds. The best five models from both training approaches were compared by considering the ball position and velocity errors. Figs. 16 and 17 display the mean ball position error and mean ball velocity error plots. The blue lines represent the mean errors of the models that were trained via the proposed human-in-the-loop approach. The brown lines represent the mean errors of the models that were trained via the baseline approach. The blue and yellow areas illustrate the variance in the related mean errors ± SEM (Standard Error of the Mean). The ball position error was obtained from the position difference between the ball and the center of the tray. A similar strategy was employed for ball velocity error.

As may be examined in Fig. 16, the models trained via the human-in-the-loop

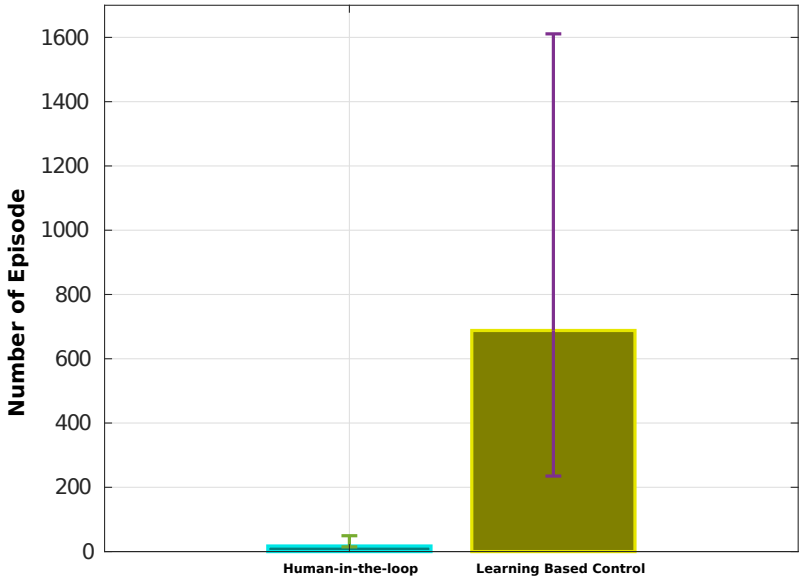

Figure 15: Number of episodes in which training is complete

approach demonstrated more favorable ball balancing ability as they maintained the ball at the center of the tray with relatively lower variance, compared to the baseline approach. Furthermore, the mean error remained close to zero. In contrast, the mean ball position error of models that were trained via the baseline approach exhibited an oscillatory behavior; the ball was kept around the center but could not be stabilized. This performance difference can also be viewed in Fig. 17. The models that were trained via the proposed approach kept the ball stationary, but the other models could not show the same regulatory behavior. In light of these results, it is observed that the proposed training approach yielded models that outperformed the baseline approach.

Moreover, the manipulability index of the human-in-the-loop-trained model was around 0.97 during the test. This value is quite close to its maximum value of 1. Therefore, it is observed that the skill acquisition was attained whilst preventing singular configurations.

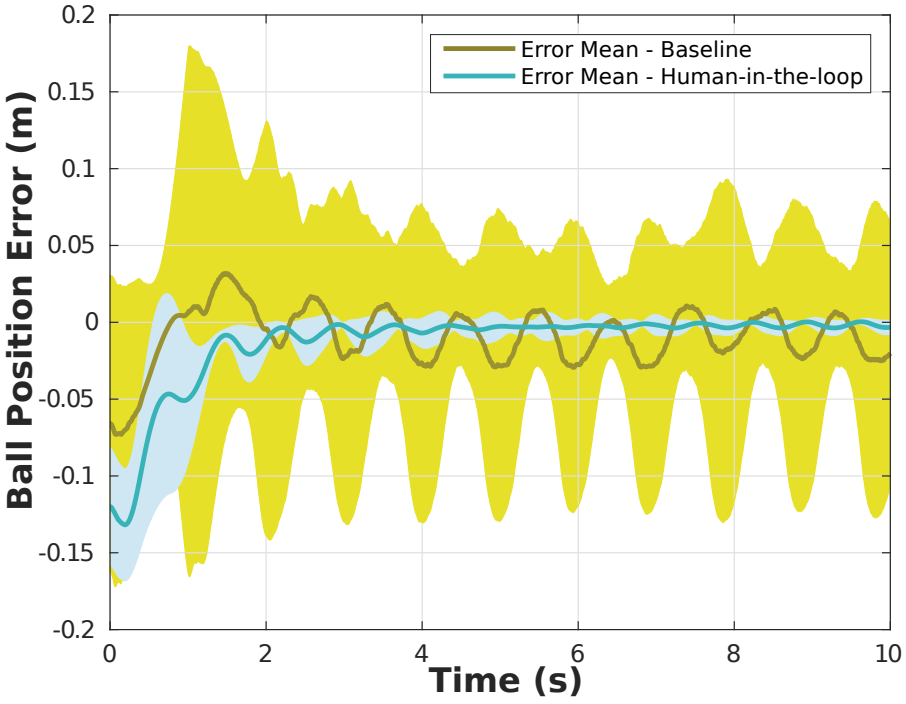

Figure 16: Ball position error - randomized test procedure. The lines represent mean errors, shaded areas represent ±SEM.

