Theoretical and Experimental Investigation Of Free-Floating Space Manipulator Motion Control Using Reinforcement Learning

Ahmad Al Ali

A dissertation submitted to
The faculty of graduate studies
In partial fulfillment of the requirements
For the degree of

Doctor of Philosophy

Graduate Program in
Mechanical Engineering
York University
Toronto, Ontario
August 2024

Abstract

This doctoral research is motivated by the need to capture a non-cooperative target in Earth orbits by autonomous manipulators for debris removal. It investigates the critical motion planning problem for a 6DOF free-floating space manipulator using model-free Reinforcement Learning. This problem is caused by dynamic coupling between the spacecraft and robotic manipulator, which significantly affects control and precision in the space environment. This research addresses critical requirements for efficient and effective manipulation in space, including accurate pose alignment between the end-effector and target debris, collision avoidance with both the target and other links of the manipulator and external obstacles, smoothing of joint velocities using optimization terms integrated into the reward function, and adaptation to the high mass ratio between the manipulator and its base spacecraft. Recognizing the imperfection of real-world sensors, this research also incorporates observation noise in the training process to enhance the agent's resilience to noise.

Additionally, the reinforcement learning agent is trained with varying initial conditions that are randomly set at the start of each episode, including the manipulator's initial joint angles, target positions, and obstacle locations. The trained agent can find suitable paths for any target location and avoid obstacles regardless of the manipulator's initial position. The study provides a solid foundation for the application of reinforcement learning in complex free-

floating space robotic operations and offers insights for future missions.

In addition to numerical verification, ground experimental validation is essential. To overcome the difficulty in mimicking the 3D microgravity environment on Earth, this doctoral research has devoted efforts to designing and building a hardware-in-the-loop ground testbed with active gravity compensation in our lab. The testbed involves two industrial 6DOF robotic arms, one robotic finger gripper and various sensing equipment, including cameras and force/torque sensors, to mimic the 6DOF motion maneuvers that a free-floating robotic manipulator or tumbling satellite/target in microgravity. Initial experimental results in motion control, computer vision, and sensing capabilities are presented to show the potential of the testbed. This facility will be an invaluable tool for the future development and validation of space robotic manipulators, ultimately improving their effectiveness and reliability in space missions.

Dedication

I dedicate my dissertation work to my late father, Ibrahim Al Ali.

Your unwavering love, guidance, and belief in me have been the foundation of my success. Although you are no longer with us, your memory continues to inspire and motivate me every day.

This work is a testament to the values you instilled in me — hard work, perseverance, and a thirst for knowledge.

I am forever grateful for the lessons you taught me and the strength you gave me. This accomplishment is as much yours as it is mine.

Thank you, Dad. I miss you dearly.

Acknowledgments

"If I have seen further, it is by standing upon the shoulders of giants."

-*Isaac Newton*

I would like to express my sincere appreciation to those who have contributed to this dissertation during this doctoral journey. This research work would not have been possible without their encouragement and support.

First and foremost, I would like to extend my heartfelt and sincere gratitude to my supervisor and mentor, Prof. George Z. H. Zhu, for believing in me and providing me with an opportunity to complete my Ph.D. dissertation. I appreciate his contributions of time and constructive suggestions to make my work productive. His scientific guidance, continuing support, and constant optimism encourage me to be better at what I do. I benefit greatly from his insightful feedback, rigorous academic attitude, and excellent example.

I am extremely grateful to my supervisor committee members, Prof. Dan Zhang and Prof. Nima Tabatabaei, for their insightful comments and important suggestions at my research evaluation's annual meetings. These comments inspired me to improve my work.

I would also like to acknowledge my fellow group members and friends, Mr. Bahador Beigomi, Mr. El Ghali Asri, Mr. Fuzhen Yao, Mr. Qi Zhang, Mr. Zhengze Liu, Eng. Sukhjinder Lal, Eng. Angela Huang, Dr. Khalid Alzoubi, Dr. Sulaiman Bashabsheh, Mr. Ziad Malawi and many others for companionship, rewarding academic exchanges, and interesting conversations.

Last but not least, I wish to thank my beloved family for believing in me and supporting me all the way. I would like to thank my parents for their wise counsel and sympathetic ear. You are always there for me. I would like to thank my brothers Mohammad, Mahmoud, Amin, and my sister Nada for being part of my foundation and for all their advice and wise words. I would especially like to express my love and gratitude to my mother, Nahla, for her great companionship, constant support, warm encouragement, and never-ending patience in my life.

Thank you, everyone, for your support during my Ph.D. study; I wish you all the best.

Table of Contents

List of Tables

List of Figures

List of Symbols

$\boldsymbol{A}_0$	Base rotation matrix relative to the inertial frame
$\mathbf{A}_e$	Manipulator end effector orientation
$\mathbf{a}$	RL agent actions
$\mathbf{c}_b$	Coriolis effects acting on the base spacecraft
$\mathbf{c}_m$	Coriolis effects acting on the manipulator
d	Position error between EE and target
d_0	Range for successful position alignment
J_i	Manipulator joint pose
$\boldsymbol{J}_b$	Jacobian matrix of the base spacecraft
$\boldsymbol{J}_m$	Jacobian matrix of the manipulator
$\mathbf{J}_g$	Generalized Jacobian Matrix
K_d, K_ϑ	positive reward coefficients for the position and orientation errors
$\mathbf{L}$	Angular momentum, kg.m²/s
$\mathbf{M_b}$	Inertia matrix of the base spacecraft, kg.m²
$\mathbf{M_m}$	Inertia matrix of the manipulator, kg.m²
$\mathbf{M_{bm}}$	Inertial matrix reflecting the dynamic coupling between the manipulator and the base spacecraft, kg.m²
$\mathbf{P}$	Linear momentum, kg.m/s
$\mathbf{p}_e$	Manipulator end effector position

p_l	Penalty for self collision
p_{Ob}	Penalty for external obstacle collision
p_{cone}	Penalty if robotic EE is located outside the safety cone
R_t	Reward given at each timestep
$\boldsymbol{r}_0$	Base position from the inertial frame
r_d, r_ϑ	Sparse rewards at each timestep capture occurs
$\boldsymbol{s}$	RL environment observations
$\mathbf{u}$	Control torque vector of all joints, Nm
$\mathbf{v}_0$	Base linear velocity, m/s
$\mathbf{v}_e$	Manipulator end effector linear velocity, m/s
$\boldsymbol{\omega}_0$	Base angular velocity, rad/s
$\boldsymbol{\omega}_e$	Manipulator end effector angular velocity, rad/s
Θ_{Ad}	Desired joint angles for robot A, rad
Θ_{Bd}	Desired joint angles for robot B, rad
Σ_A	Robot A base frame
Σ_B	Robot B base frame
Σ_I	HIL system inertial frame
Σ_S	Space manipulator base frame
ϑ	Orientation error between EE and target
ϑ_0	Ranges for successful orientation alignment

List of Abbreviations

ADR	Active Debris Removal
AI	Artificial Intelligence
CM	Center of Mass
CPP	Coverage path planning
CSA	Canadian Space Agency
DDPG	Deep Deterministic Policy Gradient
DOF	Degrees of freedom
EE	End Effector
ESA	European Space Agency
GA	Genetic Algorithm
GEO	Geostationary Earth Orbit
GJM	Generalized Jacobian Matrix
GPU	Graphics Processing Unit
HIL	Hardware in the loop
ISS	International Space Station
JAXA	Japan Aerospace Exploration Agency
LEO	Low Earth Orbit
MDP	Markov Decision Process
NASA	National Aeronautics and Space Administration
OOS	On Orbit Servicing
PC	Personal Computer

PDW	Path Dependent Workspace
PIW	Path Independent Workspace
RL	Reinforcement Learning
Yolo	You only look once

Chapter 1 INTRODUCTION AND JUSTIFICATION

Summary: This chapter introduces and reviews the applications and research activities on the space robotic manipulator system. It outlines the research objectives of this dissertation and details the methodologies employed. Finally, the chapter presents the structure of the dissertation and includes a comprehensive list of publications resulting from this doctoral study.

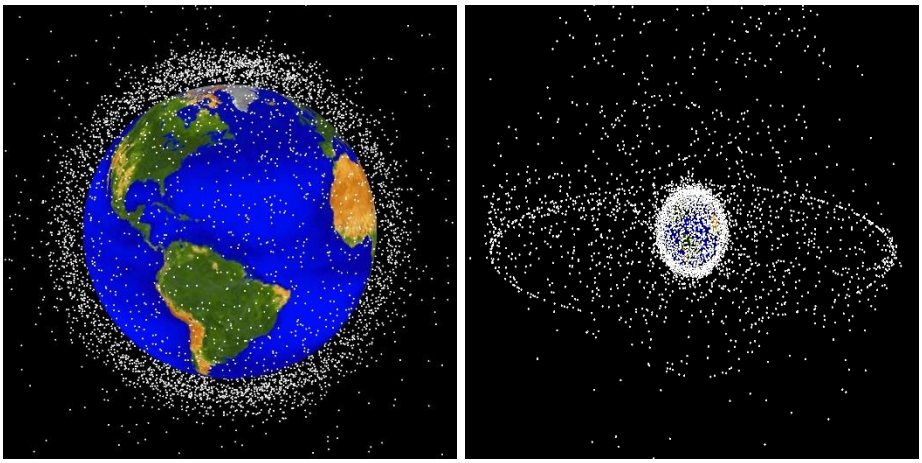
1.1 Background

Space exploration has always been the frontier of scientific discovery and technological innovation. In human history, space technologies have enabled us to make our dreams a reality, from space exploration to the colonization of different planets. In 1957, the first artificial satellite in the world was launched into space 'Sputnik 1', marking the beginning of humanity's ambitions in space [1]. Since then, tens of thousands of satellites have been sent into space for various objectives, including communications, navigation, and Earth observation. These satellites have a limited lifespan because they eventually run out of fuel and could drift in the Earth's orbits for extended periods after decommissioning. The density of space debris like spent rockets, malfunctioned or end-of-mission satellites, and other manmade

objects with their fragments in Earth's orbits is about to reach a dangerous point, where the density of debris may increase by cascade colliding with each other or functional satellites. This cascade effect is known as the Kessler Syndrome [2]. This effect will potentially compromise all future space missions if no preventive measurements are taken. They will also threaten the still functioning satellites in orbit and the lives of astronauts living in space stations.

Although space debris mitigation measurements have been implemented in the industry for decades, the total number of space debris is continuously increasing. Currently, more than 5,720 satellites are still in space [3]. Research has suggested that active debris removal (ADR) must be done without delay, removing five large pieces of debris per year from orbit [4] to stop the increasing trend of the space debris population.

Around half of the defunct satellites can divide into space debris, resulting from collisions and explosions of satellites and other fragmentation events. There are around 27,000 space debris in the low earth orbit (LEO) and geostationary earth orbit (GEO) objects (see Figure 1.1), which are regularly tracked by the United States Space Surveillance Networks. Space debris seriously threatens the safety of our spacecraft, satellites, and the International Space Station (ISS), and the rising population increases the potential danger [5].



(a) (b)

Figure 1.1 Distribution of cataloged objects in space[1]:

(a) LEO region; (b) GEO region

Many approaches have been devoted to space debris removal to reduce the potential collision risk of spacecraft with debris, such as the space robotic arm, space tether, gripper mechanism, and some contactless approaches [6,7]. Among them, space robotic manipulators are one of the more appealing approaches.

Due to its high technical readiness level, space robotics are considered one of the most practical applications for on-orbit servicing (OOS) missions, such as docking, repairing, and orbital debris removal [5]. This is because robotic manipulators are well-matched to execute highly repetitive tasks that would be too time-consuming, risky and expensive for astronauts to perform. Space robots also require much less infrastructure than human astronauts

[1] Retrieved from https://www.orbitaldebris.jsc.nasa.gov/photo-gallery/ in July 2024

(such as life support systems), making them preferable in space [8].

Numerous nations have dedicated substantial resources to space missions to explore the potential of space robotic applications. Among these prominent organizations are The National Aeronautics and Space Administration (NASA) in the USA, the European Space Agency (ESA), the Canadian Space Agency (CSA), and the Japan Aerospace Exploration Agency (JAXA), alongside various space companies and academic institutions. These collective efforts have yielded significant advancements in space robotics, exemplified by groundbreaking projects.

Among the first projects was the Shuttle Remote Manipulator System (SRMS), also known as the Canadarm [9], developed by the Canadian Space Agency (CSA) and MDA. Since 1981, the Canadarm was used in NASA's space shuttle program until decommissioned in 2011. The robotic arm deployed, captured, and repaired satellites, positioned astronauts, maintained equipment, and moved cargo. It was 15 meters long, with 6 DOF at its disposal, one camera attached to its wrist, but no touch sensors. It was remote-controlled and could place any load within 5cm of the target.

Its successor, the Space Station Remote Manipulator System (SSRMS), also known as the Canadarm2 [9], was launched in 2001 and played an essential role in the construction of the International Space Station. It remains on the station to conduct maintenance, move equipment and supplies, support

astronauts working in space, and handle payloads until this day. The arm was 17 meters in length with 7DOF, four colored cameras attached to it, and a force-moment sensor. In June 2002, when both Canadarms were in operation, they were used in tandem, a process nicknamed the "Canadian handshake."

The next-generation Canadarm3 is currently under development and will be permanently stationed on the Lunar gateway [9]. It is designed to be 8.5 meters long, with 7DOF, and includes force-moment sensors. It will be primarily controlled autonomously but can also be controlled from the ground or by astronauts on the Lunar Gateway.

In 2008, the Dextre (Special Purpose Dexterous Manipulator) was installed on the International Space Station [10]. Dextre has two arms over three meters long, each arm with 7DOF allowing movement in all directions, and each gripper has sensors that give it a human-like sense of touch. To this day, it is the only robot that can repair itself in space.

The Japanese Experiment Module (JEM) Remote Manipulator System (JEMRMS) is one of the National Space Development Agency of Japan's (NASDA) contributions to the ISS [11]. Launched in 2008, the arm was 10 meters long and had 6DOF permanently attached to the JEM pressurized module.

The European Space Agency (ESA) has developed its European Robotic Arm (ERA), which will be attached to the ISS [12]. It has a relocatable base

intended to install solar panels on the Russian Solar Power Platform, handle payloads, and carry out maintenance tasks. The arm can reach up to 11 meters with 7DOF and can be operated by the ISS crew.

The Experimental Test Satellite VII (ETS-VII), nicknamed Kiku-7 by the Japanese, was launched in 2007 and is considered the first space robotic OOS demonstration [13]. Its objective was to conduct different space experiments, including capturing and repairing a target satellite. The arm was two meters long with 6DOF.

In the same year, 2007, the Defense Advanced Research Projects Agency (DARPA), in conjunction with Boeing, successfully launched and accomplished the Orbital Express mission [14]. It demonstrated different technologies for spacecraft servicing, including autonomous rendezvous, docking, in-orbit refueling, and robotic replacements. During the mission, the robotic arm autonomously transferred a supplemental battery and backup computer to a target spacecraft designed to be serviced. The success of this mission proved the concept of small base-mounted robotic arms used for different OOS and is the closest mission to be studied in our research.

In 2010, NASA developed a set of concept missions to study satellite servicing; these missions are called the Notional Missions (NM) and are comparable to the orbital Express [15]. The GEO Supersync (NM1) mission aimed to develop a servicing spacecraft that could capture and control

noncooperative satellites, relocate them and dispose of them. The target satellite was assumed to tumble at 0.25 °/s in each axis. It was designed with four identical 2m robotic arms at the front corners connected to the inner main structure to match the dynamic load in the initial capturing stage.

The GEO Refueling (NM2) objective was to develop two spacecraft: a refueler spacecraft and an on-orbit fuel depot that would provide necessary fueling service to GEO satellites with extended life. The spacecraft included two identical 2 m robotic arms and one refueling probe.

The LEO Refurbishing (NM3) aimed to develop a service spacecraft called the dexterous service module (DSM) for different servicing objectives. The main highlight of the DSM design was the two robotic arms in the front of the DSM, which were 14m and 7m long. The longer arm was designed with two links to guide the docking, separation, and transfer of astronauts to the service position. The shorter arm was developed for dexterous manipulation via a specialized end effector.

The HEO Human/Robotic Refurbishing (NM5) objective was to develop a human/robot telescope servicer (HRTS) that would enable astronauts and robots to service the Advanced Technology Large Aperture Space Telescope (ATLAST) being developed by NASA. It was equipped with four robotic arms: two 20-meter-long arms were used to capture the ATLAST and introduce docking, while two 2-meter-long arms were used to conduct detailed

manipulation.

In 2011, DARPA introduced the Phoenix mission [16]. Drawing inspiration from the phoenix rising from the ashes, the Phoenix mission aimed to recover durable components, including antennas, from discarded satellites and then conducted integration with on-orbit assembly to obtain a new spacecraft. The key component of the servicer was the FREND arm, which could meet the multiple demands of GEO conditions to provide operations including assembly, disassembly, maintenance, repair and refueling.

The previous space robotic missions conducted are summarized and listed in Table 1.1 .

Table 1.1 List of space robotic missions

Mission	Robotic Arm DOF	Robotic Arm Length	Robotic Arm Mass	Spacecraft Base Mass	Base-Arm Mass Ratio
Canadarm [9]	6	15 m	410 kg	78,000 kg	0.53%
Canadarm [9]	7	17 m	1,497 kg	408,000 kg	0.37%
Canadarm3 [9]	7	8.5 m	715 kg	-	-
Dextre [10]	3	3.7 m	1,710 kg	408,000 kg	0.42%
JEMRMS [11]	6	10 m & 2 m	780 kg	408,000 kg	0.19%
European Robotic Arm [12]	7	11 m	630 kg	408,000 kg	0.15%

ETS-VII [13]	6	2 m	45 kg	2,450 kg	1.84%
Orbital Express [14]	6	3m	71 kg	953 kg	7.45%
Rotex [15]	6	1 m	-	78,000 kg	-
GEO Supersync (NM1) [15]	7	2 m	77 kg	3,694 kg	2.08%
GEO Refueling (NM2) [15]	7	2 m	77 kg	1,894 kg	4.06%
LEO Refurbishing (NM3) [15]	3	14 m & 7 m	-	4,350 kg	-
HEO Human/ Robotic Refurbishing (NM5) [15]	-	20 m & 2 m	-	31,870 kg	-
Human/ Robotic Assembly (NM6) [15]	-	-	-	Up to 7 tons	-
Pheonix [16]	7	1.8m	77 kg	-	-

1.2 Justification of Research

Space technologies have materialized our wildest dreams, from exploring space to the potential colonization of different planets. As we venture further into the cosmos, the need for sophisticated tools and technologies becomes increasingly apparent. Among these tools, space robotic manipulators

stand out as crucial assets for various missions, including on-orbit servicing, in-space assembly of large structures, and space debris removal.

The proliferation of space debris poses a significant risk to operational spacecraft and future missions. Robotic manipulators can be used to capture and remove debris from orbit, helping to mitigate the growing problem of space debris and safeguarding valuable assets in space.

By developing innovative manipulator designs, control algorithms, and operational strategies, we can unlock new opportunities for exploring, discovering, and utilizing space resources.

Designing spacecraft servicing systems to minimize mass is crucial for cost-effective development and launches. The emergence of a new robotic manipulator integrated into compact satellite bases heralds a promising future for tasks like space debris removal and various space operations. However, this innovation introduces a challenge: the manipulator's movement can perturb the base due to the dynamic coupling between the manipulator and the spacecraft, stemming from momentum conservation principles. In contrast to the well-established Canadarm on the ISS, where the base spacecraft's mass vastly exceeds that of the manipulator, disturbances caused by the manipulator's motion are practically negligible. However, with the smaller base mass of the satellite-mounted manipulator, even slight disturbances can impact operations significantly. While the advantages of satellite-mounted

robotic manipulators are substantial, meticulous study of their motion planning and control algorithms is imperative. These studies must carefully account for dynamic coupling to ensure optimal performance and mission success.

Experimental validation for any research is never without difficulties. This is truer for space applications where access to space for experimental validation is limited and expensive. Launching hardware to space for testing purposes requires careful planning and coordination, often involving collaboration with space agencies or commercial launch provider. Despite this, researchers and engineers strive to overcome these challenges through innovative approaches, such as ground testbeds that can mimic the space environment accurately. Continuous improvement in validation methodologies and technologies is essential to ensure the reliability and effectiveness of space manipulators for future space exploration and applications. With that in mind, this study aims to design and build a ground testbed that can mimic the 6DOF motion of a space robotic manipulator in micro-gravity environment.

1.2.1 Space Manipulator Control

1.2.1.1 Challenges of Space Manipulator Control

Controlling space robotic manipulators poses several challenges due to the unique environment and characteristics of space operations:

(i) **Limited Communication**. Communication delays and bandwidth constraints between the ground control and the robotic manipulator can hinder real-time control, requiring the development of robust autonomous control algorithms. Furthermore, real-world sensing capabilities are always subject to noisy signals.

(ii) **Dynamic Coupling.** Dynamic coupling between the manipulator and the spacecraft base can lead to undesired disturbances during operation, necessitating sophisticated control strategies to mitigate these effects.

(iii) **Safety and Collision avoidance.** Ensuring the safety of space assets and avoiding collisions with other objects, including spacecraft and debris, is critical during robotic manipulation tasks, requiring obstacle and collision avoidance algorithms.

(iv) **Power and Resources.** Spacecraft have limited power and resources available for control systems, imposing constraints on control algorithms' complexity and computational requirements.