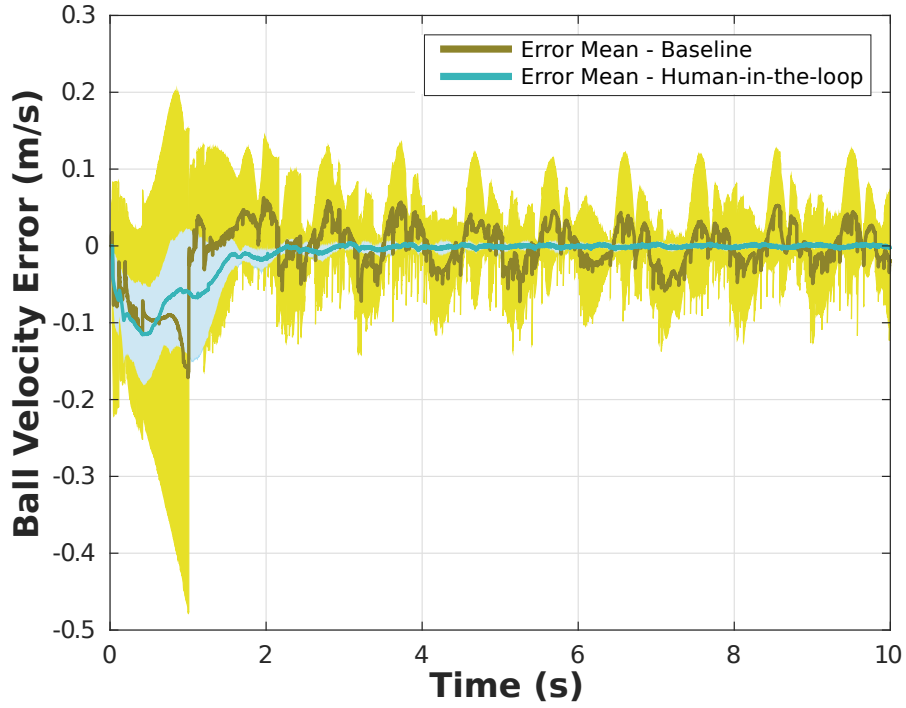

Figure 17: Ball velocity error - randomized test procedure. The lines represent mean errors, shaded areas represent ±SEM.

4.3 Testing Trained Models: A Deterministic Procedure

For a deterministic evaluation, the ball was initialized at the center of the tray, however, external perturbations were applied to the end effector. Such perturbations were not previously encountered during model training. In this procedure, each model underwent a single episode that lasted 30 seconds. Perturbations appeared as step functions with an amplitude of 50 N along both vertical and horizontal axes. They were applied twice: from fifth to sixth seconds, and from tenth to eleventh seconds. Ball position and velocity error variations can be viewed in Figs. 18 and 19. In these figures, the solid green and purple lines show mean error variations for the proposed and baseline training approaches, respectively. Light green and purple areas indicate mean±SEM.

Observing Figs. 18 and 19, it may be judged that both training approaches yielded models that exhibited the ability to regulate ball position despite disturbances. On this note, the models trained via the proposed human-in-the-loop approach displayed robustness against disturbances; see Fig. 18. The models that were trained via the baseline approach exhibited higher variability, especially within the presence of disturbances. In particular, the spikes in mean velocity errors observed in Fig. 19 indicate that these models were highly influenced by the disturbances, causing sudden accelerations. In contrast, the models that were trained using the human-in-the-loop approach effectively slowed down the ball upon the appearance of a disturbance, bringing the mean velocity error close to zero as shown in Fig. 19.

Similar to the previous test, the manipulability index of the human-in-the-loop-trained model was around 0.97 during this test as well. Hence, the model was able to achieve the task while avoiding singular configurations.

Exemplary video clips from the training procedure and performance tests can be accessed via https://youtu.be/9vLlbr7ypXk.

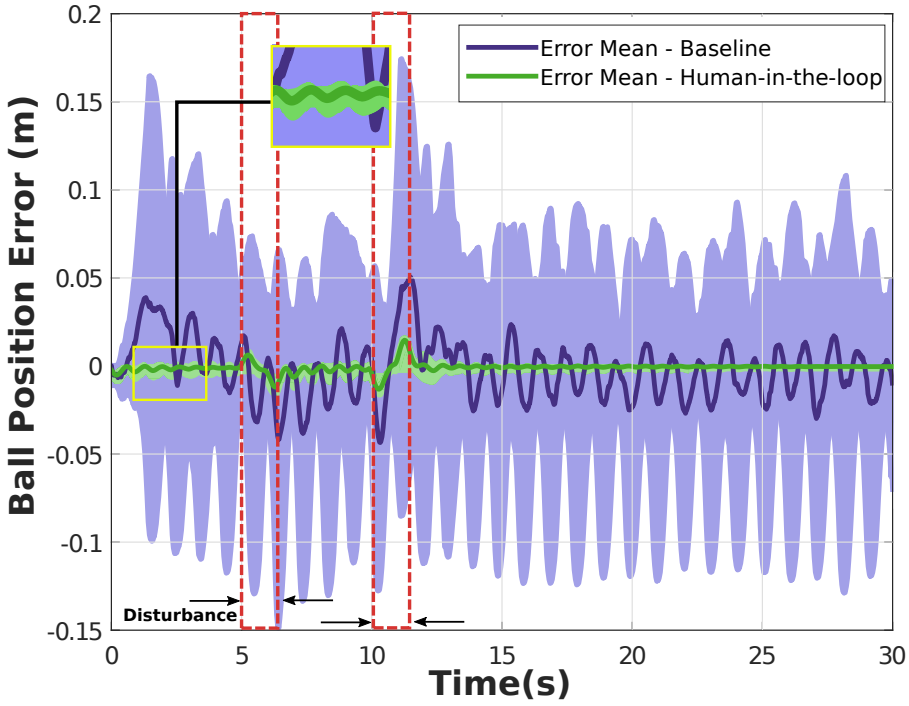

Figure 18: Ball position error - deterministic test procedure. The lines represent mean errors, shaded areas represent ±SEM.

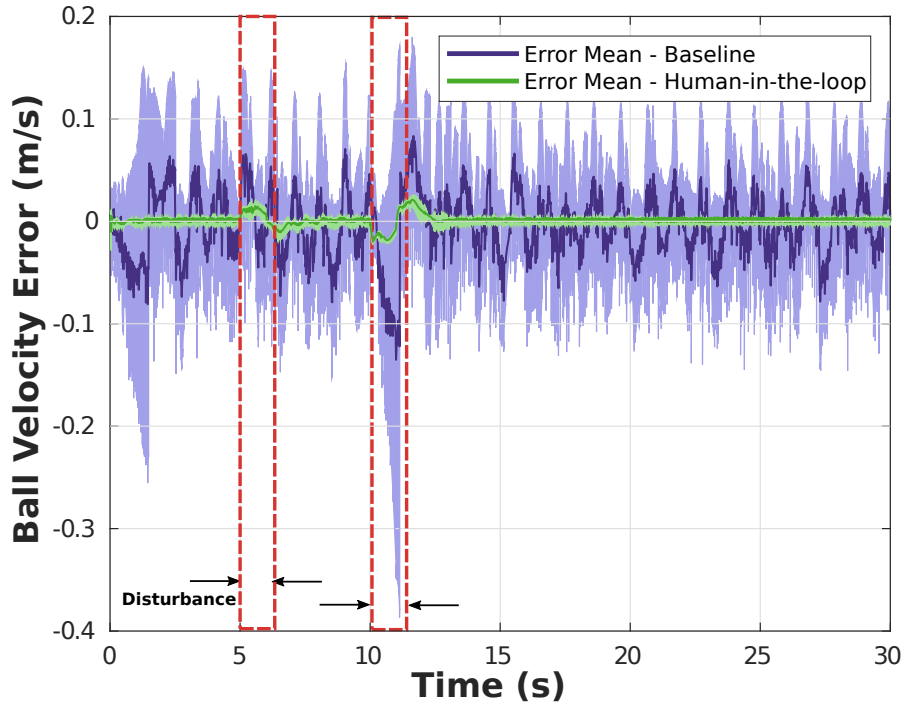

Figure 19: Ball velocity error - deterministic test procedure. The lines represent mean errors, shaded areas represent ±SEM

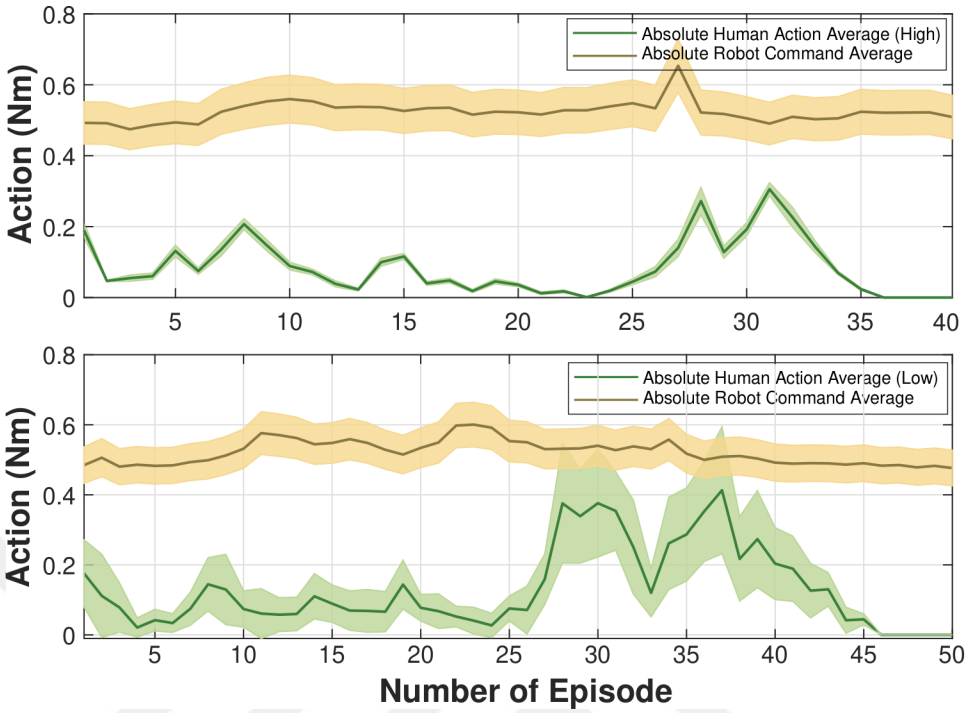

Figure 20: Average action command of high/low performing models. Lines represent absolute average action, shaded areas represent ±SEM

4.4 Individual Differences: High and Low Performing Models

While the proposed training approach is assessed to be efficient in many aspects, it is essential to acknowledge individual differences where personal factors may be influential. To evaluate these differences, low and high-performing human-in-the-loop-trained models were identified based on their cumulative rewards. We compared the keyboard actions (M_{yh}) that were sent to the environment by the human agents. The results are depicted in Fig. 20, where the brown and green lines respectively represent the average actions generated by the PPO and human agents during training. The green and orange shaded regions correspond to the variances in these actions with ±SEM.

Observing Fig. 20, both individuals generated low-magnitude actions initially and they started to increase their average action following the 25^{th} episode. This increased

the human intervention of the PPO agent and facilitated improved ball balancing, ultimately culminating in successful task acquisition.