1.2.1.2 Limitations of Existing Studies

To date, existing studies on space robotic manipulators face several limitations, some of which include:

(i) **Simplified Simulation Environments**. Many studies rely on

simplified simulation environments that may not accurately represent the complexities of space conditions, such as microgravity and dynamic coupling. These simplifications can lead to discrepancies between simulated results and real-world performance.

(ii) **Assumptions about System Dynamics.** Control algorithms often rely on simplified models of robotic manipulators and their dynamics. However, these assumptions may not fully capture the complexities of the space environment, such as dynamic uncertainties, external disturbances, and noisy sensor input.

(iii) **Single-Task Optimization.** Many studies focus on optimizing control algorithms for a specific task or scenario, such as performing a single task or grasping the same target. However, space robotic manipulators are expected to capture different targets from various positions, requiring versatile and adaptive control strategies.

(iv) **Integration with Artificial Intelligence.** Although AI techniques hold promise for enhancing the autonomy and adaptability of space robotic manipulators, their integration into control algorithms is still in its early stages. Challenges include data scarcity and safety considerations in autonomous decision-making.

(v) **Limited Experimental Validation**. While simulation studies are valuable for theoretical analysis and initial testing, they often lack

comprehensive experimental validation in real space environments. Limited access to space and prohibitive costs associated with space missions make it challenging to validate control algorithms under actual operating conditions.

1.2.2 Space Manipulator Experimental Validation

This dissertation details our efforts to design and construct a hardware-in-the-loop ground testbed with active gravity compensation at York University. The testbed can perform the 6 degrees-of-freedom (DOF) motion maneuvers that a free-floating robotic manipulator or a tumbling satellites/ target can achieve in microgravity conditions. It incorporates two industrial-sized 6DOF robotic arms, along with a number of sensing equipment such as cameras and force/torque sensors. This work also includes the initial experimental results in motion control, computer vision, and sensing equipment capabilities. The facility will serve as an invaluable tool for the development and validation of space robotic manipulators capture maneuvers, ultimately enhancing their effectiveness and reliability in space missions.

1.3 Objectives of Proposed Research

To address the existing challenges and limitations highlighted in the previous section, this dissertation studies the dynamic behavior and control of

space robotic manipulators and their application to remove or retrieve space debris. The research objectives of this study are presented as follows:

(i) Study the control problem of a space robotic manipulator system for autonomous guidance and capture of a target.

(i) Develop the framework for using artificial intelligence (AI) to solve the space robotic manipulator motion control problem.

(ii) Investigate different mission scenarios, constraints and optimization criteria that can be integrated into the AI motion planning solution.

(iii) Verify the effectiveness of the proposed control strategy using a realistic simulation environment in microgravity, under dynamic conditions, and with noisy inputs.

(iv) Construct a fully operational micro-gravity experimental system in our lab using a Hardware-in-the-Loop testbed to mimic the space environment and test various control strategies.

1.4 Methodology of Approach

The methodology adopted in this dissertation initiates an in-depth exploration of the complex dynamics governing space robotic manipulator systems. It delves into the primary challenges encountered in controlling these systems, particularly emphasizing the dynamic coupling between the base and manipulator, which often leads to dynamic singularities. Subsequent

investigations focus on adapting and extending various robotic control and optimization techniques to address the motion control aspects of space robotic manipulators.

A pivotal aspect of this research involves the development of a comprehensive framework tailored to address the guidance and control dilemmas inherent in space robotic manipulators, leveraging artificial intelligence, specifically reinforcement learning. An in-depth examination of reinforcement learning algorithms is done, necessitating a profound understanding of their core components. Integrating the space manipulator system into the reinforcement learning framework demands a nuanced comprehension of these components to devise viable solutions to the control problem.

The framework comprises essential elements, notably the agent, a neural network functioning as the system's cognitive center. Upon thorough training, this agent gains the capacity for autonomous control over the robotic manipulator. Additionally, the meticulous design of the reward function is imperative and tailored to steer the system towards predefined objectives. These objectives primarily encompass guiding the robotic end-effector towards targets, achieving precise position and orientation alignment for soft capture, circumventing collisions between robotic links or obstacles, and enhancing path optimization based on specified criteria and constraints.

Furthermore, the environment within which the reinforcement learning agent operates is meticulously constructed to mirror real-world conditions. It encompasses the space manipulator system and its constituent elements, integrating dynamic targets, obstacles, and realistic sensor data, often subject to noise. This environment serves as the training ground for the agent to explore and ascertain feasible solutions.

Ultimately, the efficacy and robustness of the developed space robotic manipulator system are validated through practical demonstrations on our ground testbed. These demonstrations include the motion of a tumbling target satellite in space, the space robotic manipulator motion obtained from a trained agent to achieve capture, and different sensory data acquisition. This testbed features two industrial robotic arms configured in a Hardware-in-the-loop arrangement, providing a platform for careful evaluation and validation of the system's performance.

1.5 Outline of this Dissertation

The dissertation includes eight chapters. Chapter 1 gives an introduction and justification. Chapter 2 presents a comprehensive review of the literature on various path planning and control methods used in fixed-base robotic systems. It also includes an in-depth examination of free-floating manipulator systems and explores the experimental validation technologies

employed in space manipulators. Chapter 3 provides the mathematical formulations for studying the free-gloating manipulator control and the HIL ground testbed modeling. Chapter 4 focuses on reinforcement learning, the mathematical formulas that govern the training, and how it is implemented to achieve our desired outcomes. Chapter 5 includes the simulation results and discussion. Chapter 6 focuses on multi-scenario reinforcement learning and includes the simulation results and comparisons. Chapter 7 describes our experimental testing and validation of the HIL space robotic manipulator system's feasibility and effectiveness in maneuvering a tumbling target in a microgravity environment provided by the 6DOF industrial robots. Finally, Chapter 8 summarizes the contributions of this research and states the potential future research aspects.

1.6 List of Publications

The following is a complete list of conference and peer-reviewed journal publications associated with this dissertation.

1. Zhu, Z H., **Al Ali, A.**, Review on current path planning algorithms for autonomous robotic applications. *CSME Congress 2021*, online, June 27-30, 2021. (Reference Paper A)
2. **Al Ali, A.**, Zhu, Z H., Development of 6DOF hardware-in-the-loop testbed by dual robotic manipulators with active gravity compensation. *33rd*

AAS/AIAA Space Flight Mechanics Meeting, Texas, USA. January 15-19, 2023, (Reference Paper B)

3. **Al Ali, A.**, Zhu, Z H., Development of 6DOF hardware-in-the-loop testbed by dual robotic manipulators with active gravity compensation. *CSME Congress 2023, Sherbrooke*, Canada. May 28-31, 2023. (Reference Paper C)
4. **Al Ali, A.**, Zhu, Z H., Path planning and control of free-floating space manipulators using deep reinforcement learning. *CSME Congress 2024, Toronto, Canada. May 26-29, 2024.* (Reference Paper D)
5. **Al Ali, A.**, Zhu, Z H., Towards the development of 6DOF hardware-in-the-loop ground testbed for space robotic manipulators. *45th COSPAR Scientific Assembly 2024, Busan, South Korea*. July 13-21, 2024. (Reference Paper E)
6. **Al Ali, Ahmad**, and Zheng H. Zhu. "Reinforcement learning for path planning of free-floating space robotic manipulator with collision avoidance and observation noise." *Frontiers in Control Engineering* 5 (2024): 5:1394668. (Reference Paper F)

   Doi: 10.3389/fcteg.2024.1394668
7. **Al Ali, Ahmad**, Jian-Feng Shi, and Zheng H. Zhu. "Path Planning of 6-DOF Free-Floating Space Robotic Manipulators Using Reinforcement Learning." *Acta Astronautica,* Vol. 224, pp. 367-378, (2024). (Reference

Paper G)

Doi: 10.1016/j.actaastro.2024.08.015

8. **Al Ali, Ahmad**, Bahador Beigomi, and Zheng H. Zhu. "Development of 6DOF Hardware-In-the-Loop Ground Testbed for Autonomous Robotic Space Debris Removal." (2024). (Reference Paper H)

   Doi: 10.20944/preprints202408.1425.v1

9. **Al Ali, A.**, Zhu, Z H., Multi-Scenario Reinforcement Learning for Path Planning of Free-Floating Space Robot with Collision Avoidance and Observation Noise, final editing for submission. (Reference Paper I)

Chapter 2 LITERATURE REVIEW

Summary: In this chapter, we review the literature on different robotic path planning and control methods, extending these methods to space robotic manipulators and using reinforcement learning to tackle this problem.

2.1 Robotic Path Planning and Control

Robotic systems are growing increasingly popular each year, with a wide variety of applications in almost all aspects of our lives. These applications range from mobile, industrial, surgical, and even space robots. Now, it might seem that the wide use of robots in modern life implies that the motion-planning problem has already been solved. However, this is far from true. With increasingly more ideas being implemented by robotics systems in different applications, the desire for autonomous path planning methods is at an all-time high. This section will review the most up-to-date autonomous path planning techniques. Furthermore, we will list the advantages and disadvantages of these methods and conclude this review with some of the currently open research problems in the field.

In general, the goal of path planning algorithms is to achieve the pose alignment between robotic EE and target; additionally, this path can be

optimized by some criteria, such as the least amount of energy (the closest path), reducing the travel speed, and processing time (computation). More importantly, the path can avoid Obstacles and collisions. This section will contain the following methods that are currently being researched in different robotic systems: Coverage Path Planning, the Random Walk, Sampling-Based Motions Planning (this includes the Rapidly Exploration of Random Trees and the Probabilistic Road Map), the Artificial Potential Field method, Greedy Algorithms (such as Dijkstra's, A*, etc.), Genetic Algorithms, Swarm intelligence (such as: Ant Colony Optimization, Particle Swarm Optimization, etc.), and finally the Machine Learning Algorithms, and Reinforcement Learning; two of the most hot research topics currently being explored in robotic systems.

Coverage path planning (CPP) determines the best path out of all points between initial and final states while detecting and avoiding obstacles in a target environment [17]. The goal of the CPP algorithm is to compute the optimal path and project a collision-free trajectory within an area of interest (AOI). CPP algorithms can be classified into two approaches: classical algorithms and heuristic-based algorithms, summarized in Figure 2.1.

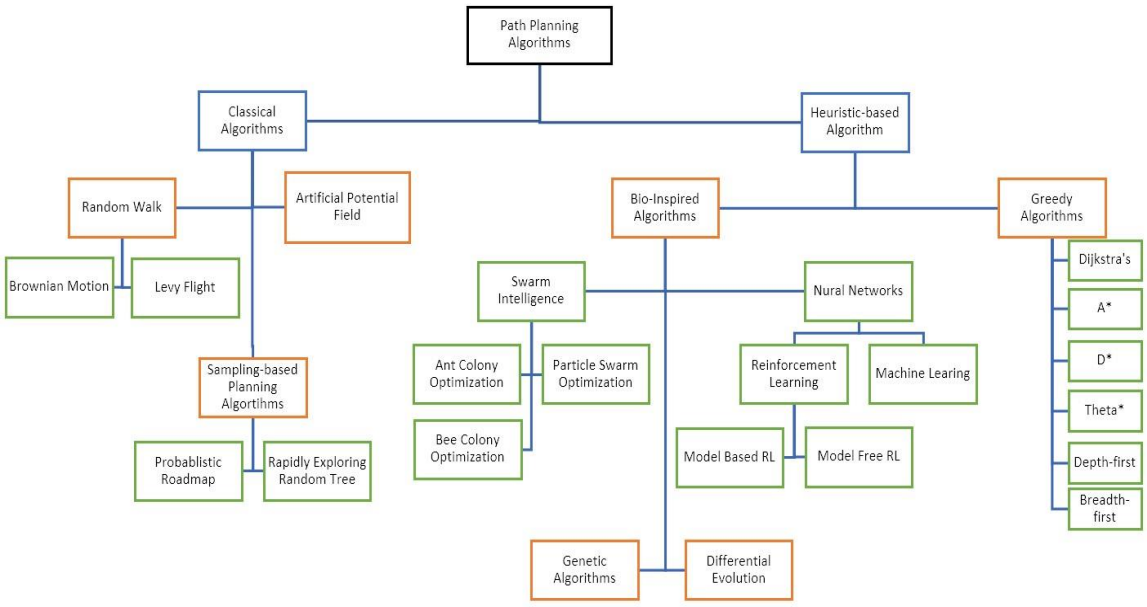


Figure 2.1 Coverage Path Planning Algorithms

The Random Walk (RW) method is used to solve one of the complex problems in robotics: environment exploration. The main goal is to explore an unknown area to achieve collision-free path planning. The cleaning robot is the best example of an autonomous robot using random walks [18]. There are two main types of RW:

a) Fixed linear. A robot with a fixed linear approach randomly turns at an angle and frequently moves at a straight line until it collides with the wall or obstacle boundaries. These are shown in Figure 2.2.

Figure 2.2 Motion path of RW, spiral, and S-shape algorithms [18]

b) Variable step. This method computes a set of RW directions based on the probability distribution of step lengths taken by the robot. The variable step includes two methods: Brownian motion (BM), where the robot based on BM repeatedly moves in a step length with a normal distribution and randomly turns in a direction, and Levy flight, where the robot of LF travels a distance in which the step length depends on Lévy's probability distribution.

For a small environment, one robot may be enough to explore and map an unknown area; however, for large environments, it is inefficient to have one robot to explore it. So, multi-robots are preferred; to be more specific, swarm robots are used widely for this type of area exploration because of their robustness, flexibility, and scalability [19].

In Sampling-Based Planning algorithms, instead of mapping the whole environment (which requires more computation), the algorithm will map the configuration space using randomly generated nodes. These sampling-based planning algorithms include:

a) Probabilistic Roadmap (PRM). This algorithm consists of two phases [20]. First, the planning phase randomly generates several nodes in the robot configuration space and checks if they are collision-free (if they are in the free configuration space, they are kept, but if they are in the obstacle space, they are discarded). Then, it connects the remaining nodes with a straight line. Secondly, the query phase creates a path between the starting and final positions.

The connection between nodes is called the Local Path Planner, and instead of simply using a straight line, PRM can be optimized by combining it with a greedy search algorithm such as A* [21]. In [22], the probabilistic road map with an improved A* algorithm was used for the path planning of an industrial robot with 6Dof. The improved A* method was implemented to obtain a better path (which is usually zig-zagged due to the random placement of nodes). Figure 2.3 shows the difference between paths. Furthermore, the number of nodes required was less than the old A* method.

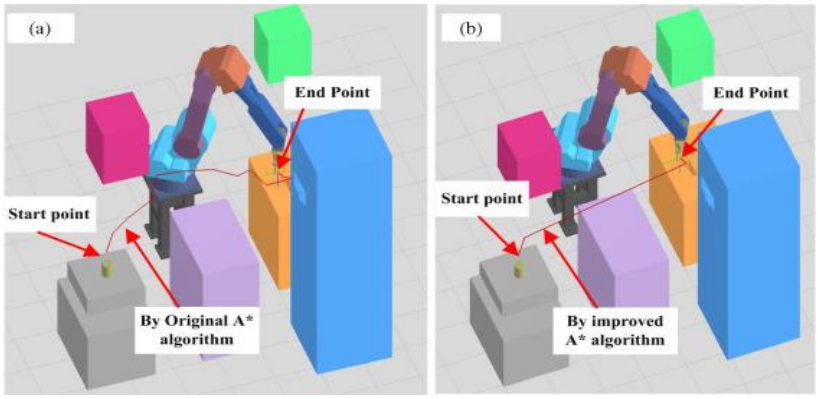


Figure 2.3 a) path obtained by A*. b) path obtained by improved A* [22]

b) Rapidly exploring random tree (RRT): This algorithm will construct a graph to search and explore the configuration space using a tree-like form. The tree incrementally grows from the starting point to the final point. First, a configuration is randomly selected in the configuration space; if it lies in the free obstacle space, a connection is attempted to the nearest point in the tree [23].

For single query problems, RRT is faster than PRM. It does not need to sample the configuration space and construct a roadmap (skipping the learning phase). Another good improvement on the RRT was done in [24]. Using two trees, one tree at the starting point rapidly explores towards the final point, and another at the final point rapidly explores towards the start. As they grow towards each other, making a connection on both trees to generate the shortest path between them.

Although the path generated by the RRT is fast, it is not the optimized solution for the path planning problem. An improved version called RRT* can solve that. The RRT* implements the A* algorithm to get the optimal path from the starting point to each new node on the tree, meaning that at every new node in the tree, the optimal path is found from the start point to that node. Therefore, when the tree reaches the final point, it will generate the optimal path from the start to the final positions [25]. This gives us the optimal path, but as you can imagine, it will take more time to compute (at every new

node, the optimal path to that node must be computed).

In [26], path planning with obstacle avoidance for a 6 DOF industrial robot manipulator was done using three different RRT-based algorithms and a comparison between them. This study implemented the RRT method first, then the RRT*, and thirdly, optimized the path created using Bezier curves (the Bezier curve is a tool for smoothing, which can reconstruct the local route without changing the shape of the whole path). The different results can be seen in Figure 2.4.

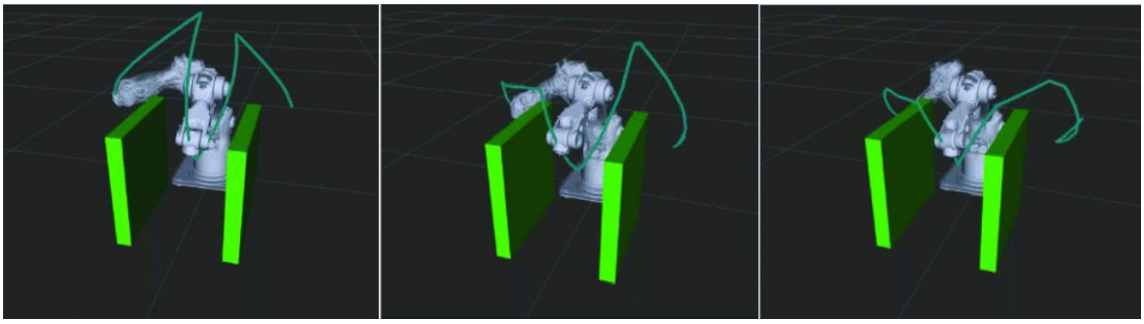


Figure 2.4 Path obtained by RRT, RRT*, and Bezier curve [26]

The Artificial Potential Field (APF) method is mainly used for obstacle avoidance while planning a path towards the goal position. An artificial (fictional) attraction force is created at the goal position to "pull" the robot towards it, and another artificial (fictional) repulsive force is created at the obstacle to "push" it away from it. The resultant force will create an optimized path that will reach the goal positions and, at the same time, avoid obstacles along the path [27].

The artificial potential field method has been widely used in UAV trajectory planning because of its simplicity, high efficiency, and smooth trajectory generation [28]. Furthermore, this method has gained popularity for obstacle avoidance path planning of industrial 6DOF manipulators [29–31]. In these papers, the artificial potential method, typically implemented in 2D space, has been extended to 3D space. In [32], an improved artificial potential field method was used in the path planning of a 6DOF, a free-floating space manipulator, to move through a window-shaped obstacle and avoid a collision. This was done by setting the square-shaped obstacle first and then implementing the APF. Furthermore, calculating the distances between the robot EE plus links and the window will create a path to ensure the robot avoids the obstacle. Finally, the EE path and attitude were continuously tracked using the space robot Generalized Jacobian Method. Figure 2.5 below shows the EE path result of this study.

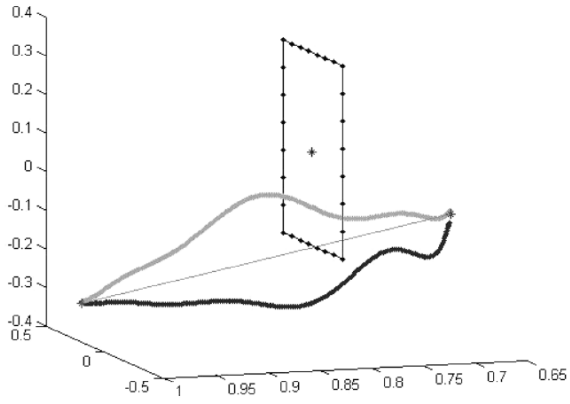
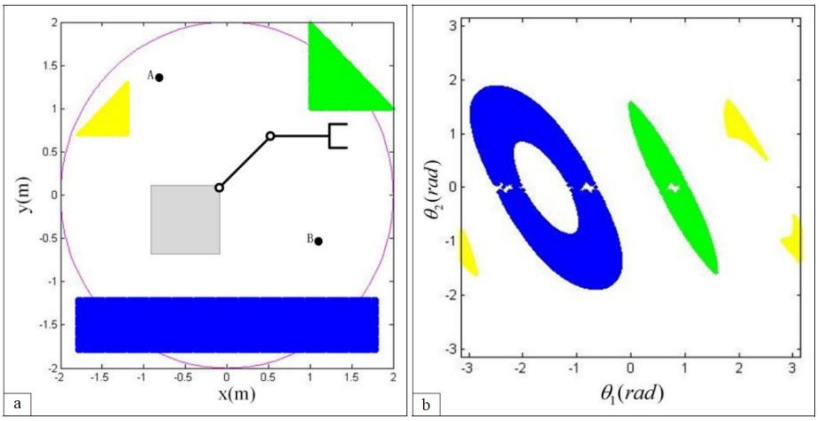


Figure 2.5 End Effector trajectory using APF and extended APF [32]

When using the APF method (setting the attraction force at the goal location and repulsive force along the window obstacle), the EE path would go under the window (black line in the figure) and cause a collision between the robot manipulators' links and the obstacle. To overcome this, the goal position (attraction force) was placed in the middle of the window. Then, after the EE passed through the window, the goal position was removed from the middle and placed at the final goal position.