Following the 25^{th} episode, the magnitude of actions increased for both participants, but the individual with high performance demonstrated faster task adaptation from this point onward. The duration of the remaining training time depends on the participant's adaptation ability to the collaborative process with the robot. As observed in Fig. 20, the participant with low performance generated rapid changes in response. In comparison, the individual with high performance modulates his/her average actions in a somewhat controlled manner. These differences in responses among participants significantly influence the reactions of the corresponding models to disturbances. Participants with high performance delegated control to the robot after the 35^{th} episode, whereas those with low performance did so after the 45^{th} episode. To conclude, we observe that the actions of the high-performing individuals had significantly lower variance when compared to low-performing individuals.

During performance tests, both models demonstrated the capability to balance the ball at the center. However, the model trained via a low-performing human agent showed abrupt movements when the robot was perturbed. In contrast, the model trained via a high-performing human agent could attenuate the perturbations in a somewhat compliant manner and regulate the ball position successfully.

4.5 Individual Differences: Expert and High Performing Models

In this performance assessment, the model that trained with the human-in-the-loop approach and achieved the highest average reward in the tests, was selected as the high performer. Deniz Yılmaz, the person who conducted this thesis study, was designated as the expert. As indicated in the comparison of high and low-performing models, the high-performing model increased magnitude of the actions after the 25^{th} episode, and adaptation to the task started after that episode. As illustrated in Fig. 21, the expert

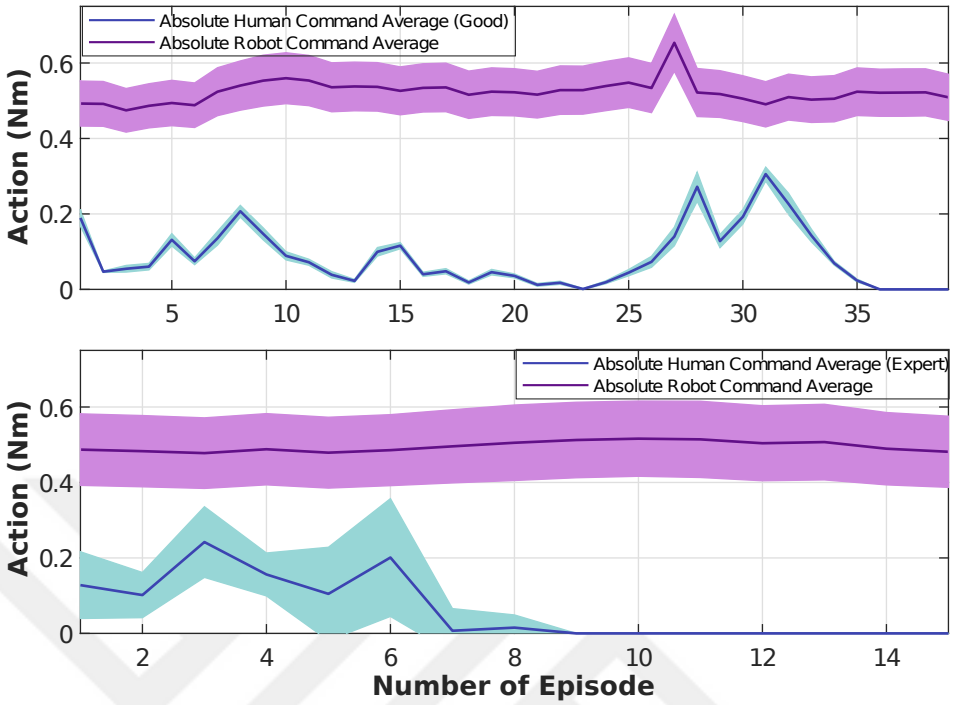

Figure 21: Average action command of expert/high performing models. Lines represent absolute average action, shaded areas represent ±SEM

exhibited the adaptation of the task starting from the first episode. Consequently, she guided the robot's actions by implementing high-magnitude actions at the beginning. The variance represented by the blue area that expert's action has, is due to quick response to every action produced by the robot. After the 9^{th} episode, robot become autonomous. As can be seen in Fig. 21, expertise in the task facilitates faster training procedure without the need for an adaptation process to the environment and the task. Expert trainers may not always be available, however, the proposed method also works well with novice human trainers.

4.6 Achieving Ball Balance: End Effector Trajectory Reference

The trained models demonstrated the capability to balance the ball at the center through collaborative learning with a human subject as in the proposed method. The relevant models have been tested using a deterministic procedure, and their