A study was done in [33]. In this case, obstacle avoidance using the artificial potential field method was implemented on a planar 2DOF free-floating space manipulator (in 2D space). In this study the objective was for the robot to reach its goal while avoiding obstacles along the path, and to minimize the dynamic coupling effect that exists in a free-floating robot manipulator. This was done by Creating a hybrid map in configuration space as seen in Figure 2.6.



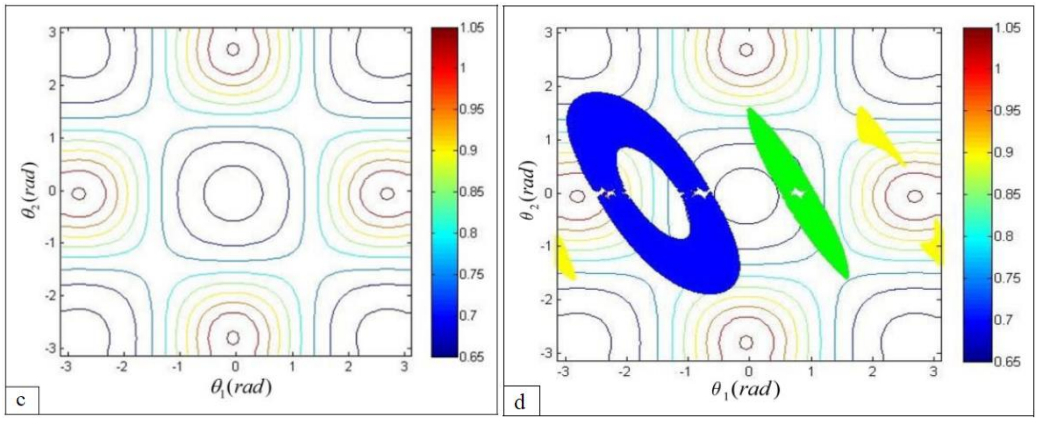


Figure 2.6 a) obstacle map in Cartesian space. b) obstacle map in configuration space. c) coupling effect map in configuration space. d) hybrid map in configuration space [33]

Greedy Algorithms are common path finding algorithms, they obtain the best possible local path choices at each node in order to find the best global path solution. greedy algorithms are simple, easy to implement, and generally fast. These path finding algorithms can be categorized into two types [34].

The informed search algorithms. where the algorithms take into consideration the location of the destination relative to existing location (such as Dijkstra's algorithm, A*, D*, and Theta*). The Uninformed search algorithms, where the algorithm does not have the concept of location of the destination relative to existing location (such as Depth-first, and Breadth-first

Genetic Algorithms (GA) are algorithms inspired by nature (Darwinian evolution) in order to obtain the best solution for an optimization problem. This involves an initialization method, variation operators (crossover and mutation),

and a fitness function to evaluate each chromosome. the goal is to minimize/ maximize this cost function. The GA begins by choosing an initial population which represents the possible solutions (chromosomes) to the problem to be optimized; this is done by an initialization method or at random. Then each possible solution (chromosome) is evaluated using the cost function to determine its quality. Next a genetic operator will choose the parents which can reproduce (depending on their value). After that crossover is applied to create new offspring by recombining data from the two parents selected in the previous step. Another genetic operator called mutation is used to guarantee the diversity of the population by changing the genetic structure of some individuals according to a mutation rate. This evolutionary cycle is repeated until the stopping criteria is satisfied [35]. In this paper the GA based approach was used for path planning of autonomous mobile robot.

Genetic Algorithms can be further improved by using more than one cost function. In [36] a multi objective genetic algorithm was used for optimization of a redundant serial robot to be used in high-quality pipe cutting. The first objective function was used to reduce vibration effects, whereas the second objective function was used to get the optimal motion.

Swarm Intelligence is the method of solving optimization problems inspired from the behavior of different living organisms. It comes from the collective intelligence that comes from individuals working together in a

swarm. The main goal of swarm intelligence is to solve optimization problems with a probability-based search algorithm. The three main types of swarm intelligence are: Particle swarm optimization (PSO), ant colony optimization (ACO), and bee colony optimization (BCO).

PSO is one of the most popular and used swarm optimization methods. It gained increasing popularity among researchers as a robust and efficient technique for solving difficult optimization problems. In PSO each particle in the swarm represents a possible solution that moves through the problem search space looking for the optimal solution. Each particle transmits its current position with its neighbouring particles. The position and velocity of each particle is adjusted depending on 3 main components [37], The particles velocity (rate of change), the difference between the particles current position, and the best position it has found so far, and the difference between the particles current position, and the best position found so far by its neighbours.

In [38] the Trajectory planning of a 7 DOF redundant free-floating space robot was done using Particle Swarm Optimization (PSO).

Today, a wide range of different rule-based AI systems are being implemented in autonomous robots without human intervention [39]. However, these types of deterministic algorithms cannot be adapted to a dynamic environment and are superseded by Machine Learning based AI that uses neural networks. Figure 2.7 shows the types of Machine Learning.

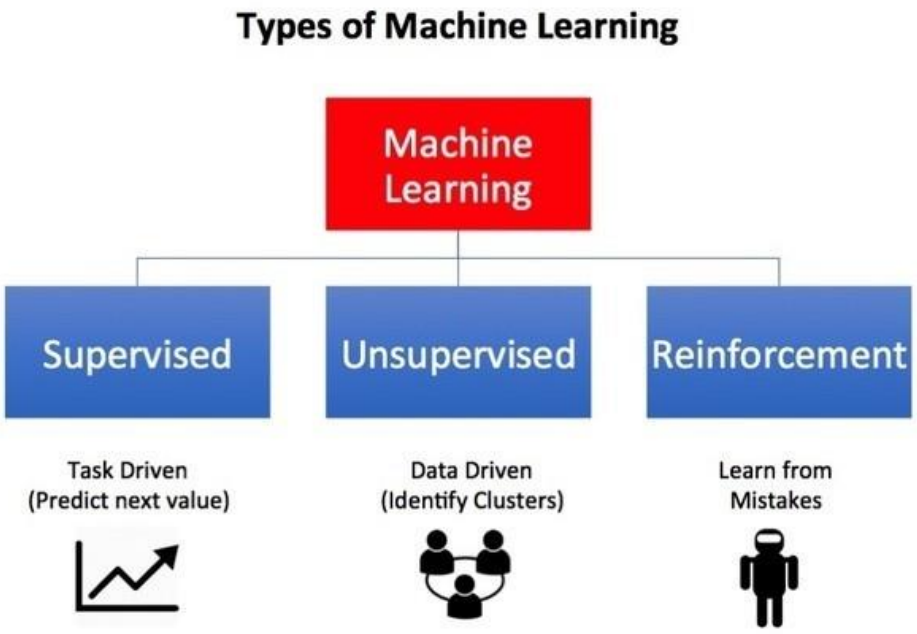


Figure 2.7 Types of Machine Learning Algorithms

Machine Learning (ML) classifies, identifies patterns, and makes predictions about the input data; they learn from examples they have been shown, such as a training set, correlating data with labels, to create a model that can classify anything. The problem with implementing this in an autonomous space debris removal robot is the lack of training sets; there is very limited available data for ML training.

Reinforcement Learning (RL) is proposed in order to overcome this problem; it has the ability to learn a policy for decision-making by interacting with the environment while using a pre-defined reward function instead of relying solely on input data sets [40]. This happens by allowing a piece of

software called an agent to explore, interact with, and learn from the environment. The agent can take an action which affects the environment, changing its state, and the environment then produces a reward for that action.

However, reinforcement learning (RL) has its drawbacks. Since the agent learns through exploration and interaction with the environment, it must explore all possible outcomes, including low-reward areas. This results in slower learning approach, as the agent spends time exploring suboptimal regions during the training process. Another disadvantage is the high risk of damage; the agent will explore a random or unpredictable task.

To address these challenges, the training process can be conducted in a simulated environment, with the trained agent later deployed in the real world. For effective training with minimal discrepancies between simulation and reality, it is crucial to build an accurate model of the physical environment. The model refers to the different dynamic states of an environment and how these states lead to a reward. By doing so, the RL will need a smaller number of interactions between the robot & environment and will have a faster approach to the optimal solution. Furthermore, it will not explore and random or unpredictable tasks that can damage the robot and its surroundings.

These methods, implemented algorithms, robot type used, advantages, and disadvantages are summarized in Table 2.1. With more detailed literature on Reinforcement in later sections.

Table 2.1 Summary of Path Planning Algorithms

#Ref	Algorithm	Method	Robot	Advantages	Disadvantages
[18] [19]	Random Walk	- RW, spiral, and S-shape algorithm. -Improved RW	-Vacuum cleaning robot. -swarm robots	-Simple and low computation cost. -Can explore unknown area.	-Less coverage efficiency with obstacles. -Overlapping of explored areas.
[20] [21] [22]	Probabilistic Road Map (PRM)	-PRM -PRM with A* local path planner. -improved PRM with A*	-mobile robot -industrial robot manipulator	-low computation (only samples of environment). -Can be used for high dimensional configuration space.	-Limited coverage near boundaries & obstacles. -Not suitable for dynamic environment. -Not global optimal solution.
[23] [24] [25] [26]	Rapidly exploring random tree (RRT)	-RRT -RRT* -improved RRT* with Bezier curves	-mobile robot -industrial robot manipulator	-RRT faster than PRM -RRT* can generate smooth path	-Not optimal solution. -Not suitable for dynamic environment
[27] [28] [29] [30] [31] [32] [33]	Artificial Potential Field (APF)	-APF -improved APF -Hybrid between APF and minimizing robot-base coupling.	-mobile robots -UAV's -industrial robots -space robots	-fast response -low computation time -smooth path -best at avoiding obstacles	-hard to implement in real environment -can get stuck in local minimum solution,
[34]	Greedy algorithms	-Dijkstra algorithm A*, D*, Theta*, Depth-first, and Breadth-first.	- Warehouse robot	-simple to implement -low computation -D* can work in a dynamic environment with fast replanning.	-large memory -cannot work in dynamic environment -D*'s fast replanning will have high computational cost in large space.
[35] [36]	Genetic Algorithms	-GA algorithm -Multi-objective GA	-mobile robot - redundant serial robot	-capable of global search. -capable of multiple objectives search.	-high computation time -high modelling complexity.

[37] [38] [39]	Particle Swarm Optimization	-PSO algorithm -multi objective PSO	-mobile robot -UAV's -industrial robot. -space robot	-capable of global search. -capable of multiple objectives search.	-can be trapped in local minimum. -slower convergence towards the end.
[40] [41] [42]	Reinforcement Learning	- Model Free RL -Model Based RL		-can be trained in parallel. -high efficiency to avoid obstacles -can handle dynamic environments -does not require training data set	-slow convergence in training phase. -requires agent to directly affect environment; this will cause high wear & tear of the robot, and high risk of damage.

2.2 Space Manipulator Path Planning and Control

Path planning for space robotic manipulators presents unique challenges distinct from those encountered with fixed-base manipulators. Typically, a space robotic manipulator system includes a base spacecraft with one or multiple robotic manipulators, see Figure 2.8.

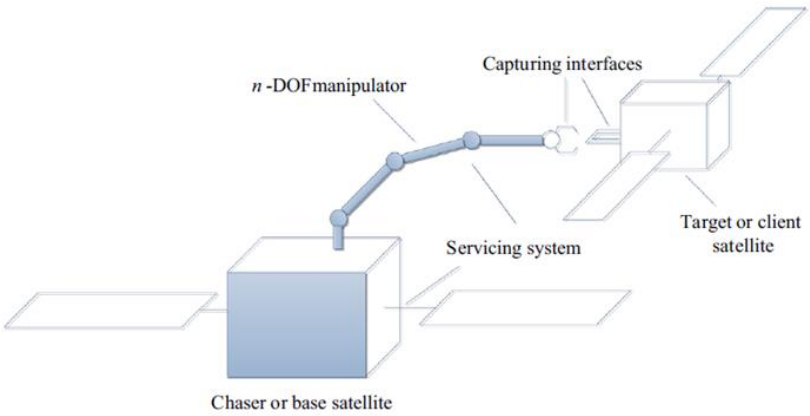


Figure 2.8 Space Robotic Manipulator system [5]

Space robotic manipulator systems are divided into two categories based on how they operate: free-flying and free-floating [43]. In a free-flying manipulator, the pose, or position and orientation, of the base spacecraft is actively controlled by thrusters while the manipulator performs its tasks, making it similar to a fixed-base manipulator. The path planning and control of the manipulator in such conditions have been well-studied [44]. However, this comes at the cost of fuel consumption, which reduces the orbital life of the system. In contrast, free-floating manipulator systems have a base spacecraft whose pose is not controlled, except the manipulator's joints are actively controlled. The free-floating configuration is advantageous as it can save spacecraft fuel (that would be consumed for base attitude control) and avoid possible collision with the target due to the excess excessive attitude control on the base spacecraft. This configuration leads to dynamic coupling between the manipulator and the base spacecraft due to the conservation of momentum, which presents unique challenges for path planning and control, as manipulating the end-effector can disturb the base spacecraft's pose and vice versa. As a result, the end-effector's pose depends on not only current joint angles but also the previous velocities of the joints and base spacecraft [45]. The nonholonomic characteristic of the angular momentum means that path-planning solutions typically used for fixed-base or free-flying space manipulators cannot be directly applied to free-floating space manipulators

[46].

When using a free-floating space manipulator, the system may experience Dynamic Singularities [47]. The existence of dynamic singularities is shown first by writing the kinematics and conservation equations (section 3.1). The end-effector's linear and angular velocities can be expressed solely as a function of the velocities of the manipulator controlled joint angles, and that they do not depend upon the uncontrolled linear and angular velocity of the base spacecraft, resulting in the Generalized Jacobian Matrix (GJM).

The rank of this Jacobian matrix is deficient at given points in the manipulator's joint space which results in the manipulator being unable to move its end-effector in some direction in inertial space. These singular points cannot be determined solely from the kinematic structure of the system and instead depend upon a system's masses and inertias; hence they are called dynamic singularities.

Since they are not fixed in a manipulator's inertial workspace, dynamics singularities are path dependant. This is because the end-effector location in inertial space depends upon the history of the spacecraft attitude which is determined by the path taken by the end-effector.

This path dependence is due to the non-integrability of the angular momentum of the system. A result of this property is that, if the end-effector returns to its initial inertial location and orientation after having followed a

closed path in inertial space, the spacecraft's orientation and the manipulator's configuration will be different than the initial ones. Up to date there are only two methods to handle Dynamics singularities:

1 Transform the complex dynamic singularity problem into a real-time kinematic singularity avoiding problem [48]. For this method, the robotic EE velocities are expressed relative to the base frame (similar to fixed base) and the base attitude and angular velocity must be measured by corresponding sensors in real-time.

   The Jacobian matrix is used and is independent on the mass parameters, meaning the singularities are kinematic singularities. Although the dynamic singularity points cannot be identified beforehand, when the angular velocity and the attitude of the base are measured in real-time (at each sampling period), the simplified kinematics are derived. transforming the complex dynamic singularity problem into a real-time kinematic singularity avoiding problem.

2 Plan the path of the free-floating space manipulator only within its Path Independent Workspace (PIW). In [49], it is shown that the workspace for a free-flying space manipulator (where the attitude of the base is controlled) is a sphere with its center at the CM of the system. Furthermore, for a free-floating space manipulator (where the attitude of the base is not controlled) points in this space can be reached but with suitable path selection.

This concept was tested by Papadopoulos [50]. Using a planar 2DOF free-floating space manipulator as shown in Figure 2.9.

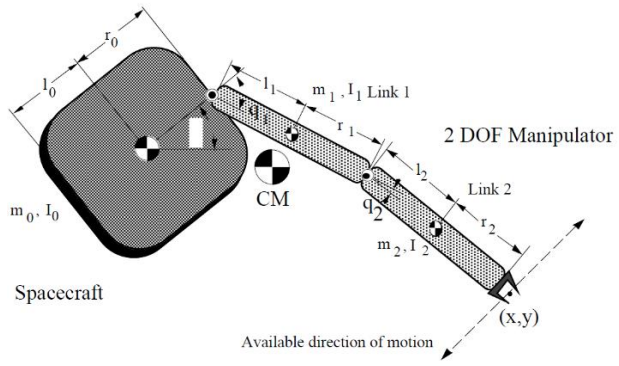


Figure 2.9 A planar 2-DOF free-floating robotic system [50]

He derived the kinematics equations for the manipulator, constructed the Generalized Jacobian Matrix, and built the workspace as seen in Figure 2.10.

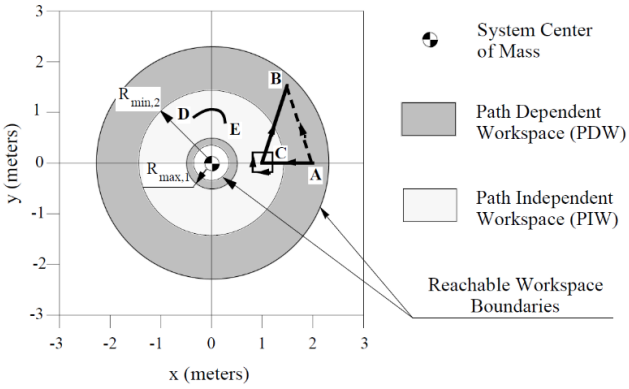


Figure 2.10 The reachable, Path dependant, and Path independent Workspace [50]

Within the reachable workspace of the manipulator there are two main regions; the Path Independent Workspace (PIW), where any point can be

reached by any path, and it will never lead to dynamic singularities. An example of the PIW was moving the manipulator EE from point E to D in the Figure 2.10. It moved without any issues. The Path Dependent Workspace (PDW), where any point can be reached but may or may not encounter dynamic singularities. Meaning that point can be reached with some paths but not with others. An example of the PDW was moving the manipulator EE from point A to B; the direct path encountered a dynamic singularity and failed. However, moving from point A to C first then moving from point C to B worked, and the manipulator did not encounter a dynamic singularity.

To plan the path of the free-floating space manipulator only within its PIW is not practical since the PIW is a fraction of the full workspace that can be utilized. As proven by Papadopoulos, with trial and error we can reach any point in the workspace (even within the PDW) by using a different path. Leading research towards Optimization techniques and more recently Reinforcement Learning.

Various techniques have been proposed to address the challenge of the path-planning problem of free-floating space manipulators, which impose difficulties not encountered for fixed-base manipulator control [51–53]. One direct solution to path planning is based on the inverse kinematics of a space robotic manipulator while satisfying the nonholonomic angular momentum

constraints [51]. However, the optimality of the path cannot be guaranteed due to unknown dynamic singularities, which can seriously degrade the performance of free-floating manipulators [53].

Optimization theory offers better path-planning solutions, in [38] the Trajectory planning of a 7 DOF redundant free-floating space robot was done using Particle Swarm Optimization (PSO). Treating the path planning problem as an optimization problem and taking into consideration the existence of the kinematic and non-holonomic redundancies. Three simulation cases were done; The first case was to find an optimized path for the free-floating robot's EE that can reach the target location (with an admitted position and orientation error). In the second case a multi objective optimization problem was solved, with two objectives; the EE was required to reach the desired pose, and at the same time minimize the base orientation disturbance. In the third case another objective was introduced; along with the first two objectives above, the third objective was to maximize the robot's finial manipulability. In [53] a non-singular terminal sliding mode controller was used to find a path planning solution for a free-floating manipulator with improving the transient performance and reduce unexpected base motion in the presence of disturbance.

Although optimization techniques succeed in finding a feasible solution for free-floating manipulators, these methods suffer from heavy computational

loads of optimization, making real-time implementation difficult to handle uncertain disturbances. Furthermore, if the initial conditions of the system change, the solution must be reoptimized and solved again.

To overcome these challenges, recent advancements have pivoted to machine learning [54], particularly reinforcement learning (RL) [55]. RL enables an agent to learn a decision-making policy for complex tasks in dynamic and uncertain environments, such as the robotic manipulator in this context, to maximize a pre-defined reward function [56,57]. In such a process, the agent sends commands to change the manipulator's state and receives feedback in the form of rewards based on the outcomes of that action in a Markov Decision Process (MDP) without any prior samples and rules. The training under RL is conducted in joint space, which bypasses the kinematic and dynamic singularities prevalent in the Cartesian space of free-floating robotic manipulator. If a singularity occurs and the target is not reached in the training, the RL algorithm simply discards the episode due to lack of reward, which is contingent on successful path planning.

Numerous Reinforcement Learning algorithms, such as Q-learning, Deep Deterministic Policy Gradient (DDPG), and Actor-Critic algorithms, have demonstrated success in addressing path-planning challenges for free-floating robotic manipulators in continuous action space. For instance, the soft Q-learning algorithm has been effectively used with a 3-DOF space robotic

manipulator to guide the end-effector towards a target while minimizing joint torques [58]. Although Q-learning is suited to environments with discrete action sets, policy-gradient-based methods like DDPG are better for continuous state and action spaces [59]. DDPG has been widely applied to various manipulators. It was used for a 2-DOF space robotic manipulator in [60] with a reward function that accounted for obstacle avoidance, coupled base-manipulator dynamic disturbances, and path length constraints. Another application [61] involved the path planning for a 3-DOF space robotic manipulator, where DDPG facilitated faster approach by incorporating pre-trainings to improve learning efficiency. Similarly, for a dual 3-DOF arm space robotic manipulator [62], DDPG was used with velocity constraints to avoid arm self-collision in its reward function. The Actor-Critic algorithm was also applied to a 6-DOF space robotic manipulator for target capture tasks [63]. Moreover, the DDPG algorithm was used for collision-free path planning of a 7-DOF fixed-base manipulator [64] and free-floating space manipulator [65] by representing obstacles with artificial potential fields in the reward functions. In [66] the DDPG was also used for motion planning of a 3-DOF space manipulator in order to learn to achieve capture and avoid collisions between the robotic links with themselves and target.