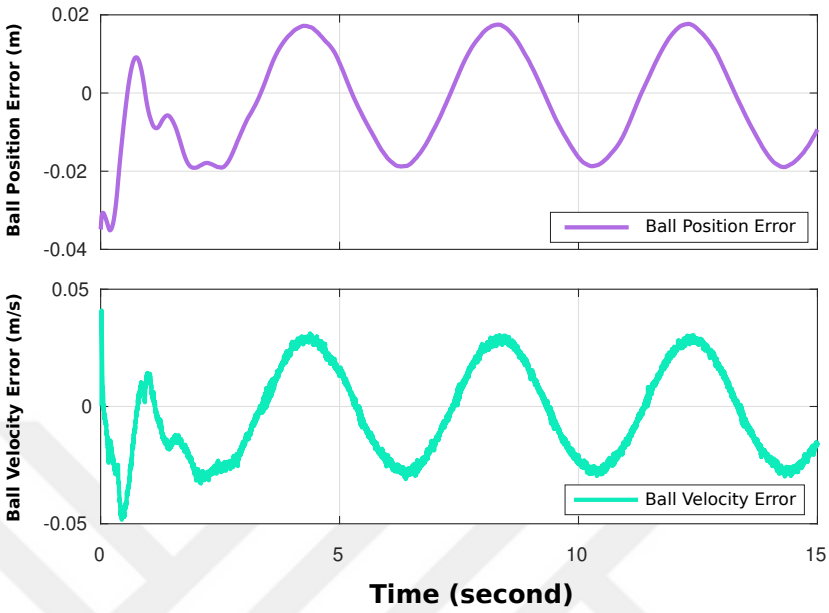

Figure 22: Ball position and velocity error - 0.25 Hz disturbance

performances have been discussed. Subsequently, the adaptability of the trained models was evaluated under dynamic conditions by introducing three different cosine waves as linear velocity references to the end effector in the x-axis as shown in Figs. 3, 4, 5. This setup induced a continuous sinusoidal motion in the tray. In this test procedure, the ball is initiated from the center of the tray. The most successful model trained with a human subject was used for this performance test.

The aim was to assess the model's ability to adapt to varying frequencies and successfully balance the ball under the influence of the sinusoidal disturbance. Figs. 22, 23, and 24 depicting ball velocity and position errors were utilized for a comprehensive evaluation. The obtained results illustrate the model's effectiveness in responding to dynamic inputs and maintaining stability in the face of the introduced sinusoidal trajectories.

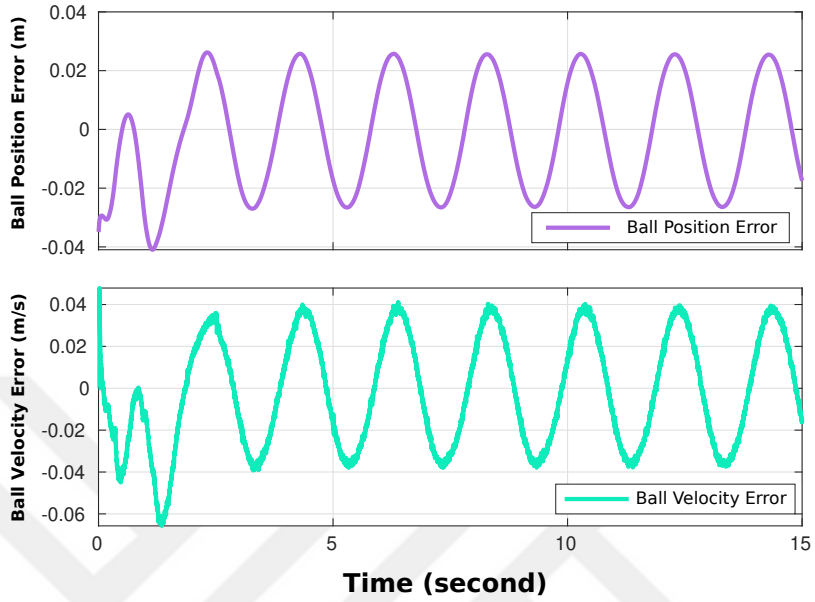

Figure 23: Ball position and velocity error - 0.50 Hz disturbance

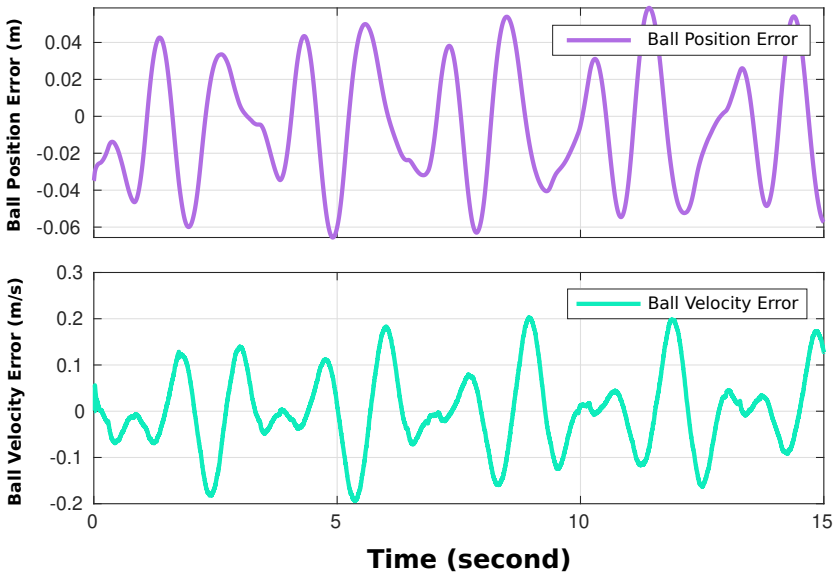

Figure 24: Ball position and velocity error - 1 Hz disturbance

4.7 Discussion

In the context of learning speed, the human-in-the-loop training approach demonstrates an average reduction in training time of 97.6%. Furthermore, when assessing performance metrics such as ball position error and ball velocity error, it becomes evident that the human-in-the-loop approach outperforms the baseline approach; see Figs. 16 and 17.

The deterministic test procedure assessed the agent's response to disturbance effects that were not encountered during training. Figs. 18 and 19 indicate that the models that were trained via the proposed human-in-the-loop approach effectively stabilize the ball, reducing mean errors to approximately zero with lower variance.

In addition to randomized and deterministic test procedures, the most successful trained model was disturbed with linear velocity references of different frequencies. The relevant disturbance caused a sinusoidal change in the position of the tray. The success of the trained model in performing the ball balancing task in the disturb environment was measured. As seen in Figs. 22,23 and 24, despite the disturbance, the trained model was able to balance the ball's position and velocity error by keeping it close to zero.

The human-in-the-loop approach, dependent on human involvement in the training process, inherently exhibits individual differences. The experiment engaged 24 different participants, revealing variations in the performance of the learned models based on the performance of the participants and responses during the task teaching process. Notably, the high-performer completed training 20% faster than the low-performing one. Although both models successfully learned to balance the ball, the subject with high performance secured a 190% higher cumulative reward.

The same training procedure was applied in a way that only the human sent an action command to the environment without the PPO agent's action, but no task learning occurred. With longer training humans may discover a suitable exploration

policy that may allow PPO to learn, but within the experimental settings we had, this was not observed, thus subject data alone was not sufficient for PPO to converge to a solution.

CHAPTER V

CONCLUSION

This thesis presented a training procedure for reinforcement learning-based robot control to achieve skill acquisition with reduced training time and data requirements. This is achieved by actively integrating both human-provided actions and those generated by the learning algorithm. As the algorithm progressively masters the desired skill, the reliance on human inputs decreases, leading to more autonomous task execution. To show the effectiveness of the proposed method, a ball-balancing task emphasizing manipulability index maximization was selected. In this scenario, a 7-degree-of-freedom robot arm underwent skill acquisition through reinforcement learning algorithm, employing two distinct training approaches: i) autonomous training and ii) a human-in-the-loop training approach. The objective is to bring a ball to the center of a tray attached to its end-effector, starting from varied initial ball positions while facing external perturbations.

For the baseline approach as autonomous training, only the proximal policy optimization algorithm was used and training was performed for the ball balancing task. For the proposed human-in-the-loop training approach, human agents were able to send action commands, and training was carried out in the same environment as in the baseline approach. The presented human-in-the-loop training approach was verified in simulations with human participants. After undergoing two distinct training procedures, the trained models were evaluated using performance metrics such as the position and velocity error of the ball. Various test procedures, including both randomized and deterministic, were employed to evaluate the performance of the trained

models. Consistent with the test results, it becomes apparent that the human-in-the-loop approach surpasses the baseline approach when assessing performance metrics.

In evaluating the training procedure concerning learning speed, the human-in-the-loop training approach exhibits an average reduction in training time of 97.6% for the specified task. Following the tests conducted using the randomized procedure, the models trained through the human-in-the-loop approach showcased superior ball-balancing capabilities. They consistently kept the ball at the center of the tray with relatively lower variance compared to the baseline approach. The deterministic test procedure evaluated the agent's response to disturbance effects not encountered during training. Models trained through the proposed human-in-the-loop approach effectively stabilized the ball, minimizing mean errors to nearly zero with lower variance.

The human-in-the-loop approach, reliant on human involvement in the training process, inherently demonstrates individual differences. The experiment involved 24 different participants, uncovering variations in the performance of the learned models based on the participants' capabilities and responses during the task-teaching process. Significantly, the high-performing participant completed training 20% faster than the low-performing one. While both models successfully learned to balance the ball, the high-performing subject achieved a 190% higher cumulative reward.

PPO is a stochastic algorithm and produces a noisy action for exploration. Therefore, as seen in the results, there is chattering when there is no human agent in the training. The human agent intervenes with continuous actions, thus leading to smoother and more compliant regulation behavior compared to baseline approach. What is more, it was observed that the proposed approach yielded better-performing policies.

The proposed method in this thesis, not only showed that human involvement

facilitates faster skill acquisition and learning of more robust policies for the ball-balancing task but also facilitates smoother exploration during the training phase.