However, these studies often overlook factors such as observation noise and the suppression of robotic link oscillations. Additionally, they typically

presume relatively small manipulator-base spacecraft mass ratios. This assumption suggests a similarity to fixed-base robotic manipulators, which may not accurately represent the operational dynamics of certain free-floating robotic manipulator.

Table 2.2 summarizes the application of RL in the path planning of space robotic manipulators and associated reward functions in the literature. It shows the predominant application of the DDPG algorithm across successful implementations. Most studies investigated 3-DOF robotic manipulators, with relatively fewer works for 6-DOF or beyond. A common oversight in these studies is the disregard for joint velocity constraints. Notably absent is consideration for quaternion-based attitude control and the impact of state measurement noise that is critical to real-world applications. Furthermore, the utilization of scalar reward functions in the MDP may inadvertently restrict the search space or fall into local optima, potentially due to overfitting. The challenge of striking a balance between exploration and exploitation in RL continues to be a focal point of ongoing research [67].

Table 2.2 Literature employing RL for Space robotic Control

Ref	Algorithm	DOF	Reward Function Expressions	Constraints Included		
				EE Position	EE Attitude	Joint Velocity
[60]	MRDDPG	2	Depends on constraints (safety,	Yes	No	No

			disturbance, path)			
[58]	Soft Q-Learning	3	$R=-(t/T)\begin{bmatrix}K_1d^2+K_2\log\left(d^2+\delta\right)\\+K_3\left\|\tau\right\|^2\end{bmatrix}$	Yes	No	No
[61]	DDPG + pre-training	3	$R=-0.3^{0.99t}\begin{bmatrix}K_1d^2+K_2\log\left(d^2+\delta\right)\\+K_3\left\|\tau\right\|^2\end{bmatrix}$	Yes	No	No
[62]	DDPG	3×2	$R=-\left(d_1+d_2\right)^2-\log\left(d_1+10^{-8}\right)$ $-\log\left(d_2+10^{-8}\right)-\left\|v_{1,e}\right\|/(d_1^2+1)$ $-\left\|v_{2,e}\right\|/(d_2^2+1)-1/(100d_{\min}^2+0.5)$	Yes	No	Yes
[63]	Deep RL	6	$R_1=(2-d)^2+(1-d/2)(\sqrt{3}\pi-\vartheta)-33.6$ $R_2=\begin{cases}2d-0.1\vartheta-1.9 & d\geq 0.1\\1+e^{\vartheta}-10d^2 & d<0.1\end{cases}$ $R_3=\begin{cases}-2d-0.1\vartheta-1.9 & d\geq 0.1\\-\cos(\vartheta)-10d^2 & d<0.1\end{cases}$ $d=\sqrt{x^2+y^2+z^2},\quad \vartheta=\sqrt{\alpha^2+\beta^2+\gamma^2}$	Yes	Yes (Euler angle)	No
[65]	DDPG	7	$U(d,\vartheta)=-K_1d+K_2/[(d+1)(\vartheta+1)]$ $R=U_{t+1}(d,\vartheta)-U_t(d,\vartheta)$	Yes	Yes (Euler angle)	No
[66]	DDPG	3	$R=-K_1d-p_C-K_2\sum\left(\dot{\varphi}\right)^2$	Yes	No	Yes

R is the reward gained each timestep t, t/T is a scaler that amplifies the reward in the final phase of the capture approach, d is the distance between the robot's EE (end-effector) and target, τ is the sum of all joint torques, (K_1,

K_2, K_3) are the weights, v_e is the EE's linear velocity, ϑ is the orientation difference between the EE and the target in Euler angles (α, β, γ), $U(d, \vartheta)$ and $U_{t+1}(d, \vartheta)$ are the potential energy at the time t and t+1, p_C is a collision penalty, and $\sum(\dot{\varphi})^2$ is the sum of joint velocities.

Table 2.3 also summarizes the application of RL in the path planning of space robotic manipulators in the literature, with special focus on the different constraints used during training. In order to create a robust agent that can autonomously control a space robotic manipulator under any condition, the training must be done in a dynamic and changing environment, the literature lacks this aspect.

Table 2.3 Gaps in Literature employing RL for Space robotic Control

Ref		[60]	[58]	[61]	[62]	[63]	[65]	[66]
Algorithm		MRDDPG	Soft Q-Learning	DDPG + pre-training	DDPG	Deep RL	DDPG	DDPG
DOF		2	3	3	4	6	7	3
Target Capture Constraints	EE* Position	✓	✓	✓	✓	✓	✓	✓
	EE* Attitude	-	-	-	-	✓ (Euler angles)	✓ (Euler angles)	-
	Joint Velocity	-	-	-	✓	-	-	✓
Obstacle Avoidance		-	-	-	-	-	-	-
Observation Noise		-	-	-	-	-	-	-
Random IC's* during training	Joint angles	-	-	-	✓	-	✓	-
	Target position	-	-	-	-	-	-	-

	Obstacle position	-	-	-	-	-	-	-

* EE – End Effector. * IC – Initial Conditions.

This Research introduces an improved reward function strategy in RL, implemented through the DDPG algorithm with Actor-Critic networks, for path planning of a 6-DOF free-floating robotic manipulator. This approach emphasizes the formulation of reward functions that facilitate rapid approach in environments subject to disturbances. Key considerations include:

-Aligning the manipulator's end effector (EE) with the target using a quaternion-based approach for orientation comparison, while mitigation of dynamic coupling between the manipulator and the base spacecraft.

-Incorporating joint velocity constraints to achieve smoother EE trajectories.

-Measures to prevent self-collision of the manipulator's links and, importantly, incorporates external obstacle avoidance capabilities during target approach.

-A safety shield, cone penalty constraint triggered when the manipulator's EE is in close-proximity of the target - a feature not previously implemented in the literature.

Recognizing the imperfection of real-world sensors, we further enhance our model's practical applicability by including white noise in the observations during training. This ensures that the trained agent is resilient to signal noise and capable of achieving successful target capture. The effectiveness and

advantages of our proposed algorithm are validated through numerical simulations.

Lastly, training will occur in a dynamic environment. The reinforcement learning agent will train with different initial conditions that are randomly appointed at the start of each episode, this includes the manipulator initial joint angles, target, and obstacle positions. The trained agent is expected to find a suitable path for any target location, avoiding an obstacle placed anywhere in the environment, wherever the manipulator initial position exists. The algorithm's efficacy and its advantages are demonstrated through numerical simulations.

2.3 Space Manipulator Experimental Validation

It is critical when controlling the space robotic manipulator to achieve safe capture /soft capture, i.e., capturing the free-floating malfunctioned spacecraft or debris (target) without pushing them away while destabilizing the attitude of the base spacecraft that carries the robot. After the safe capture and post-stabilization of the spacecraft-target combination, the on-orbit service can be performed safely [2]. The operation involves physical contact with the target by the robot and is extremely risky. Thus, the capture system and control laws must be thoroughly tested and verified on Earth before being launched into space [68].

Here is where the research gap exists between all the theoretical path planning methods done so far using simulations, and the inability to test these methods experimentally in space.

Ground-based testing and verification of a space robot's dynamic responses and associated control laws in contact with an unknown 3D target in microgravity are exceedingly complicated. Different methods exist to try to mimic the microgravity environment here on earth.

The most common method being an air-bearing testbed [69], Figure 2.11 shows our lab air-bearing table at York university [70,71]. In Figure 2.12. also shows the Polish academy of sciences planar testbed [69]. Although air-bearing can almost completely eliminate friction and can achieve a zero-gravity experiment, it cannot emulate the 6DOF motion of a space robot and is limited to 3DOF and planar base motions.

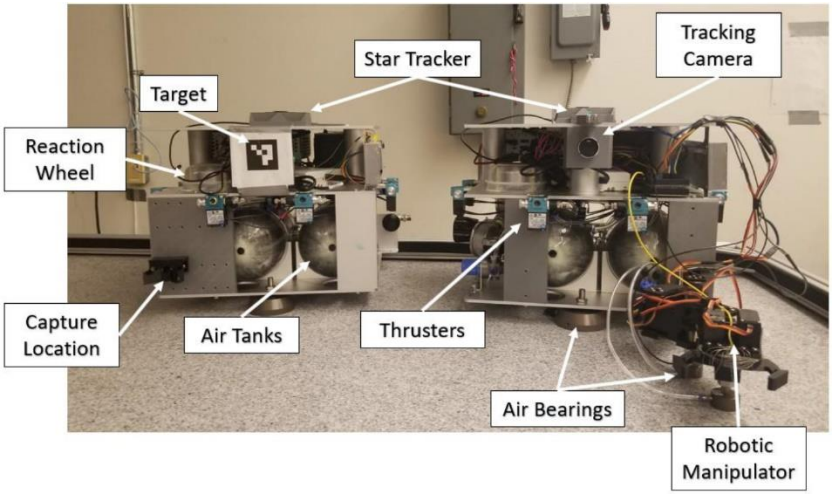


Figure 2.11 Planar Air bearing table at York University [70]

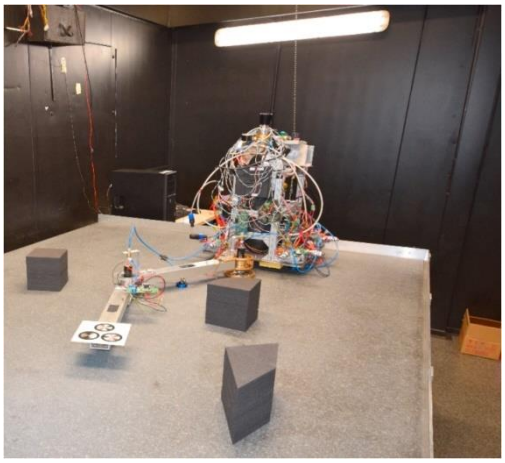


Figure 2.12 Planar air-bearing microgravity simulator at the Space Research Centre of the Polish Academy of Sciences [69]

Another method is utilizing active suspension for gravity compensation [72], seen in Figure 2.13. This method can achieve the 6DOF motion, but the system may become unstable due to the coupled vibration of the space manipulator and suspension system, also it is very difficult to identify the kinetic friction and compensate it in the tension control system precisely.

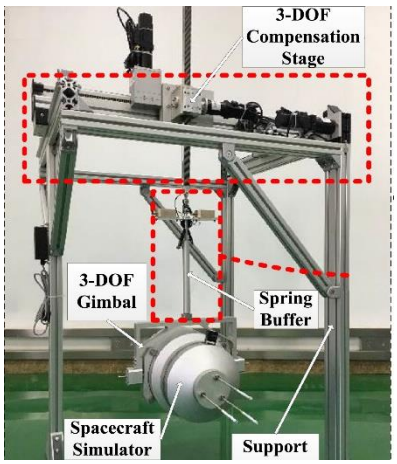


Figure 2.13 Suspention system for gravity compensation [72]

In [73] neutral buoyancy was used to create a micro-gravity experiment in a pool or water shown in Figure 2.14, and it was able to test 6DOF motions of a space robotic arm. However, these robotic arms must be custom built; to avoid damage from the water submersion. Furthermore, fluid damping and inertia can affect the results when the system's dynamic behavior is important.

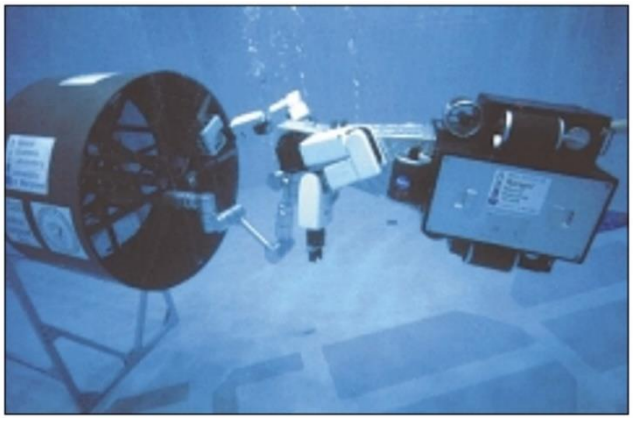


Figure 2.14 Ranger NBV grappled to satellite mockup at the University of Maryland Neutral Buoyancy Research Facility [73]

The method that is closest to achieve a zero-gravity environment on earth is the parabolic flight, airplane free-falling. Except for the parabolic flight, none of these methods can truly emulate the 6DOF motion in microgravity. For this reason, my previous colleagues in our lab have tested our own satellite tether deployment system before [74] shown in Figure 2.15.

Figure 2.15 Suspention system for gravity compensation [74]

Even so, parabolic flight cannot be used for testing space robotic motions, since the microgravity period is short (20-30 seconds), which is not sufficient to test the entire process of robotic capture of a target, the cost is high, and the testing space in the airplane cabin is limited. The 6DOF hardware-in-the-loop (HIL) ground testbed has the ability to overcome these difficulties. The HIL system consists of a hybrid combination of mathematical and mechanical models; the mathematical model calculates the free-floating space manipulator motions in micro-gravity using simulation, then the mechanical model compels a physical robot to move according to the simulated motion in the real world. This method is relatively cheaper and can mimic a space capture mission with ease [75]. Figure 2.16 Shows the HIL testbed at the Shenzhen Space Technology Center in China [76]. Figure 2.17 also shows the European proximity operations simulator at the German Aerospace Center [77]. The CSA developed a complex HIL simulation system called STVF (SPDM Task Verification Facility), shown in Figure 2.18, to simulate the dynamic behavior

of the space robot Special Purpose Dexterous Manipulator performing maintenance on the ISS [78]. It is considered the formal verification tool for the SPDM. Figure 2.19 shows the manipulator task verification facility (MTVF) developed by China Academy of Space Technology [79,80]. And Figure 2.20 shows the dual robotic testbed at the Research Institution of Intelligent Control and Testing, in Tsinghua University [81].

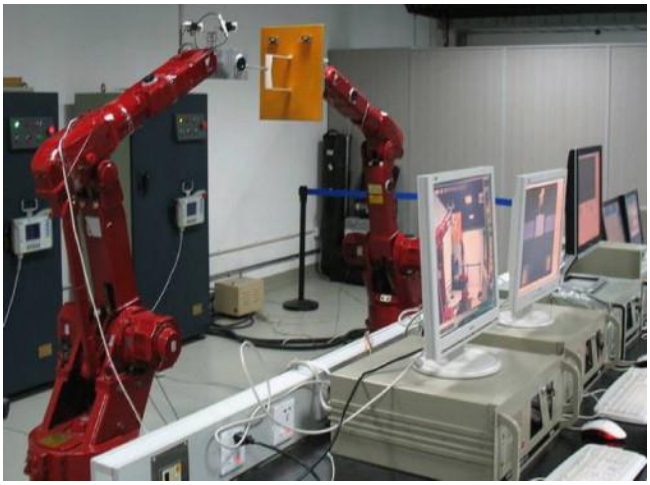


Figure 2.16 Shenzhen Space Technology Center HIL testbed[76]

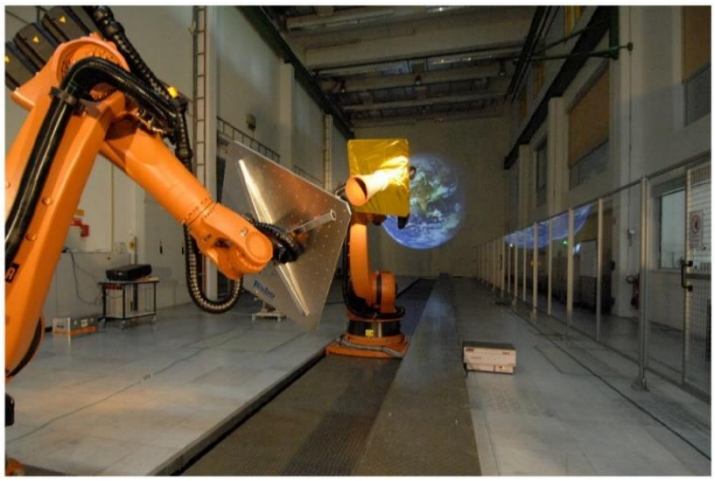


Figure 2.17 European proximity operations simulator at the German

Aerospace Center [77]

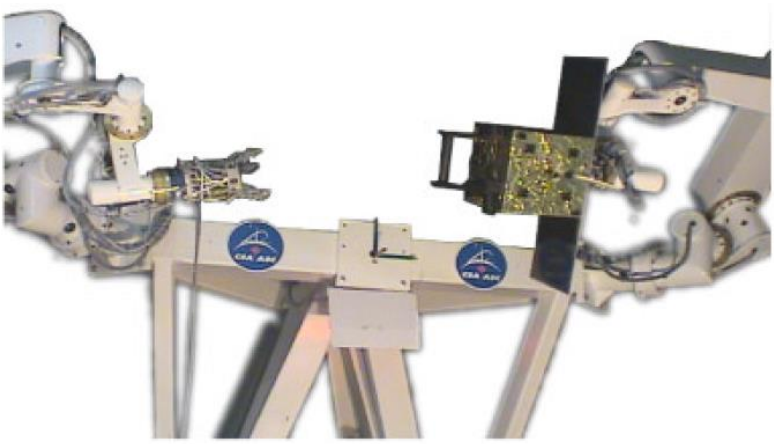


Figure 2.18 CSA SPDM Task Verification Facility [78]

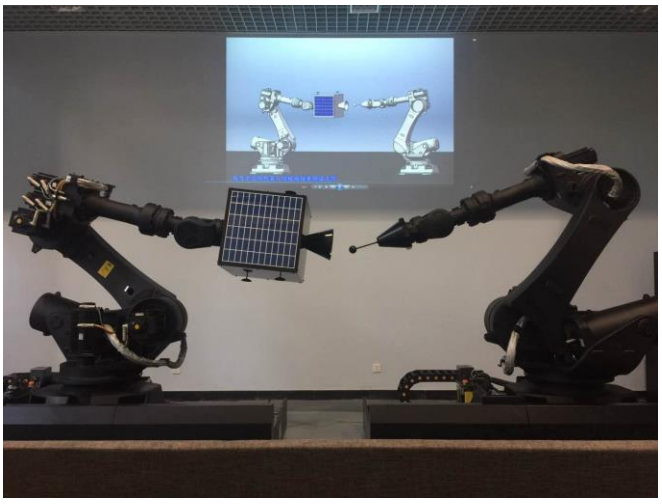


Figure 2.19 Manipulator task verification facility (MTVF) developed by China Academy of Space Technology [79]

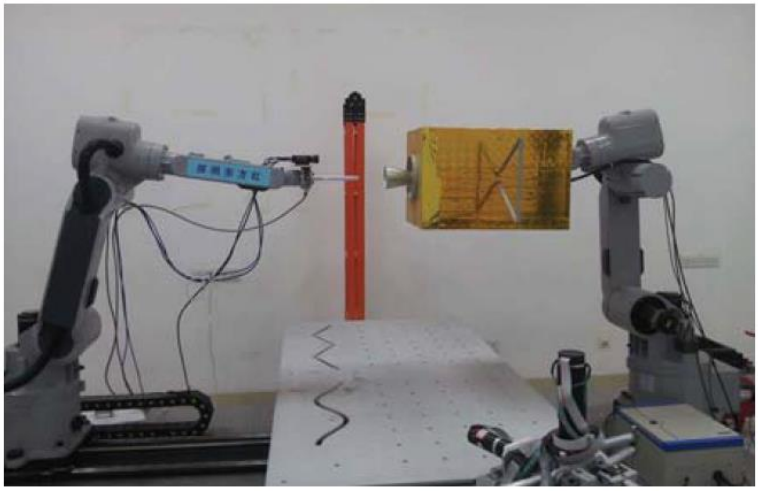


Figure 2.20 Research Institution of Intelligent Control and Testing, Tsinghua University [81]

This research details our efforts to design and construct a hardware-in-the-loop ground testbed with active gravity compensation at York University. The testbed can perform the 6 degrees-of-freedom (DOF) motion maneuvers that free-floating robotic manipulators or tumbling satellites/targets achieve in microgravity conditions. It incorporates two industrial-sized 6DOF robotic arms, along with a number of sensing equipment such as cameras and force/torque sensors. This work also includes the initial experimental results in motion control, computer vision, and sensing equipment capabilities. The facility will serve as an invaluable tool for the development and validation of space robotic manipulators capture maneuvers, ultimately enhancing their effectiveness and reliability in space missions.

Chapter 3 MODELING OF SPACE ROBOTIC MANIPULATOR SYSTEM

Summary: This chapter presents the dynamic model of a general free-floating space robotic manipulator, which serves as the simulated environment in the reinforcement learning training. Furthermore, the physical model for the 6DOF HIL ground testbed, the components it consists of, and the equations that govern it are detailed.

3.1 Free-floating Space Robotic Manipulator Model

A spacecraft manipulator system is here defined to be free-floating when the 6DOF pose of the base-spacecraft is not actively controlled in orbit, where only the manipulator joints are actively controlled. The system moves only under the effect of the internal reactions due to the actuation of the manipulator's joint motors. As robotic manipulators mounted on spacecraft typically only use revolute joints, these internal reactions are typically torquing.

Consider a generalized free-floating space robotic manipulator in free space, as shown Figure 3.1. The model consists of a base spacecraft and a manipulator with n rigid links connected by n independently actuated revolute joints in series. An inertial frame, OXYZ, is defined in free space, where the

base spacecraft (B0) is treated as the link 0 with 6DOF.

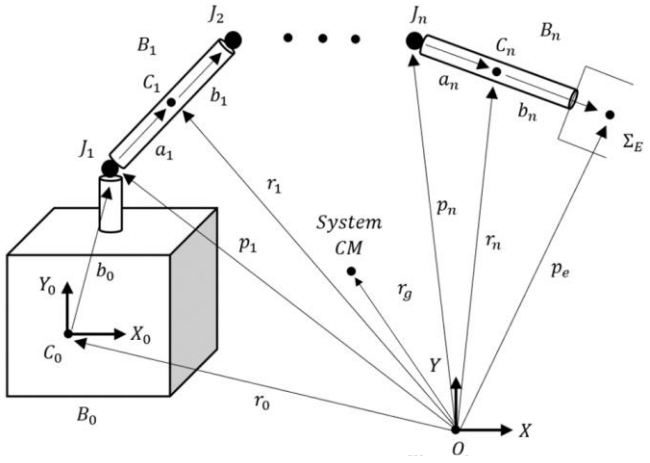


Figure 3.1 Schematic of a generalized free-floating space robotic manipulator with *n* rigid links

The center of mass of the base spacecraft is treated as joint 0 and denoted by C_0, its distance from the origin of the inertial frame is $\boldsymbol{r}_0 \in \mathbb{R}^3$, and its rotation matrix relative to the inertial frame is denoted by $\boldsymbol{A}_0 \in \mathbb{R}^{3\times3}$.

The manipulator's 1st link is connected to the base spacecraft at joint J_1, which is denoted as $\mathbf{b}_0 \in \mathbb{R}^3$ from C_0 with a rotation matrix $\mathbf{A}_1 \in \mathbb{R}^{3\times3}$. Each link, labeled B_i, is connected by a joint J_i $(1 \le i \le n)$. The position and rotation of J_i is represented by a position vector $\mathbf{p}_i \in \mathbb{R}^3$ and a unit vector $\mathbf{j}_i \in \mathbb{R}^3$, respectively. The end-effector is treated as the $(i+1)^{\text{th}}$ link. Its position and rotation are denoted by $\mathbf{p}_e = \mathbf{p}_{i+1}$ and $\mathbf{A}_e$, respectively.

In this configuration, the position ($\mathbf{p}_e$) and orientation ($\mathbf{A}_e$) of the end-effector can be determined by propagating the transformations from the base spacecraft to the end-effector, such that:

$$\mathbf{p}_e = \mathbf{r}_0 + \mathbf{b}_0 + \sum_{i=1}^{n} (\mathbf{p}_{i+1} - \mathbf{p}_i) \tag{4.1}$$

$$\mathbf{A}_e = \mathbf{A}_o \prod_{i=1}^{n} \mathbf{A}_i \tag{4.2}$$

where $\mathbf{A}_i$ is the rotation matrix of the i^{th} link.

Differentiating Equations (4.1)-(4.2) with respect to time yields the linear ($\mathbf{v}_e \in \mathbb{R}^3$) and angular ($\boldsymbol{\omega}_e \in \mathbb{R}^3$) velocities of the end-effector.

$$\mathbf{v}_e = \mathbf{v}_0 + \boldsymbol{\omega}_0 \times (\mathbf{p}_e - \mathbf{r}_0) + \sum_{i=1}^{n} \left[(\dot{\varphi}_i \mathbf{j}_i) \times (\mathbf{p}_e - \mathbf{p}_i) \right] \tag{4.3}$$

$$\boldsymbol{\omega}_e = \boldsymbol{\omega}_0 + \sum_{i=1}^{n} \boldsymbol{\omega}_i \,,\ \boldsymbol{\omega}_i = \dot{\varphi}_i \mathbf{j}_i \tag{4.4}$$

where $\mathbf{v}_0 \in \mathbb{R}^3$ and $\boldsymbol{\omega}_0 \in \mathbb{R}^3$ are the linear and angular velocities of the spacecraft, respectively. And $\boldsymbol{\omega}_i = \dot{\varphi}_i \mathbf{j}_i$ with $\dot{\varphi}_i \in \mathbb{R}$ and $\mathbf{j}_i \in \mathbb{R}^3$ being the i^{th} joint's angular velocity and direction in the inertial frame, respectively.

Define the states of the base spacecraft and the end-effector as $\dot{\mathbf{x}}_b = [\mathbf{v}_0, \boldsymbol{\omega}_0]^T$ and $\dot{\mathbf{x}}_e = [\mathbf{v}_e, \boldsymbol{\omega}_e]^T$, respectively. Equations (4.3)-(4.4) can be rewritten in compact matrix form.

$$\dot{\mathbf{x}}_e = \mathbf{J}_b \dot{\mathbf{x}}_b + \mathbf{J}_m \dot{\mathbf{\Phi}} \tag{4.5}$$

where $\dot{\boldsymbol{\Phi}} = [\dot{\phi}_1, \dot{\phi}_2, \cdots \dot{\phi}_n]^T$ is the vector of joint angular velocity, and ($\boldsymbol{J}_b \in \mathbb{R}^{6\times 6}$ and $\boldsymbol{J}_m \in \mathbb{R}^{6\times n}$) are the Jacobian matrices dependent on the base spacecraft and manipulator motions, respectively, such that:

$$\mathbf{J}_b = \begin{bmatrix} \mathbf{I} & \langle \mathbf{r}_0 - \mathbf{p}_e \rangle \\ 0 & \mathbf{I} \end{bmatrix} \in \mathbb{R}^{6\times 6} \tag{4.6}$$

$$\mathbf{J}_m = \begin{bmatrix} \langle \mathbf{p}_1 - \mathbf{p}_e \rangle \mathbf{j}_1 & \langle \mathbf{p}_2 - \mathbf{p}_e \rangle \mathbf{j}_2 & \cdots & \langle \mathbf{p}_n - \mathbf{p}_e \rangle \mathbf{j}_n \\ \mathbf{j}_1 & \mathbf{j}_2 & \cdots & \mathbf{j}_n \end{bmatrix} \in \mathbb{R}^{6\times n} \tag{4.7}$$

Here, $\langle\ \rangle$ represents a 3×3 skew-symmetric matrix of corresponding vectors of the cross product. The detailed expressions of $\mathbf{J}_b$ and $\mathbf{J}_m$ are given in [82]. Equation (4.5) indicates that the motion of the end-effector can be determined with the feedback of the states of the base and joints.

The assumption of a free-floating space robotic manipulator implies that the linear and angular momentums of the system are conserved in free space because no thrust is applied to the base spacecraft. For zero initial linear and angular momentums, the momentum conservation can be expressed as:

$$\mathbf{P} = m_0 \mathbf{v}_0 + \sum_{i=1}^{n} m_i \left(a_i \dot{\mathbf{p}}_i + b_i \dot{\mathbf{p}}_{i+1} \right) = 0 \tag{4.8}$$

$$\mathbf{L} = \left(\mathbf{I}_0 \boldsymbol{\omega}_0 + \mathbf{r}_0 \times m_0 \mathbf{v}_0 \right) + \sum_{i=1}^{n} \left[\mathbf{I}_i \boldsymbol{\omega}_i + \mathbf{r}_i \times m_i \left(a_i \dot{\mathbf{p}}_i + b_i \dot{\mathbf{p}}_{i+1} \right) \right] = 0 \tag{4.9}$$

Or in compact matrix form:

$$\begin{bmatrix}\mathbf{P}\\ \mathbf{L}\end{bmatrix} = \mathbf{M_b}\dot{\mathbf{x}}_\mathbf{b} + \mathbf{M_m}\dot{\mathbf{\Phi}} = 0 \tag{4.10}$$

where m_i $(i = 0,1,\cdots,n)$ is the mass of the i^{th} link, $\boldsymbol{I}_i \in \mathbb{R}^{3\times3}$ is the inertia moments of the i^{th} link at its center of mass, (a_i, b_i) are the distances from the CoM to the two ends, , $\mathbf{M_b} \in \mathbb{R}^{6\times6}$ is the inertia matrix of the base spacecraft, and $\mathbf{M_m} \in \mathbb{R}^{n\times n}$ is the coupling inertia matrix of the manipulator. The detailed expression of these inertia matrices can be found in [82].

Combining Equations (4.5) and (4.10) yields

$$\dot{\mathbf{x}}_e = \left(\mathbf{J}_m - \mathbf{J}_b\mathbf{M}_b^{-1}\mathbf{M_m}\right)\dot{\mathbf{\Phi}} = \mathbf{J}_g\dot{\mathbf{\Phi}} \tag{4.11}$$

where $\mathbf{J}_g \in \mathbb{R}^{6\times n}$ is the Generalized Jacobian Matrix (GJM) dependent on the base spacecraft attitude, joint angles, and mass properties of the base spacecraft and the links.

Accordingly, the dynamics of the free-floating robotic manipulator can be derived [82]:

$$\mathbf{M}\ddot{\mathbf{q}} + \dot{\mathbf{M}}\dot{\mathbf{q}} + \mathbf{C}\left(\dot{\mathbf{q}}, \mathbf{q}\right) = \mathbf{F} \tag{4.12}$$

where,

$$\mathbf{M} = \begin{bmatrix}\mathbf{M}_b & \mathbf{M}_{bm}\\ \mathbf{M}_{bm}^T & \mathbf{M}_m\end{bmatrix} \tag{4.13}$$

$$\mathbf{F} = \begin{bmatrix}\mathbf{0}\\ \mathbf{u}\end{bmatrix} \tag{4.14}$$

$$\mathbf{C}(\dot{\mathbf{q}},\mathbf{q}) = \begin{bmatrix} \mathbf{c}_b \\ \mathbf{c}_m \end{bmatrix} \tag{4.15}$$

$$\mathbf{c}_b = -\frac{1}{2}\frac{\partial}{\partial \mathbf{x_b}}\left(\dot{\mathbf{x}}_b^T \mathbf{M}_b \dot{\mathbf{x}}_b + \dot{\boldsymbol{\Phi}}^T \mathbf{M}_m \dot{\boldsymbol{\Phi}} + \dot{\mathbf{x}}_b^T \mathbf{M}_{bm} \dot{\boldsymbol{\Phi}} + \dot{\boldsymbol{\Phi}}^T \mathbf{M}_{bm} \dot{\mathbf{x}}_b\right) \tag{4.16}$$

$$\mathbf{c}_m = -\frac{1}{2}\frac{\partial}{\partial \boldsymbol{\Phi}}\left(\dot{\mathbf{x}}_b^T \mathbf{M}_b \dot{\mathbf{x}}_b + \dot{\boldsymbol{\Phi}}^T \mathbf{M}_m \dot{\boldsymbol{\Phi}} + \dot{\mathbf{x}}_b^T \mathbf{M}_{bm} \dot{\boldsymbol{\Phi}} + \dot{\boldsymbol{\Phi}}^T \mathbf{M}_{bm} \dot{\mathbf{x}}_b\right) \tag{4.17}$$

and $\mathbf{q} = \left[\mathbf{x_b}, \boldsymbol{\Phi}\right]^T \in \mathbb{S}^{6+n}$ is the state vector of the space robotic manipulator, $\mathbf{M_{bm}} \in \mathbb{R}^{6\times n}$ is the inertial matrix reflecting the dynamic coupling between the manipulator and the base spacecraft, $\mathbf{c}_b \in \mathbb{R}^6$ and $\mathbf{c}_m \in \mathbb{R}^n$ are the Coriolis effects acting on the base spacecraft and the manipulator, and $\mathbf{u} \in \mathbb{R}^n$ is the control torque vector at all joints.

Equation (4.12) forms the mathematical foundation for the simulation of the free-floating space robotic manipulator in free space, which serves as the environment in the training of reinforcement learning-based control policy.

In order to build our simulation environments with outmost credibility and ascertain the algorithm's performance and effectiveness under real-world conditions, the parameters used in the simulations are based on actual space robotic manipulator missions [15]. Table 1.1 shows a summary of past space robotic manipulator missions. Notably the mass ratios in most cases are small except for the Orbital Express mission, with a 7.4% mass ratio being the highest.

Furthermore, the simulation studies that have been done on space robotic manipulator systems have been reviewed and summarized these studies in Table 3.1.

Table 3.1 List of Space Robotic Simulation Studies

Study	Robotic Arm DOF	Robotic Arm Length	Robotic Arm Mass	Spacecraft Base Mass	Arm-Base Mass Ratio
[83]	6	6.25 m	125 kg	1700 kg	7.353%
[84]	6	6.2 m	63 kg	800 kg	7.875%
[85]	6	2.165 m	25 kg	400kg	6.25%
[82]	6	6.5 m	125 kg	1700 kg	7.35%
[45]	3	4 m	200 kg	2000 kg	10%
[86]	7	3.496 m	29 kg	200 kg	14.5%
[87]	2	3 m	11kg	60 kg	18.3%
[88]	7	3.21 m	66.7 kg	375 kg	18%
[33]	2	4 m	60 kg	300 kg	20 %

3.2 Hardware-in-the-Loop Ground Testbed Model

The HIL system consists of a hybrid combination of mathematical and

mechanical models. The mathematical model calculates the motions of the free-floating satellite and spacecraft-manipulator system in microgravity. The output from the mathematical model is then fed into the mechanical model compels a physical robot to move the mock-up satellite and the spacecraft-manipulator system. There are two laboratory concepts shown in Figure 3.2.

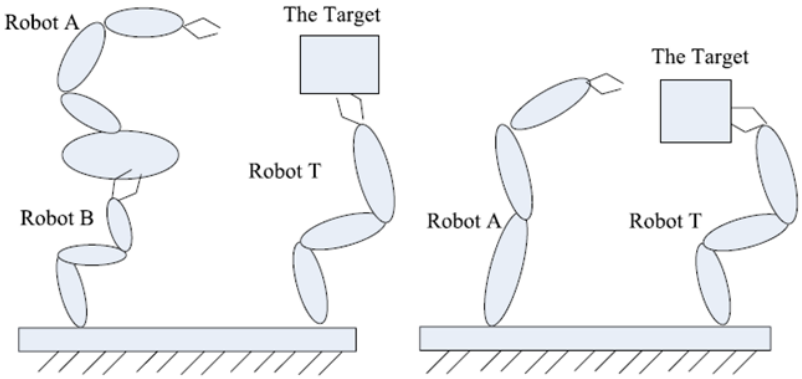


Figure 3.2 Dual-robot testbed Concepts [76]

On the left, the target is mounted on Robot T's end-effector and Robot T's motion is computed from the given motion of the target in space. Robot A's base is mounted on Robot B's end-effector. Robot A's motion is computed using the equations of free-floating robot, while Robot B's motion is calculated from the base spacecraft reactions to the manipulator motions. This is done in order to decouple the manipulators motion and base spacecraft attitude disturbance.

On the right, Robot A's base is fixed on the floor and its motion is computed from the 6DOF space robotic manipulator EE. This includes the manipulator motion and the base spacecraft reactions. Since the concept on the right side is relatively simple in hardware, we choose to design a ground

experiment system based on it. In comparison with the left side, which requires specially special design of two robots mounted over one another.

The design of the HIL testbed consists of two sub-systems, shown in Figure 3.3. The motions of the blue components (space manipulator and target) are simulated, and the red components are real robots that will achieve the simulated motion in the real-world.

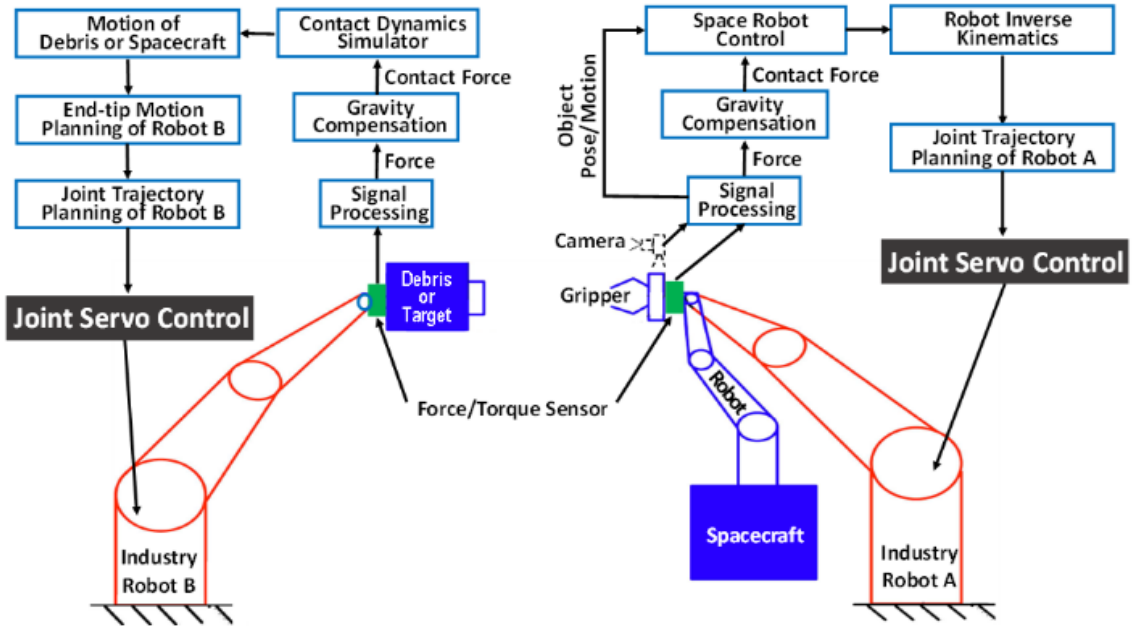


Figure 3.3 Schematic of Dual-robot HIL Testbed

The testbed includes two identical 6DOF industrial robotic manipulators to deliver the computer-generated 6DOF relative dynamic motions of the space robotic end-effector and the target.

A mock-up target is mounted on the end-effector of one robot (B), while a robotic finger gripper is mounted to the end effector of the second robot (A)

with a camera in the eye-in-hand configuration. The gripper can autonomously track, approach, and grasp the target guided by the camera.

Two computers are used to emulate the dynamic responses of the free-floating spacecraft-borne robot, and the free-floating target based on multibody dynamics and contact models in microgravity environment.

Once the path of the space manipulator's EE is planned in simulation, we use kinematic equivalence [76] in order to execute the EE's path by the industrial robot. Kinematic equivalence means that the EE's motion of the lab robot equals that of the space manipulator, i.e. the EE's motion of the space manipulator is realized by the lab robot. Using the experiment system, the algorithms of the space robots of any geometry and mass properties can be tested. There are two modes of kinematic equivalence to be described.

Figure 3.4 shows the first mode of kinematic equivalence, where the base of robots A and robot B are fixed with respect to the inertial frame. This means that Σ_A and Σ_B are fixed relative to Σ_I, but Σ_S is free floating relative to Σ_I. In the lab testbed, robot A is used to implement the absolute motion of robot S's end-effector, and robot B is used to implement the motion of the target. Robot A's motion is planned based on the hand-eye configuration and measurements. And the target motion is determined according to the target dynamics.

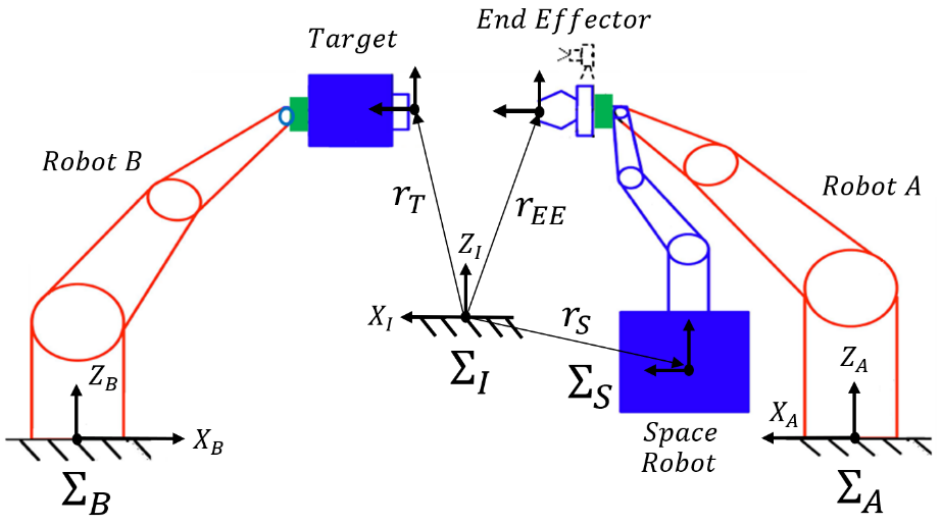


Figure 3.4 First mode of Kinematic equivalence

After generating the space manipulator's EE path in simulation, the EE's pose relative to (Σ_A) is calculated in Eq.(4.18).

$$^{A}T_{EE} = (T_A)^{-1} T_{EE} \tag{4.18}$$

Afterwards, the desired joint angles for industrial robot A is calculated using the inverse kinematic equation of robot A as seen in Eq.(4.19).

$$\Theta_{Ad} = f_A^{-1}(^{A}T_{EE}) \tag{4.19}$$

where f_A is the direct kinematic equation of robot A, calculated using the DH parameters of the Fanuc M-20iD/25 robotic manipulators [89] shown in Table 3.2.

For the Space Target, we calculate its path with respect to the camera (mounted on the space manipulator's EE) in simulation, and the calculated path is achieved by the industrial robot B using Eq.(4.20).

$$^{B}T_{T} = (T_{B})^{-1}T_{T} \tag{4.20}$$

The desired joint angles for the industrial robot B are then calculated using the inverse kinematic equation of robot B as seen in Eq.(4.21).

$$\Theta_{Bd} = f_{B}^{-1}(^{B}T_{T}) \tag{4.21}$$

where f_B is the direct kinematic equation of robot B, calculated exactly as robot A earlier.