APPENDIX A

IMPLEMENTATION OF THE LEARNING ENVIRONMENT

```
#Libraries
import numpy as np              #Numpy
import raisimpy as raisim       #Raisimpy for the Raisim simulation environment
import os
import time
#Functions
from Franka_Jac_Calculation import *    #Symbolic Function for Jacobian Derivation of Panda
#GYM
from gym.utils import seeding           #To use random seed in the learning


class Franka_BallBalance_Env:           #Class - Environment for Ball Balancing Task

def seed(self,seed):    #Random Seed Function
print("Seed= %d"%seed)
self.np_random,seed=seeding.np_random(seed)

def __init__(self): #Define and Initialize

#URDF & License (Raisim)

raisim.World.setLicenseFile(os.path.dirname(os.path.abspath(__file__))
+ "/../rsc/activation.raisim")
panda_urdf_file = os.path.dirname(os.path.abspath(__file__))
+ "/../../rsc/Panda/urdf/panda.urdf"

#World & Server

self.world = raisim.World()
self.server = raisim.RaisimServer(self.world)
```

```
#Franka & World
self.ground = self.world.addGround()
self.franka = self.world.addArticulatedSystem(panda_urdf_file)
self.franka.setName("Panda")

#Simulation Parameters

self.dt = 0.001 #Timestep
self.time_step = self.dt
self.world.setTimeStep(self.dt)

#Add Ball to the Environment

self.ball = self.world.addSphere(0.015, 0.17)
self.ball.setAppearance("green")
self.ball.setVelocity(np.array([0,0,0]),np.array([0, 0,0]))

#Server Connection
self.server.launchServer(8080)

#Joint Configuration - Fig. 1

q1_ref = 0   * math.pi / 180
q2_ref = -90 * math.pi / 180
q3_ref = 0   * math.pi / 180
q4_ref = -90 * math.pi / 180
q5_ref = 0   * math.pi / 180
q6_ref = 179 * math.pi / 180
q7_ref = 90  * math.pi / 180

self.joint_position_ref = np.array([q1_ref, q2_ref, q3_ref,q4_ref, q5_ref, q6_ref,q7_ref,0])
#Additional joint as 0 for the tray
self.tray_index = self.franka.getFrameIdxByName("panda_tray_joint")  #Index of the Tray Joint

#Array Definition

self.q_dot_array = np.zeros([8])       #Joint Velocity
self.trayWrench = np.zeros([6])        #Wrench
self.eulerXYZAng = np.zeros([3])       #Array for Euler Angles

#PD Controller Gains
```

```
Kp1 = 200
Kp2 = 700
Kp3 = 200
Kp4 = 500
Kp5 = 200
Kp6 = 200
Kp7 = 200
self.Joint_KP = np.array([Kp1,Kp2,Kp3,Kp4,Kp5,Kp6,Kp7,0])
Kd1 = 25
Kd2 = 700
Kd3 = 25
Kd4 = 800
Kd5 = 25
Kd6 = 25
Kd7 = 10
self.Joint_KD = np.array([Kd1,Kd2,Kd3,Kd4,Kd5,Kd6,Kd7,0])

#Definition of Action - Observation Space (Size)

self.observation_dim = [3]
self.action_dim = [1]

self.steps_beyond_terminated = None

def reset(self):   #Reset Function

#Measure Ball Position
ball_pos = self.ball.getPosition()
ball_pos_x = ball_pos[0]

#Measure Ball Velocity
ball_velocity = self.ball.getLinearVelocity()
ball_velocity_x = ball_velocity[0]

#Measure Tray Position
tray_pos = self.franka.getFramePosition(self.tray_index) #Tray Joint
tray_pos_x = tray_pos[0]

#Ball Position Error in x-axis
ball_pos_error_x = tray_pos_x - ball_pos_x
```

```
#Set Joint Position
self.franka.setGeneralizedCoordinate(self.joint_position_ref)
self.franka.setPdGains(self.Joint_KP,self.Joint_KD)
self.franka.setPdTarget(self.joint_position_ref, np.zeros([8]))

#Set Ball Position & Velocity
ball_x_in = np.random.uniform(-0.3,-0.6) #Random Ball Position
self.ball.setPosition(np.array([ball_x_in,0.008, 0.94]))
self.ball.setVelocity(np.array([0,0,0]),np.array([0, 0,0]))

#Observation Space
self.observation = np.array([ball_pos_error_x,ball_velocity_x,0])
#Initial pitch angle as 0
self.observation = (self.observation-np.min(self.observation))/(np.max(self.observation)
-np.min(self.observation))

self.world.setWorldTime(0.0) #Reset simulation time
self.steps_beyond_terminated = None

return self.observation

def step(self,action,t):    #Step Function
#Measure Position of the Tray
tray_pos = self.franka.getFramePosition(self.tray_index)
tray_pos_x = tray_pos[0]

#Measure Position of the Ball
ball_pos = self.ball.getPosition()
ball_pos_x = ball_pos[0]
ball_pos_z = ball_pos[2]

#Measure Velocity of the Ball
ball_velocity = self.ball.getLinearVelocity()
ball_velocity_x = ball_velocity[0]

#Position Error of the Ball in x-axis
ball_pos_error_x = tray_pos_x - ball_pos_x

#Measure Joint Position & Velocity
mea_joint_position = self.franka.getGeneralizedCoordinate()
```

```
mea_joint_velocity = self.franka.getGeneralizedVelocity()

#Jacobian Calculation
franka_jacobian = Franka_Jacobian(mea_joint_position[0],mea_joint_position[1],
mea_joint_position[2],\mea_joint_position[3],mea_joint_position[4],
mea_joint_position[5],mea_joint_position[6])

#Action
MyRef = action[0]*100

#Wrench
self.trayWrench[0] = 0
self.trayWrench[1] = 0
self.trayWrench[2] = 0
self.trayWrench[3] = 0
self.trayWrench[4] = MyRef
self.trayWrench[5] = 0

#Tau PPO
tauPpo = np.matmul(franka_jacobian.transpose(),self.trayWrench)
tauPpoArray = np.array([tauPpo[0],tauPpo[1],tauPpo[2],tauPpo[3],
tauPpo[4],tauPpo[5],tauPpo[6],0])

#Tau G - Nonlinearity
f_gravity = self.world.getGravity()
panda_nonlin = self.franka.getNonlinearities(f_gravity)

#Torque Command
tau_app = tauPpoArray + panda_nonlin #Additional Tau PD is added
with "setPdTarget()"" function
self.franka.setGeneralizedForce(tau_app)

#Reward Function = r = R_alpha + R_V + R_kw + R_P + R_Pe

#R_kw (Manipulability)
kw = 0.5
w =  math.sqrt( np.linalg.det( np.matmul(franka_jacobian,franka_jacobian.transpose())))
R_kw = kw * (1-(w*w))

#R_P
pos_error_square_x = pow(ball_pos_error_x,2)
```

```
pos_error_max_square_x = 0.0642
norm_error_x = pos_error_square_x/pos_error_max_square_x
R_Pos = (norm_error_x*10)

#R_Pe
R_Pe = math.exp(-20*norm_error_x)

#R_V (Velocity)
weight_velocity = 0.333
margin_x = math.floor(norm_error_x)
velocity_error = (ball_velocity_x**2) + 5*margin_x
R_Vel = weight_velocity*velocity_error

#R_alpha (Orientation)

orient_j7 = self.franka.getBodyOrientation(7)   #Orientation of the Tray
self.eulerXYZAng[0] = math.atan2(-orient_j7[1][2],orient_j7[2][2])
cosqb = math.sqrt(orient_j7[1][2] * orient_j7[1][2] + orient_j7[2][2] * orient_j7[2][2])
self.eulerXYZAng[1] = math.atan2(orient_j7[0][2],cosqb)
self.eulerXYZAng[2] = math.atan2(-orient_j7[0][1],orient_j7[0][0])

pitch_square = self.eulerXYZAng[1] * self.eulerXYZAng[1]
ori_weight = 1

R_alpha = (ori_weight*pitch_square)

#Reward Conditions

if ball_pos_z < abs(0.836): #Threshold position of the ball in z-axis

reward = -5000
done = True

else:

reward = -1*(R_alpha + R_Pos + R_Vel) + R_Pe + R_kw
done = False

#Observation
self.observation = np.array([ball_pos_error_x,ball_velocity_x,self.eulerXYZAng[1]])
self.observation = (self.observation-np.min(self.observation))/
```

```
(np.max(self.observation)-np.min(self.observation))


return self.observation, reward, done, None


def kill(self):
self.server.killServer()
```