Figure 3.6 shows the second mode of kinematic equivalence, unlike the first mode, Σ_A and Σ_B are fixed relative to Σ_S. This means that Robots A and B are used implement the motion of the space robotic EE and target relative to the space base. The relative motions are computed according to the absolute motion of the EE, target, and the space base.

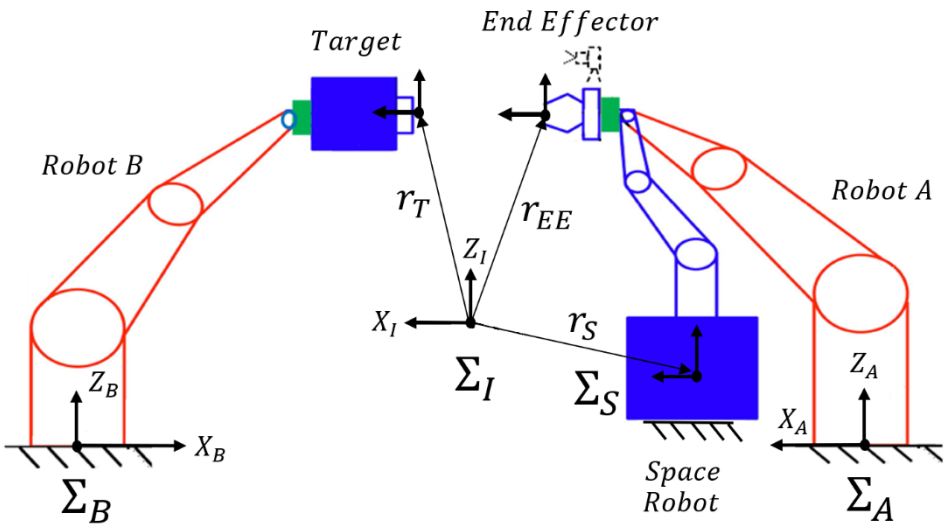


Figure 3.5 Second mode of Kinematic equivalence

After generating the space manipulator's EE path in simulation, the

EE's pose relative to (Σ_A) is calculated in Eq.(4.22).

$$^{A}T_{EE} = {}^{A}T_{S}(T_{S})^{-1}T_{EE} \tag{4.22}$$

Afterwards the desired joint angles of Robot C are determined using equation (4.19). The target pose is also calculated by its dynamics. Then the end-effector pose of robot B is calculated in (4.23).

$$^{B}T_{T} = {}^{B}T_{S}(T_{S})^{-1}T_{T} \tag{4.23}$$

The desired joint angles of Robot B are computed using Eq.(4.21). For this mode, the matrixes $^{A}T_{S}$ and $^{B}T_{S}$ are constants and can be determined beforehand.

3.2.1 Robotic Manipulator System

The 6DOF dynamic motions of the free-floating target and the free-floating end effector of the space robotic manipulator (shown in blue in Figure 3.3) are delivered by two Fanuc M-20iD/25 robotic manipulators [89]. See Figure 3.6(a). These robots have six links and can control the end effector's pose (3DOF translational and 3DOF rotational motion) with great accuracy. The payload capacity and reach of the end effector are 25kg and 1,831 mm, respectively. Each robot is controlled individually by one computer.

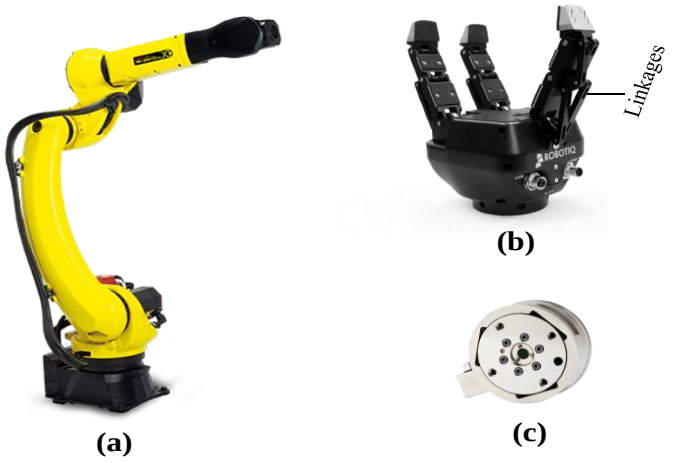


Figure 3.6 (a) Fanuc Manipulator, (b) Robotiq gripper,

(c) ATI force/load sensor

The robotic arm with a mock-up target attached to its end effector; robot (B) in Figure 3.3, is designed to mimic the target's free-floating motion in space. A force and torque sensor from ATI industries shown in Figure 3.6(c) is mounted between the end effector and the mock-up target as shown in green in Figure 3.3. It measures the force and torque caused by the weight of the mock-up target and the contact force in case capture occurs. These measurements, together with the contact forces measured by the tactile sensors on the gripper, can provide the contact load on the target. The latter is then fed into the dynamic model of the target. The target's motion is simulated, either free-floating motion before capture or controlled motion after being captured by the gripper with contact force input. Afterwards, the target's simulated motion is converted into joint commands that can be fed into the

industrial manipulator.

The Fanuc industrial robot is designed for manufacturing environments, where it consistently reaches the same desired position with high repeatability (less than 0.02 mm) at high speeds. However, using an Ethernet cable, we have attained an interface between the FANUC robot main controller and our external computer. With the StreamMotion protocol, we can control the six robotic joints separately by feeding a stream of joint commands from an external computer (using C++/Python codes) into the FANUC main control box. The stream of joint commands must first be sent/received within the allowable timestep (4-8ms), and the joint commands must be reachable. With this interface, we are able to move the robotic arm (all joints) with the path we desire from our computer simulations.

The chaser robot (A) in Figure 3.3 simulates the free-floating motion of the space robotic manipulator in micro-gravity. The robot is equipped with a gripper from Robotiq, see Figure 3.6(b), which is connected to the end effector via an ATI force/load sensor. The gripper has three individually actuated fingers, each with three links. However, only the base link can be controlled in two directions. The other two links are connected to the base link with linkage assemblies that allow for adaptive motion to different surface profiles of the target, see Figure 3.6(b).

The robotic manipulator is controlled by an external computer, where

the motions of the end effector are simulated by considering the dynamic coupling between the space robot and the base spacecraft in microgravity. During the capture phase, when contact is made, the ATI sensor measures the resultant force and torque, while the tactile sensors on the gripper measure the contact forces. This combination of measurements provides a detailed understanding of the pure contact load on the target and the space robotic manipulator in microgravity, allowing for active gravity compensation. The contact load information is then fed into the computer, which generates commands to control the real-world robotic manipulator's joint angles and the gripper's fingers.

Table 3.2 includes the Denavit–Hartenberg parameters for the Fanuc M-20iD/25 industrial robotic manipulator [89]. Using these parameters we calculate the Forward and Inverse kinematics of the physical robots and use the kinematic equivalence equations (4.19) and (4.21), to mimic the motions of the simulated space robotic EE and target.

Table 3.2 Fanus Industrial Robot DH Parameters

Kinematics	θ [rad]	a [m]	d [m]	α [rad]
Joint 1	θ_1	0	0.075	$\pi/2$
Joint 2	θ_2	0	0.84	π
Joint 3	θ_3	0	0.215	$-\pi/2$

Joint 4	θ_4	-0.89	0	$\pi/2$
Joint 5	θ_5	0	0	$-\pi/2$
Joint 6	θ_6	-0.89	0	π

The Denavit–Hartenberg parameters are a standardized set of four parameters used to describe the kinematic configuration of a robot manipulator, specifically to model the relationship between consecutive joints and links. For an i number of joints, the four parameters include: the joint angle (θ_i), that represents the angle between the x_i axis and the x_{i+1} axis, measured around the z_i axis. The length og each link (a_i) that describes the distance between the axes of two consecutive joints along their common normal. The link offset (d_i) describing the distance along the z_i axis of the current frame from the origin of one link frame to the next. And link twist (α_i), the angle between the z_i axis of the current joint and the z_{i+1} axis of the next joint, measured along the x_i axis.

3.2.2 Capture Interface

In Section 2.3 we can see that the previous space robotic testing facilities have a very simple Capture interface that is use only to achieve pose alignment between the robotic EE and target. A summary is shown in Figure 3.7. They

lack the capability of capture and post capture stabilization of the target, which requires a complex interface such as a gripper and different sensor capabilities.

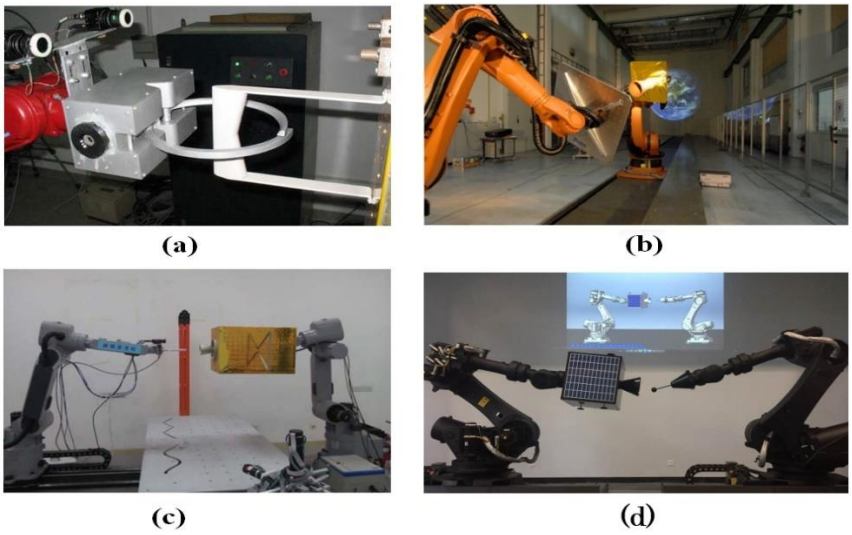


Figure 3.7 (a) Shenzhen Space Technology Center [76], (b) German Aerospace Center [77], (c) China Academy of Space Technology [81]. (d) Tsinghua University [79]

In our lab, in order to grasp a stationary or tumbling target, we use the Robotiq 3-finger gripper as a capturing interface system. The specifications of this gripper listed in Table 3.3.

Table 3.3 Robotiq 3-finger gripper specifications	
Gripper Opening	0 to 155 mm
Gripper Weight	2.3 kg
Object diameter for encompassing	20 to 155 mm
Maximum recommended payload (encompassing grip)	10 kg

Maximum Recommended Payload (Fingertip Grip)	2.5 kg
Grip Force (Fingertip Grip)	30 to 70 N

There are several reasons why the Robotiq 3-Finger gripper is a popular choice among researchers and engineers in the field of robotics:

1. **Flexibility**. As an underactuated gripper, the Robotiq 3-Finger is capable of adapting its configuration to the shape of the object being grasped, providing a high level of flexibility and reliability. This allows it to be used in a wide range of applications, from grasping irregularly shaped objects to performing delicate manipulation tasks.
2. **Repeatability**. The gripper is capable of providing a high level of repeatability with a precision of less than 0.05mm. This makes it suitable for use in applications where precise grasping and manipulation are required. Its movement is shown in Figure 3.8.
3. **Payload capacity**. The gripper can carry a relatively large payload of up to 10 kg, making it suitable for use in applications where heavy objects need to be manipulated.
4. **Compatible with most robots**. The gripper is compatible with most industrial robot manufacturers and can be controlled via Ethernet/IP, TCP/IP, or Modbus RTU. This means that it can be integrated with existing robotic systems.

5. **Grasping modes**. The gripper offers four pre-set grasping modes (scissor, wide, pinch, and basic) that allow for a wide range of grasping setups, making it suitable for different types of objects.
6. **Grasping force and speed**. The gripper can produce a grasping force up to 60N and different grasping modes for different types of objects. This allows the user to select the appropriate mode for the specific task at hand, ensuring that the gripper can pick up any object of any shape safely.

Figure 3.8 Robotiq 3-Finger Gripper Movement

3.2.3 Mock-up Target Satellite

The mock-up target used in this experiment is a scale-down model of a typical satellite. It is a 30cm cube made of one-eighth inch thick aluminum plates. It is directly bolted to the ATI force/torque sensor, and the latter is bolted to the end effector. To mimic the appearance of a real satellite, the cube is wrapped in shiny thermal sheets that reflect light in a similar manner. A dummy thrust nozzle, coupling ring, and solar panel are attached to the mock satellite, as shown in Figure 3.9. The testing room's illumination can be

controlled to simulate a space environment, such as using a single light source to replicate the sun. This set-up is intended to train AI-enhanced computer vision to recognize the target's pose in the simulated space environment.

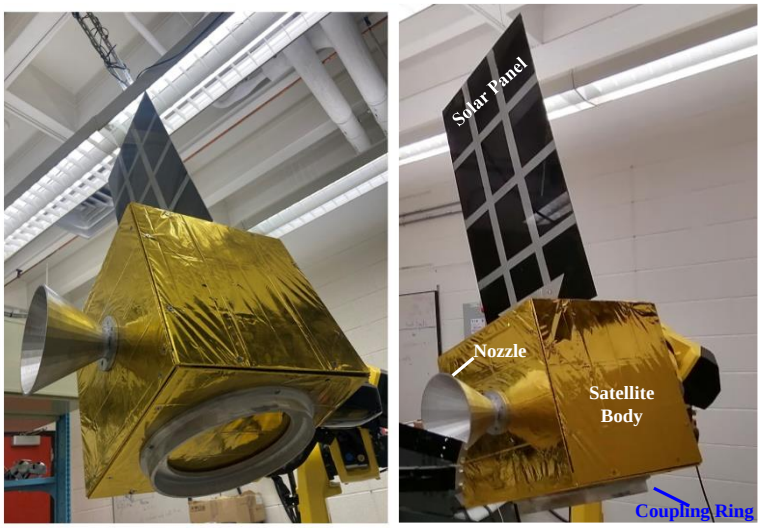


Figure 3.9 Mock-up Satellite and Components

3.2.4 Sensing Systems and Computer vision

The sensing system in the HIL testbed includes two ATI force/torque sensors, tactile sensors (Figure 3.10(a)) at each link of the gripper's fingers, and an Intel® RealSense™ depth camera D455 (Figure 3.10(b)).

Figure 3.10 (a) Tactile Sensor, (b) Intel Camera

The tactile sensors are made from pressure sensors that are bonded to the finger links. They measure the local contact forces between the gripper and the target to ensure the safety of the gripper and the target. The camera is used to determine the target's pose by either photogrammetry or AI-enhanced computer vision algorithms. This information is fed into the robotic control algorithm that guides the robot to perform autonomous tracking and grasping of the target.

The tracking of the target is done using AI computer vision – Yolo [90]. The computer vision algorithm was trained to track the relative motion of the target using the Intel depth camera, shown in Figure 3.10(b). The system was used to train the AI computer vision algorithm – Yolo V5, to track the target's position and orientation with a Hand-Eye configuration. First, the algorithm learns to identify the target and its relative positions, then learns to identify

the attitude by looking at three features of the target: nozzle, coupling ring, and solar panel from various angles.

Furthermore, six (6) high-resolution (2K) TV cameras are mounted in the testing room at different locations to monitor the testing from six angles to provide ground truth. The motions of two robotic manipulators are captured from six different angles as shown in Figure 3.11. This allows us to monitor the tracking/capturing and post-capture stabilization control thoroughly.

Figure 3.11 Full-scale monitoring of all angles

3.2.5 Computer System

Two computers are employed to control the hardware-in-the-loop (HIL) testbed in our lab. The first computer is responsible for controlling the robot and mock-up target. This computer utilizes an open-loop forward control system and takes as input the resultant contact forces in case of a capture

event. It then calculates the 6DOF motion of the target in real-time and converts it into commands for the robot's joint angles. These commands are then transmitted to the robotic control box to move the target as if it were in a microgravity environment. It is important to note that this computer does not require high computing power and a Dell desktop computer XPS 8940 is used.

The second computer is responsible for collecting measurements from the sensing systems (camera, tactile, and force/torque sensors), calculating the 6DOF motion of the space robot, and then controlling the motions of the robotic manipulator and gripper. In case of a capture event, the computer must control the robotic manipulator and gripper to synchronize the motion of the gripper with the target. This process requires a high level of computing power and accordingly a Lambda™ GPU Desktop PC - Deep Learning Workstation for this purpose. Additionally, this computer can also be used for monitoring and recording the data during the test and as a backup control system in case of any failure or malfunction. The integrated HIL testbed is shown in Figure 3.12.

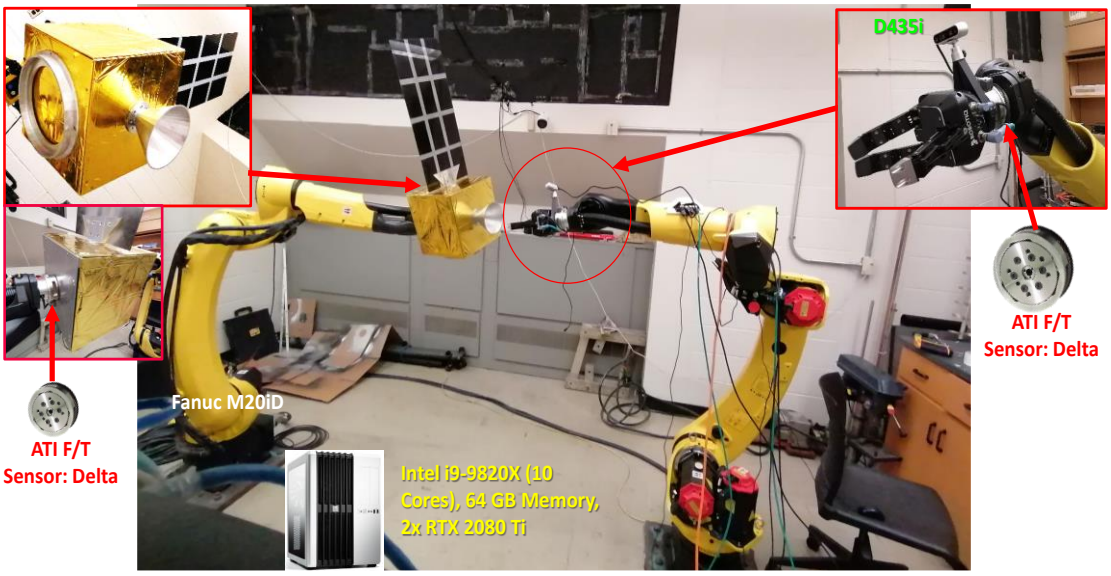


Figure 3.12 York University 6DOF hardware-in-the-loop ground testbed

Chapter 4 REINFORCEMENT LEARNING

Summary: This chapter presents the detailed description of the Reinforcement learning (RL) algorithm adopted in this work, the Deep Deterministic Policy Gradient (DDPG). A unified control framework is presented for the purpose of precise control of the space robotic manipulator joints, where the motion planning problem is recast into an RL reward maximization problem. A Comprehensive descriptions of key RL components such as the Agent, Environment, Control Actions, States/Observations, and the reward function. A robust method for constructing the reward function is introduced, designed to ensure optimal alignment of the robotic end-effector with the target pose under both simulated and real-world conditions. This chapter interpolates material from three published papers by the authors in Reference papers D, F, and G.

4.1 Deep Deterministic Policy Gradient

The motion planning and control method adopted in this study is Reinforcement Learning. RL has the ability to learn a policy for decision-making by interacting with the environment while using a pre-defined reward function instead of relying solely on input data sets [40]. It will find the best

sequence of actions that will generate the optimal outcome. This happens by allowing a piece of software called an agent to explore, interact with, and learn from the environment. The agent can take an action which affects the environment, changing its state, and the environment then produces a reward for that action. This can be seen in Figure 4.1.

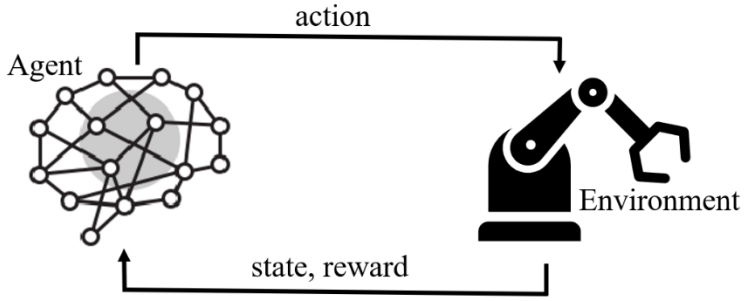


Figure 4.1 Simple Description of RL

Reinforcement learning algorithms are designed to model decision-making processes using a general MDP [91], where the outcome of an action depends on the current system and the applied action. Its objective is to drive the end-effector of a space robotic manipulator towards a specified target smoothly with precise position and orientation alignment, and without any collisions between the manipulator's links, the target, and any obstacles along the path. With this in mind the relevant states observed from the environment are defined as:

$$\boldsymbol{s}=\left[\boldsymbol{\Phi},\dot{\boldsymbol{\Phi}},\boldsymbol{p}_e,\alpha_e,\beta_e,\gamma_e,\dot{\boldsymbol{p}}_e,\boldsymbol{\omega}_e,\boldsymbol{p}_T,\alpha_T,\beta_T,\gamma_T,\dot{\boldsymbol{p}}_T,\boldsymbol{\omega}_T,\boldsymbol{p}_{Ob},d,\vartheta\right] \tag{4.24}$$

where subscripts "T", "e", and "Ob" refer to the target, the end-effector, and

obstacle, respectively. $(\boldsymbol{\Phi}, \dot{\boldsymbol{\Phi}})$ are the joint angles and angular velocities of the robotic links, $(\mathbf{p}_e, \alpha_e, \beta_e, \gamma_e)$ and $(\mathbf{p}_T, \alpha_T, \beta_T, \gamma_T)$ are the poses of the end-effector and the target, respectively, $(\boldsymbol{p}_{Ob})$ is the obstacle position, $(\dot{\mathbf{p}}_e, \boldsymbol{\omega}_e)$ and $(\dot{\mathbf{p}}_T, \boldsymbol{\omega}_T)$ are the velocities of the end-effector and the target, respectively. d and ϑ are the distance and orientation difference between the EE and the target, respectively, to be described in detail through Section 4.3.

The actions that control the robot manipulator are the torque inputs for each joint:

$$\mathbf{a} = \left[u_1, u_2, \cdots, u_n\right]^T \in \mathbb{R}^n \tag{4.25}$$

Assuming that the continuous system dynamics between states $\mathbf{s}$ and the actions $\mathbf{a}$ are discretized in the time domain at time step t_i, we obtain:

$$\boldsymbol{s}_{i+1} = \boldsymbol{f}\left(\boldsymbol{s}_i, \boldsymbol{u}_i, \boldsymbol{w}_{s,i+1}\right) \tag{4.26}$$

where $\boldsymbol{u}_i$ is the vector of control input experienced by the system and $\boldsymbol{w}_{s,i} \in \mathbb{N}(0, \boldsymbol{Q}_{s,i})$ is the noise vector that represents environmental disturbances and unmodeled system dynamics; factors that are not explicitly accounted for in the model but can still affect the system's performance.. In simulation his noise is assumed to obey a zero mean multivariate normal distribution N with covariance $\boldsymbol{Q}_{s,i}$. The control input is obtained by a function $\boldsymbol{g}$ that maps the control command action $\mathbf{a_i}$ output directly from a state-feedback control policy

based on the observed state $\mathbf{o}_i$, such that:

$$\mathbf{u_i} = \mathbf{g}\left(\mathbf{a_i}, \mathbf{o_i}, \mathbf{w_{a,i}}\right) \tag{4.27}$$

where $\mathbf{w_{a,i}} \in \mathbb{N}(0, \mathbf{Q_{a,i}})$ is the execution error vector, which obeys a zero mean multivariate normal distribution $\mathbb{N}$ with covariance $\mathbf{Q_{a,i}}$.

With the perfect observability assumption (all observations can be acquired accurately), the observed state $\mathbf{o}_i$ is obtained by an observation function $\boldsymbol{h}$,

$$\mathbf{o_i} = \mathbf{h}\left(\mathbf{q_i}, t_i, \mathbf{w_{o,i}}\right) \tag{4.28}$$

where $\mathbf{o}_i \in \mathbb{O}$ is the observation vector of the system state, $\mathbf{w_{o,i}} \in \mathbb{N}(0, \mathbf{Q_{o,i}})$ is the random processing noise in the state estimation with a zero mean and covariance $\mathbf{Q_{o,i}}$.

The optimal path planning for such a system is to find the best control input $\mathbf{u_i}(i \in \mathbb{Z})$ or an optimal policy π, to maximize the performance function J over a trajectory ρ.

$$\max_{\rho \sim \pi} J = \max_{\rho \sim \pi} \left[e\left(\mathbf{s}_0, t_0, \mathbf{s}_N, t_N\right) + \sum_{i=0}^{N-1} k_i \mathbf{f}\left(\mathbf{s}_i, \mathbf{u}_i, \mathbf{w}_{\mathbf{s},i+1}\right) \right] \tag{4.29}$$

where $e\left(\mathbf{s}_0, t_0, \mathbf{s}_N, t_N\right)$ is the terminal conditions, k_i is the weight factor, and the trajectory is a sequence of state-action pairs, satisfying the dynamics

$\rho = [(\mathbf{s}_0, \mathbf{a}_0), (\mathbf{s}_1, \mathbf{a}_1), \cdots, (\mathbf{s}_N, \mathbf{a}_N)]$.

Equation (4.29) can be recast as an RL agent to maximize a scalar return by learning an optimal policy π^* in interaction with the environment (the robot in the current work) as follows.

Let the policy $\pi : \mathrm{S} \to \mathrm{A}$ with a set of parameters $\theta \in \mathbb{R}^n$ and express it in the episodic formulation of RL. Then, the optimal policy π^* is the one that maximizes a performance index J in terms of the expectation $\underset{\tau \sim \pi_\theta}{\mathbb{E}}$ of the parameter θ^*, which is the discounted sum of rewards obtained along the trajectory ρ, such that"

$$\begin{aligned} \theta^* &= \arg\max_{\theta} J(\theta) \\ J(\theta) &= \underset{\rho \sim \pi_\theta}{\mathbb{E}} \left[\sum_{i=0}^{N-1} \gamma^i R_i \left(\mathbf{s}_{\mathbf{i}+1}, \mathbf{s}_{\mathbf{i}}, \mathbf{a}_i \right) \right], \quad \gamma \in [0,1] \end{aligned} \tag{4.30}$$

where N is the number of time steps in each episode, $R_i\left(\mathbf{s}_{\mathbf{i}+1}, \mathbf{s}_{\mathbf{i}}, \mathbf{a}_{\mathbf{i}}\right)$ is the immediate reward that maps state-action pairs $(\mathbf{s}_i, \mathbf{a}_{\mathbf{i}})$ to a scalar value and propagates the state $\mathbf{s}_{\mathbf{i}}$ to the state $\mathbf{s}_{\mathbf{i}+1}$ under the action $\mathbf{a}_{\mathbf{i}}$ under the control input $\mathbf{u}_i$. The parameter γ is a discount factor. For long-term planning, the agent treats all future rewards by $\gamma = 1$. Conversely, the agent is for short-term planning if $\gamma \ll 1$.

Equation (4.30) can be solved by the MDP with the stochastic gradient ascent update on the parameter θ [57].

$$\theta_{i+1} = \theta_i + \alpha \, \nabla_\theta J(\theta_i) \tag{4.31}$$

where $\alpha \in (0,1)$ is the learning rate and $\nabla J(\theta_i)$ is the policy gradient.

Note that a differentiable control policy is always available in the path planning problem of a space robotic manipulator. Consequently, the actions involved in the MDP are not stochastic. Then, a deterministic action can be learned directly from given states, instead of learning a set of probability distributions. Accordingly, the DDPG algorithm [91] is employed to estimate the policy gradient $\nabla J(\theta_i)$ as follows.

Let $\mu(\theta)$ be a deterministic policy and $\beta(\theta)$ a behavior policy with trajectories $\beta_\theta(\mathbf{a}|\mathbf{s})$ distinct from the deterministic policy trajectories $\mu_\theta(\mathbf{a}|\mathbf{s})$. Then, the deterministic policy gradient can be estimated by trajectories sampled from a distinct behavior policy $\beta(\mathbf{a}|\mathbf{s})$ based on an Actor-Critic approach [92], such that,

$$\nabla_\theta J(\theta) = \mathop{\mathbb{E}}_{\beta \sim \mu}\left[\sum_{i=0}^{N-1} \nabla_\theta \mu_\theta(\mathbf{a}_\mathbf{i}|\mathbf{s}_\mathbf{i}) \nabla_a Q_{\mu_\theta,i}(\mathbf{s}_i, \mathbf{a}_i)\right] \tag{4.32}$$

$$Q_{\mu_\theta,i}(\mathbf{s},\mathbf{a}) = \mathop{\mathbb{E}}_{\beta \sim \mu}\left[\sum_{i=0}^{N-1} \gamma^i R_i \,|\, \mathbf{s} = \mathbf{s}_i, \mathbf{a} = \mu_\theta(\mathbf{s}_i)\right] \tag{4.33}$$

Here, $Q_{\mu_\theta}(\mathbf{s},\mathbf{a})$ is the action-value function or Q-function that represents

the expected discounted return calculated by the state-action pair $\left[\mathbf{s}_i, \mathbf{a}_i = \mu_\theta(\mathbf{s}_i)\right]$ and corresponding rewards based on the policy μ_θ from the i^{th} time step until the end of the episode. The sole dependence of the Q-function on the environment allows for off-policy Q-learning [93] using trajectories generated from a different stochastic behavior policy $\beta(\mathbf{a}|\mathbf{s})$. Based on the greedy policy $\mu(\mathbf{s}) = \arg\max_a Q_{\mu_\theta}(\mathbf{s}, \mathbf{a})$, the Q-function is approximated by the Bellman equation [94].

$$Q_{\mu_\theta}(\mathbf{s}_i, \mathbf{a}_i) = \mathop{\mathbb{E}}_{\beta\sim\mu}\left[R(\mathbf{s}_i, \mathbf{a}_i) + \gamma Q_{\mu_\theta}(\mathbf{s}_{i+1}, \mu(\mathbf{s}_{i+1}))\right] \tag{4.34}$$

Then, the Q-function can be evaluated recursively over a parameter θ^Q using temporal difference learning.

$$\begin{aligned} &Q_{\mu_\theta}^{j+1}(\mathbf{s}_i, \mathbf{a}_i) = Q_{\mu_\theta}^{j}(\mathbf{s}_i, \mathbf{a}_i) + \\ &\alpha\left\{\mathop{\mathbb{E}}_{\beta\sim\mu}\left[R_i(\mathbf{s}_i, \mathbf{a}_i) + \gamma Q_{\mu_\theta}(\mathbf{s}_{i+1}, \mu(\mathbf{s}_{i+1}))\Big|_{\theta^Q}\right] - Q_{\mu_\theta}^{j}(\mathbf{s}_i, \mathbf{a}_i)\right\} \end{aligned} \tag{4.35}$$

The learning (function approximation) is achieved by minimizing the loss function $L(\theta^Q)$:

$$\begin{aligned} L(\theta^Q) &= \mathop{\mathbb{E}}_{\beta\sim\eta}\left[\left(Q_{\mu_\theta,i}(s_i, a_i|\theta^Q) - y_i\right)^2\right] \\ y_i &= R_i + \gamma^i Q_{\mu_\theta,i}\left(\mathbf{s}_{i+1}, \mu(\mathbf{s}_{i+1})\middle|\theta^Q\right) \end{aligned} \tag{4.36}$$

Applying Q-learning to continuous systems like the robotic manipulator is challenging because it requires optimizing actions $\mathbf{a_i}$ at each time step. Instead, a Deep Q-Network (DQN) approach [91] is used to approximate the Q-

function by Deep Neural Networks (DNN) with state and action as input and expected cumulative reward as output. The DQN algorithm involves two DNNs: main and target network. The DDPG extends this two DNN in its structure, the Actor is split into online and target actor networks, and the Critic is split into online and target networks as shown in Figure 4.2.

The Actor returns actions deterministically in a continuous space based on its interaction with the robotic manipulator (environment) via the parameterized policy μ_{θ^Q}, and is updated by gradient ascent on the policy gradient Equation (4.31). The Critic network evaluates the agent's performance by estimating the Q-value of state-action pairs $(\mathbf{s}_i, \mathbf{a}_i)$ using Equation (4.34) to guide the generation of next action.

To ensure the samples are independently distributed, a replay buffer is used to save the tuple $(\mathbf{s}_{i+1}, \mathbf{s}_i, \mathbf{a}_i, R_i)$ in stacks. The tuples are sampled from the environment in accordance with the exploration policy. At each time step, Actor and Critic are updated by uniformly sampling a minibatch from the buffer. Finally, to improve the Q-learning stability, the following update schemes are introduced for the target Actor and Critic parameters by:

$$\begin{aligned} \theta_{Q,i+1} &= \tau\theta_{Q,i} + (1-\tau)\theta_{Q,i+1} \\ \theta_{\mu,i+1} &= \tau\theta_{\mu} + (1-\tau)\theta_{\mu,i+1} \end{aligned} \tag{4.37}$$

where $\tau \ll 1$ is the smoothing factor. The stability of policy gradient methods

has been well-studied under the Lyapunov framework. [95] proved that the policy gradient is negative statistically, which is a crucial factor in ensuring the stability of DDPG methods from a statistical standpoint. The implementation of the DDPG algorithm is shown in Figure 4.2. The pseudo code that describes the detailed training process for the DDPG algorithm is shown in Table 4.1 [96].

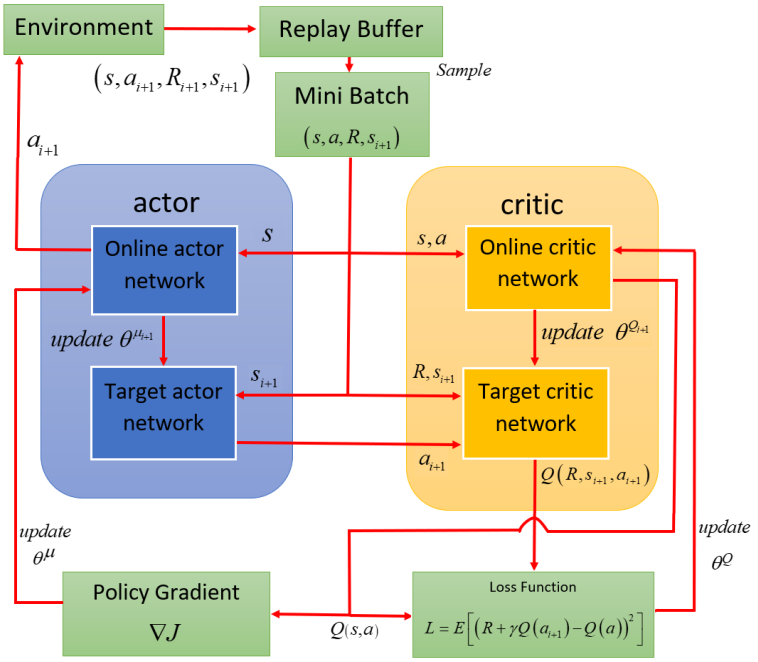


Figure 4.2 Implementation chart of DDPG algorithm

Table 4.1 DDPG Algorithm Procedure [96]

Randomly initialize online actor $\mu(\mathbf{s}|\theta^{\mu})$ and critic $Q(\mathbf{s},\mathbf{a}|\theta^{Q})$ networks by respective weights $(\theta^{\mu}, \theta^{Q})$.

Reset target actor μ' and critic Q' networks with weights $\theta^{\mu} \rightarrow \theta^{\mu'}$,

$\theta^Q \rightarrow \theta^{Q'}$.

Reset replay buffer (RAM).

For each episode M do:

For each timestep:

Select an action $\mathbf{a} = \mu(\mathbf{s} \mid \theta^\mu) + \mathbb{N}$, where s and $\mathbb{N}$ are current state and random process exploration noise.

Execute action a, observe reward R & new state $\mathbf{s}_{i+1}$.

Store the experience $(\mathbf{s}, \mathbf{a}, R_i, \mathbf{s}_{i+1})$ in the replay buffer.

Sample random minibatch of experiences $(\mathbf{s}, \mathbf{a}, R_i, \mathbf{s}_{i+1})$ from the replay buffer.

If $\mathbf{s}_{i+1}$ is the terminal state, set the value function target as

y = R

Else, Set the value function target as

$$y = R + \gamma Q'(\mathbf{s}_{i+1}, \mu'(\mathbf{s}_{i+1} \mid \theta^{\mu'}) \mid \theta^{Q'})$$

Update critic parameters by minimizing the loss L on all sampled experiences,

$$L = \frac{1}{M} \sum_{i=1}^{M} \left[y - Q\left(s, a \middle| \theta^Q\right) \right]^2$$

Update actor parameters to maximize the expected discounted reward by the sampled policy gradient,

$$\nabla_{\theta^\mu} J \approx \frac{1}{M}\sum_{i=1}^{M} \nabla_a Q\left(s, a\middle|\theta^Q\right) \nabla_{\theta^\mu} \mu\left(s\middle|\theta^\mu\right)$$

Update target actor and critic parameters by:

$$\theta^{Q'} = \tau\theta^Q + \left(1-\tau\right)\theta^{Q'} \text{ and } \theta^{\mu'} = \tau\theta^\mu + \left(1-\tau\right)\theta^{\mu'}$$

End for

End for

In Section 2.2, we observe the shift in the literature toward the Deep Deterministic Policy Gradient (DDPG) algorithm for addressing the motion planning problem of space robotic manipulators. The Reinforcement Learning algorithm employed in this study is similarly based on DDPG.

DDPG is a model-free, policy-gradient, actor-critic method renowned for its high performance in handling high-dimensional continuous action spaces. Typically, value-based methods such as deep Q-learning perform well in scenarios where the agent has a finite set of actions to choose from. In contrast, policy-gradient methods are particularly effective in environments with continuous state and action spaces.

For the proposed research, a model-free approach is advantageous due to the complex dynamics and modeling challenges of the free-floating space robotic manipulator. This algorithm does not require explicit knowledge of

model equations to derive the optimal control policy, making it adaptable to various mission scenarios.

When a stochastic policy is considered, the network returns some parameters of the underlying probability distribution (such as the mean values of a Gaussian distribution). Instead, for the deterministic policy, the network directly outputs the actions, given the current system state as input.

4.2 Agent (Actor-Critic Networks)

The Agent is a function that takes in the state and reward and gives us an action. The main classes of RL function algorithms include:

1. Policy function based.

   Policy (π) is the strategy that the agent employs to determine next action based on the current state. It trains a Neural Network that takes in the state observation and outputs the actions. It is called an Actor, and it uses Policy Gradient methods to converge.

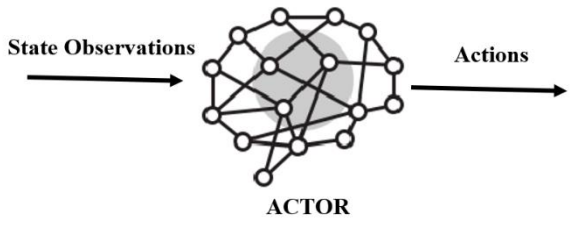


Figure 4.3 Illustration of Actor Network function

Figure 4.3 shows a typical Actor. It executes the current policy, collects the rewards. Thereby increasing the possibilities of good actions.

2. Value function based.

   Value (V) is the expected long-term return with discount, as opposed to the short-term reward R. V (π) is defined as the expected long-term return of the current state policy π . In a value-function based agent, a function would take in the state and one of the possible actions from that state. then output the value of taking that specific action. It is called the Critic, and the way it updates the value is based on some form of Bellman equation. The Bellman equation allows the Agent to solve the Q-table overtime by breaking up the whole problem into multiple simpler steps. So instead of solving the true value of the state/ action pair in one step. The agent will update the state/ action pair each time it is visited.

   It will keep doing this over again until the true values of all state/ action pairs are known to get the optimal path. The downside is that the action space needs to be small (for example it will take a very long time to converge for large or continuous actions). This brings us to the combination of the Actor-Critic based Learning algorithms. Figure 4.4 shows an Actor Network.

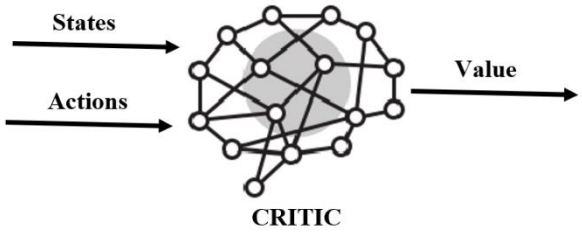


Figure 4.4 Illustration of Critic Network function

3. Actor/ Critic.

In this agent the actor tries to take the best action based on current state (policy function based). The critic is a second network that is trying to estimate the value of the state and the action pair that the actor took (value based). This works for continuous actions because the critic only looks at the action that the actor takes not all possible actions that exist.

So, the actor is trying to get the best action from the critic feedback, and the critic is getting the best value from the received rewards to properly criticize the actor. Actor-Critic based Learning algorithms the Agent takes the best parts of policy functions and value functions. This is the Agent implemented in this study.

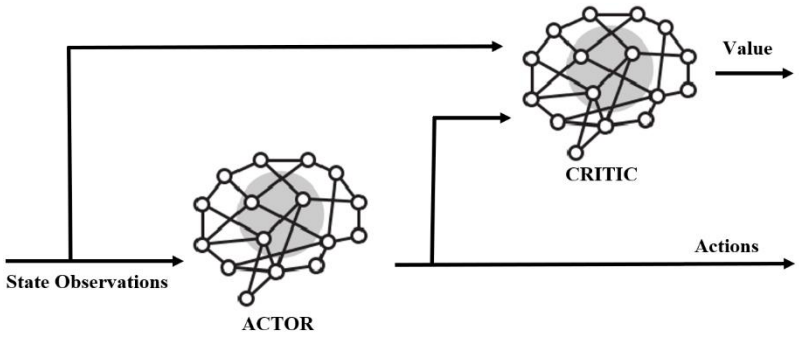


Figure 4.5 Illustration of Actor-Critic Network function

The complexity of the task directly influences the architecture of the neural networks. High-dimensional input spaces require more complex architectures to properly capture the relationships between variables.

For simple tasks with low-dimensional inputs, shallow networks may suffice. In contrast, tasks involving high-dimensional inputs such as robotic control often benefit from deeper networks. The actor and critic DNNs can be constructed by standard feed-forward neural networks with two hidden layers as shown in Figure 4.6. Hidden layers 1 and 2 have 400 and 300 neurons, respectively. This architecture is inspired by the research presented in [96] and the successful implementation of deep neural networks in various control applications, including robotics. Moreover, its effectiveness has been demonstrated for space manipulators, as highlighted in the literature summarized in Table 2.2.