REFERENCES

[1] iF Design, "Franka emika panda," 2019.

[2] C. Brosque, E. Galbally, O. Khatib, and M. Fischer, "Human-robot collaboration in construction: Opportunities and challenges," in *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pp. 1–8, 2020.

[3] P. Chang, R. Luo, M. Dorostian, and T. Padır, "A shared control method for collaborative human-robot plug task," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7429–7436, 2021.

[4] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys*, vol. 50, apr 2017.

[5] C. Arzate Cruz and T. Igarashi, "A survey on interactive reinforcement learning: Design principles and open challenges," in *Proceedings of the 2020 ACM designing interactive systems conference*, pp. 1195–1209, 2020.

[6] S. Kamthe and M. Deisenroth, "Data-efficient reinforcement learning with probabilistic model predictive control," in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics* (A. Storkey and F. Perez-Cruz, eds.), vol. 84 of *Proceedings of Machine Learning Research*, pp. 1701–1710, PMLR, 09–11 Apr 2018.

[7] P. Kormushev, S. Calinon, B. Ugurlu, and D. Caldwell, "Challenges for the policy representation when applying reinforcement learning in robotics," pp. 2819–2826, 06 2012.

[8] X. Wu, L. Xiao, Y. Sun, J. Zhang, T. Ma, and L. He, "A survey of human-in-the-loop for machine learning," *Future Generation Computer Systems*, 2022.

[9] D. Xin, L. Ma, J. Liu, S. Macke, S. Song, and A. Parameswaran, "Accelerating human-in-the-loop machine learning: Challenges and opportunities," *ACM Computing Surveys*, 2018.

[10] Z. Wang and T. Hong, "Reinforcement learning for building controls: The opportunities and challenges," *Applied Energy*, vol. 269, p. 115036, 2020.

[11] H. B. Suay and S. Chernova, "Effect of human guidance and state space size on interactive reinforcement learning," in *2011 RO-MAN*, pp. 1–6, 2011.

[12] T. Cederborg, I. Grover, C. L. Isbell Jr, and A. L. Thomaz, "Policy shaping with human teachers.," in *IJCAI*, pp. 3366–3372, 2015.

[13] R. Arakawa, S. Kobayashi, Y. Unno, Y. Tsuboi, and S.-i. Maeda, "Dqn-tamer: Human-in-the-loop reinforcement learning with intractable feedback," *arXiv preprint arXiv:1810.11748*, 2018.

[14] B. Luo, Z. Wu, F. Zhou, and B.-C. Wang, "Human-in-the-loop reinforcement learning in continuous-action space," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–10, 2023.

[15] L. Peternel, T. Petrič, and J. Babič, "Human-in-the-loop approach for teaching robot assembly tasks using impedance control interface," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1497–1502, 2015.

[16] L. Peternel, T. Petric, E. Oztop, and J. Babič, "Teaching robots to cooperate with humans in dynamic manipulation tasks based on multi-modal human-in-the-loop approach," *Autonomous Robots*, vol. 36, pp. 1–14, 01 2014.

[17] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 411–444, 2022.

[18] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018.

[19] C. Gaz, M. Cognetti, A. Oliva, P. Giordano, and A. Luca, "Dynamic identification of the franka emika panda robot with retrieval of feasible parameters using penalty-based optimization," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–1, 07 2019.

[20] J. Lenarčič and B. Siciliano, *Advances in Robot Kinematics 2020*, vol. 15. Springer Nature, 2020.

[21] W. Khalil, "Dynamic modeling of robots using recursive newton-euler techniques.," vol. 1, pp. 19–31, 06 2010.

[22] R. Featherstone, *Rigid Body Dynamics Algorithms*. Berlin, Heidelberg: Springer-Verlag, 2007.

[23] T. Yoshikawa, "Manipulability of robotic mechanisms," *The International Journal of Robotics Research*, vol. 4, p. 3–9, June 1985.

[24] P. Swazinna, S. Udluft, D. Hein, and T. Runkler, "Comparing model-free and model-based algorithms for offline reinforcement learning," *IFAC-PapersOnLine*, vol. 55, no. 15, pp. 19–26, 2022. 6th IFAC Conference on Intelligent Control and Automation Sciences ICONS 2022.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv*, vol. abs/1707.06347, 2017.

VITA

Deniz Yılmaz had taken high school education in Beşiktaş Atatürk Anatolian High School. She received his Bachelor's Degree in Mechatronics Engineering at Istanbul Bilgi University in 2020. She is a M.Sc. Student in Özyeğin University under the supervision of Assoc. Prof. Barkan Uğurlu. She is studying control systems, reinforcement learning algorithms, and robotics. She is also working as a research student at OzU Biomechatronics Laboratory.