The input and output of the actor network are the state and the action, respectively, while the input and output of the critic network are a state-action pair and the action value, respectively. In the current study, the robotic manipulator includes 6 links, i.e., $n = 6$. Thus, the dimensions of the state and action spaces are $\boldsymbol{s} \in \mathbb{S}^{32}$ in the state space and action $\boldsymbol{a} \in \mathbb{A}^{6}$, respectively. Accordingly, the structures of the actor and critic networks are illustrated in Figure 4.6.

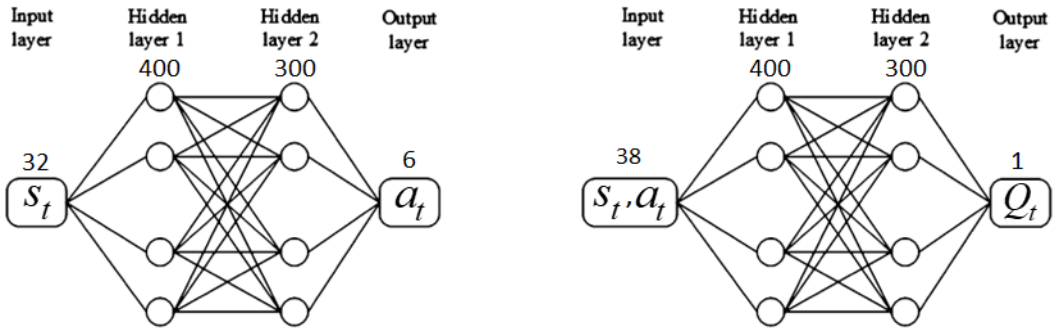


Figure 4.6 Constructed Actor and Critic networks in our system

4.3 Reward Function

As the most critical part of reinforcement learning, the reward function is directly responsible for how the algorithm trains and what objectives are achieved. Our methodology aims to expedite training progress in dynamic settings by designing reward functions that consider several critical factors.

- Aligning the manipulator's end effector (EE) with the target using a quaternion-based approach for orientation comparison.
- Incorporating joint velocity constraints to achieve smoother EE trajectories.
- Measures to prevent self-collision of the manipulator's links and, importantly, incorporates external obstacle avoidance capabilities during target approach.
- A safety shield: cone penalty constraint triggered when the manipulator's EE is in close-proximity of the target - a feature not previously implemented in the literature.

Recognizing the imperfection of real-world sensors, we further enhance our model's practical applicability by including white noise in the observations during training. This ensures that the trained agent is resilient to signal noise and capable of achieving successful target capture. The effectiveness and advantages of our proposed algorithm are validated through numerical simulations.

Lastly, we introduce a dynamic agent; the reinforcement learning agent will be trained with different initial conditions that are randomly appointed at the start of each episode, this includes the manipulator initial joint angles, target and obstacle positions. Table 2.3 highlights the gap of previous research for dynamic agents. The trained agent is expected to find a suitable path for any target location, avoiding an obstacle placed anywhere in the environment, wherever the manipulator's initial position starts.

Accordingly, the reward function must include four main Terms to incorporate these requirements. First, a guidance term must be created in order to direct the robotic EE towards the target successfully. Secondly, a sparse reward term must be introduced when successful capture is achieved. The third part includes safety terms, such as penalties (large negative reward) that activate when collision occurs, either between the robotic links with each other or with an external obstacle along the path. Another safety term includes a close proximity cone shield to ensure the EE is within a safety zone before

capture. Finally, an optimization term can be introduced in the reward function to improve the overall path, this term may include joint velocity constraints, torque minimization, energy consumption, etc. The reward function (R_t) is evaluated at each time step, and the sum of these evaluations across all time steps constitutes the episode reward.

4.3.1 Guidance Term

The primary term of the reward function is focusing on aligning the end-effector's pose with the target's pose by minimizing the pose error quickly. The pose difference is represented by the position error d and the orientation error ϑ. The position error is determined straightforward by:

$$d = \|\boldsymbol{p}_T - \boldsymbol{p}_e\| = \sqrt{(x_T - x_e)^2 + (y_T - y_e)^2 + (z_T - z_e)^2} \tag{4.38}$$

where the subscripts "*T*" and "*e*" denote the target and the end-effector, respectively.

The orientation error of the EE can be defined either by a unit quaternion (ξ) or a set of Euler angles (α, β, γ). In the previous work as shown in Table 2.2, the expression of the orientation error is based on a set of Euler angles, such that:

$$\vartheta = \sqrt{(\alpha_T - \alpha_e)^2 + (\beta_T - \beta_e)^2 + (\gamma_T - \gamma_e)^2} \tag{4.39}$$

where $(\alpha_e, \beta_e, \gamma_e)$ and $(\alpha_T, \beta_T, \gamma_T)$ are the Euler angles of the EE and the target,

respectively. It is worth pointing out that the orientation difference in such a form loses the information of rotation sequence of Euler angles. This increases the search space in the learning process and may even lead to divergence. Thus, the Euler angle representation is not used in the current work. Instead, an innovative reward function is built using the quaternion representation of orientations.

Let the orientations of the EE and the target be represented by two unit quaternions ξ_E and ξ_T, respectively, as shown in Figure 4.7.

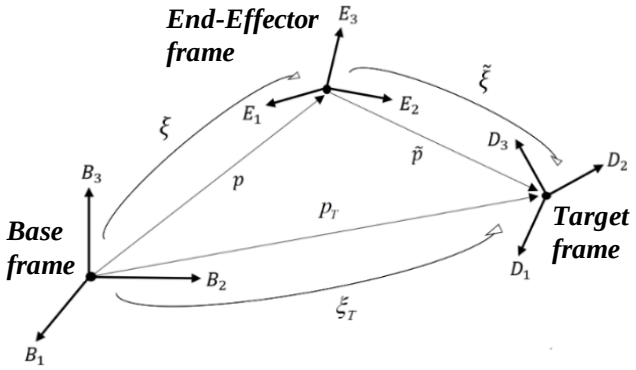


Figure 4.7 Quaternion representation of orientation difference

The orientation error of the EE with respect to the target can be expressed as [97]:

$$\tilde{\xi} = \xi_T - \xi_e = \xi_T \times \xi_e^* \tag{4.40}$$

where ξ_e^* is the conjugate of the quaternion ξ_e. Accordingly, the orientation difference between the EE and the target is determined by:

$$\vartheta = 2\arccos\left[\mathrm{Re}\left(\tilde{\xi}\right)\right] \tag{4.41}$$

As the EE approaches the desired pose, and the pose error (d , ϑ) becomes smaller, the reward given at each timestep must increase. With this in mind, the guidance term is constructed such that:

$$R_t = -K_d d - K_\vartheta \vartheta \tag{4.42}$$

where K_d and K_ϑ are positive reward coefficients for the position and orientation errors, respectively, emphasizing their importance in the error minimization process. The reward function at each timestep increases as the EE moved closer towards the target. In the end, when d and ϑ reach zero, the potential field is maximum.

Notably, As the end-effector approaches to the target pose, the pose error (d , ϑ) diminishes as well as the reward function. This can cause the agent to approach slower to obtain improved alignment. Furthermore, if an optimization penalty term is introduced in the reward function, at close proximity the penalty would become higher than the guidance term, this leads the robotic EE to stop before perfect alignment is achieved since the reward is diminished.

4.3.2 Sparse Reward Term

To achieve the desired learning outcome in such conditions, the reward

function is designed to have large positive rewards when the robotic EE achieves successful alignment, and stays within close proximity of the target, thereby ensuring the end-effector maintains its desired pose for future grasping the target throughout the remaining time steps.

Consequently, large positive rewards (r_d, r_ϑ) are added in the reward function, and are activated when the pose error (d, ϑ) falls below the pre-defined tolerances, such that:

$$R_t = -K_d d - K_\vartheta \vartheta + r_d + r_\vartheta \tag{4.43}$$

where,

$$r_d = \begin{cases} 0 & \text{if } d \geq d_0 \\ r_1 & \text{if } d < d_0 \end{cases} \quad \text{and} \quad r_\vartheta = \begin{cases} 0 & \text{if } \vartheta \geq \vartheta_0 \cup d \geq d_0 \\ r_2 & \text{if } \vartheta < \vartheta_0 \cap d < d_0 \end{cases} \tag{4.44}$$

with d_0 and ϑ_0 being pre-defined indicators for close proximity operation region for position and orientation, respectively. Once the end-effector enters this region, large positive rewards are assigned to the reward function to encourage and accelerate the agent converging towards the desired target pose.

4.3.3 Safety and Collision Terms

To facilitate safe capture, the agent must learn to avoid collisions of the robotic links between themselves, and any obstacle placed in its path. Furthermore, an additional cone safety constraint can be built into the reward function to ensure the robotic EE captures the target in a cone shape interface

(such as the dummy satellite capture interface seen in Figure 3.9). This is achieved by large negative rewards described below.

4.3.3.1 Self Collision Avoidance Term

A large negative penalty (p_l) is added to the reward function to avoid self-collision between the robotic links. If the distance between any two links is less than a predefined safe distance, the penalty p_l is added to the reward function, such that:

$$R_t = -K_d d - K_\vartheta \vartheta + r_d + r_\vartheta - p_l \tag{4.45}$$

$$p_l = \begin{cases} 0 & \text{if } \min_{i \neq j} \left\| \boldsymbol{r_i} - \boldsymbol{r_j} \right\| \geq d_l \\ p_l & \text{if } \min_{i \neq j} \left\| \boldsymbol{r_i} - \boldsymbol{r_j} \right\| < d_l \end{cases} \tag{4.46}$$

where $p_l > 0$ is a penalty for two links being too close to each other, and d_l is the minimum safety distance to avoid self collision.

4.3.3.2 Obstacle Avoidance Term

To facilitate obstacle avoidance, a substantial negative penalty (p_{Ob}) is incorporated into the reward function. This penalty (p_{Ob}) is activated whenever the distance between the obstacle's CM and any link's CM falls below a predefined safety range.

$$R_t = -K_d d - K_\vartheta \vartheta + r_d + r_\vartheta - p_l - p_{Ob} \tag{4.47}$$

$$p_{Ob} = \begin{cases} 0 & \text{if } d_{Ob} \geq d_{Ob} \\ p_{Ob} & \text{if } d_{Ob} < d_{Ob} \end{cases} \tag{4.48}$$

Figure 4.8 shows a simplified illustration of how the safety distance (d_{Ob}) is defined between any link and the obstacle.

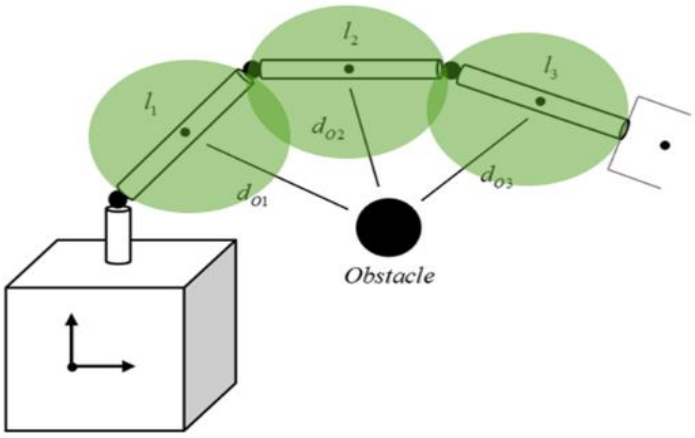


Figure 4.8 Simplified illustration of obstacle avoidance safety distances

4.3.3.3 Cone Constraint Term

In order to safely approach the target for capture, when the robotic EE is in close proximity, a penalty term is introduced if the EE position moves outside a 3D cone shape as seen in Figure 4.9.

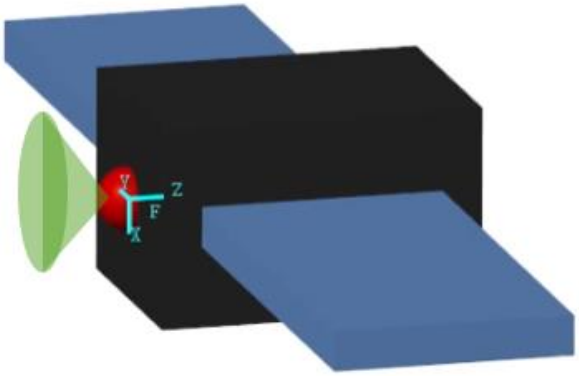


Figure 4.9 Illustration of safty cone constraint

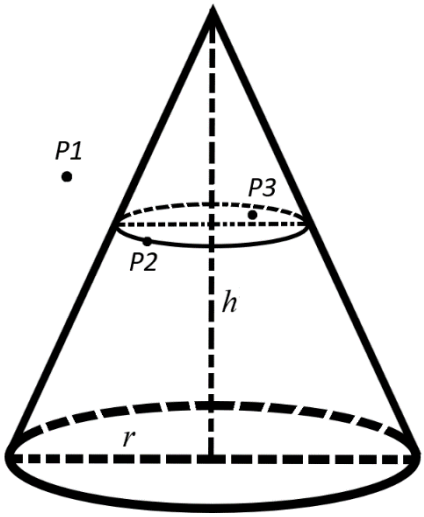


Figure 4.10 Points inside and outside the safty cone

If the axis of a right circular cone is parallel to the z-axis and passes through origin, the apex at $(0,0,h)$, base at the origin, and radius r at base, then point $P(x,y,z)$ is within the cone if and only if:

$$\begin{cases} z \geq 0 \\ z \leq 0 \\ x^2 + y^2 \leq (z-h)^2 \dfrac{r^2}{h^2} \end{cases} \tag{4.49}$$

Figure 4.10 shows a cone and what positions are considered valid. P1 would receive a negative reward, while P2, and P3 would not. The reward function becomes:

$$R_t = -K_d d - K_\vartheta \vartheta + r_d + r_\vartheta - p_l - p_{Ob} - p_{cone} \tag{4.50}$$

where p_{cone} is the penalty on the reward function if the robotic EE is located outside the safety cone constraint.

4.3.4 Optimization Term

To ensure a smooth trajectory of the end-effector with minimal fluctuations in joint velocities, a velocity penalty term is incorporated as a negative reward (p_j) into the reward function. The negative reward is proportional to the magnitude of the square sum of the joint velocities such as $-p_j \sum (\dot{\varphi})^2$ to punish abrupt joint movements. The reward function becomes:

$$R_t = -K_d d - K_\vartheta \vartheta + r_d + r_\vartheta - p_l - p_{Ob} - p_j \sum (\dot{\varphi})^2 \tag{4.51}$$

where $p_j > 0$ is a velocity penalty for smooth joint velocities by multiplying the

squared sum of joint angular velocities.

Chapter 5 SIMULATION RESULTS AND DISCUSSION

Summary: This chapter presents in detail the results of the proposed Reinforcement Learning approach for motion planning of a free-floating space robotic manipulator. First, the proposed methodology is verified on a 3DOF free-floating manipulator. Then, the methodology was extended to a more complex system and implemented on 6DOF free-floating robotic manipulator. Different reward functions are assessed and compared in detail as well as training under real-world conditions, such as sensory (observation) noise, and multiple manipulator-base mass ratios (dynamic coupling). Lastly, the proposed methodology is extended for motion planning of the free-floating space robotic manipulator with obstacle avoidance, i.e. with an external obstacle in its path.

5.1 Preliminaries

In this study, the proposed RL approach for motion planning of a free-floating space robotic manipulator is verified by simulation using MATLAB® and Simulink®. These simulations were performed on a PC equipped with an Intel® Core™ i7-10600K Processor and 32 GB of RAM. Shows the full Reinforcement Learning system.

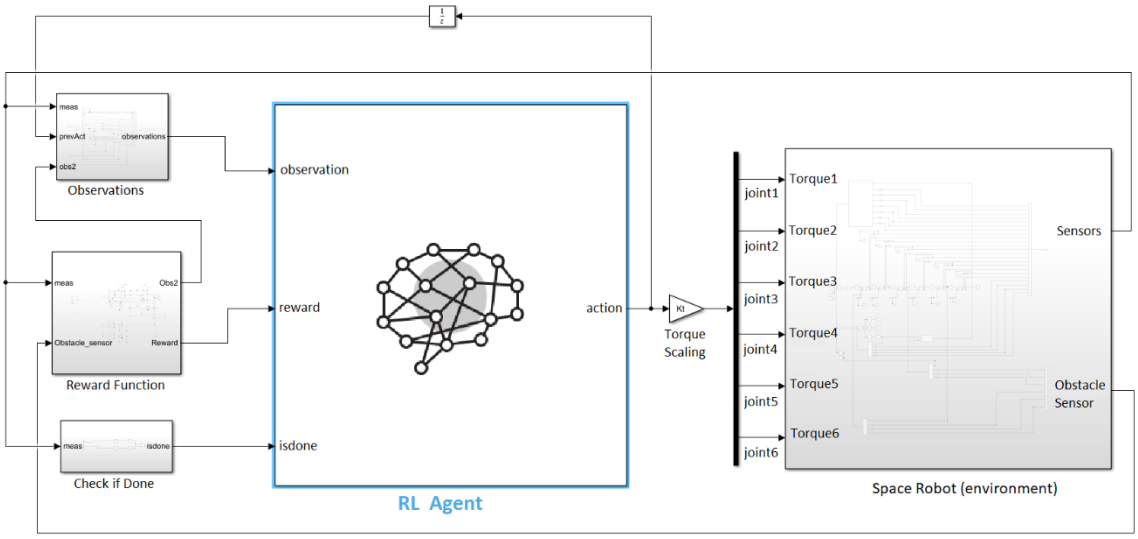


Figure 5.1 Points Scehmatic of Matlab/Simulink implementation

The Space robot environment contains a free-floating base with an n-DOF robotic arm attached to it, with the following assumptions.

- The space robotic system (containing a base, and manipulator) will consist of rigid bodies.
- The base has 6 degrees of freedom (3 for position and three for attitude) and it is not controlled but free to move in reaction with the arm.
- The space manipulator consists of n-joints; each joint has one rotational degree of freedom and is controlled by an input Torque (from the agent).

To ascertain the algorithm's performance and effectiveness under real-world conditions, the parameters used in the simulations are based on actual space robotic manipulator missions [15]. Table 1.1 shows a summary of past space robotic manipulator missions. Notably the mass ratios in most cases are

small except for the Orbital Express mission, with a 7.4% mass ratio being the highest. Furthermore, the simulation studies that have been done on space robotic manipulator systems have been reviewed and summarized these studies in Table 3.1. Accordingly, the mass ratio was selected in study to show the need to consider the free-floating base condition.

Part of theatrical and simulation results has been published in the reference paper D, F and G. All variable symbols in this chapter refer to the definitions in Section 4.3.

5.2 Three DOF Space Robotic Manipulator

To start off, the numerical simulation is done for a 3DOF free-floating space robotic manipulator to verify the effectiveness of the proposed RL control scheme. The manipulator is seen in Figure 5.2.

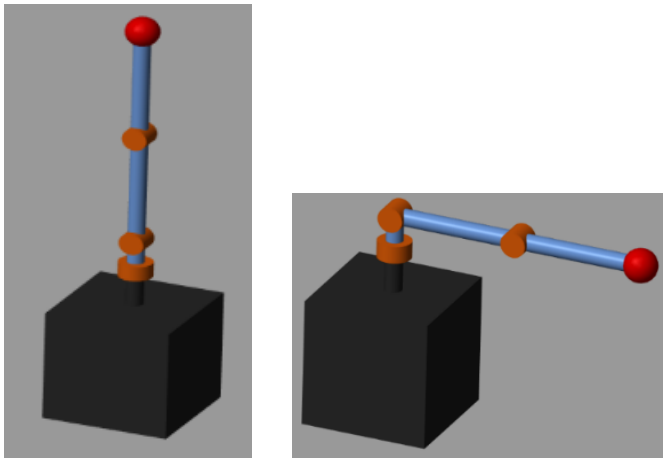


Figure 5.2 3DOF Space robotic Manipulator system

The physical parameters of the space robotic manipulator are listed in

Table 5.1. For all simulation cases, the initial position of the end-effector is $(x, y, z)_{EE} = (0, 0, 2.325)$. Similarly, the target position for case 1 is $(x, y, z)_{Target} = (1.5, 0, 1)$ and is $(x, y, z)_{Target} = (0.75, 0, 0.75)$ for the rest of the cases.

Table 5.1 Parameters of Three DOF Space Robotic Manipulator					
Body	Mass (kg)	Length (m)	*Ix* (kg/m^2)	*Iy* (kg/m^2)	*Iz* (kg/m^2)
Base	100	0.75x0.75 x0.75	9.3	9.3	9.3
Link 1	5	0.25	0.05	0.05	0.05
Link 2	5	0.2	0.05	0.05	0.05
Link 3	10	0.75	0.1	0.1	0.1
Link 4	10	0.75	0.1	0.1	0.1

The mass ratio of the robotic manipulator to base spacecraft is roughly 30%, which is high above average for commonly known space robotic missions [5]. This is done to demonstrate the effectiveness of the proposed RL control scheme in the presence of strong dynamic coupling between the robotic manipulator and the base spacecraft. The parameters for the training process and Actor-Critic networks are shown in Table 5.2.

Table 5.2 Reinforcement Learning Parameters

Parameter	Value
Learning rate	0.001

Replay Buffer	50,000
Discount factor for future reward	0.99
Smooth factor	0.001
Mini-batch size	128
Number of Episodes	3000
Episode timesteps	400
Timestep size	0.025 s
Episode time	10 s
Neural network architecture for Actor and Critic	Hidden layers (400, 300)
Lower & upper bounds for joint torques	-3 Nm, 3 Nm

5.2.1 Case 1: 3DOF Fixed Base Manipulator

The first simulation was done on a fixed-base 3 DOF robotic manipulator, with the task of reaching its target for capture. The reward function at each timestep is given as:

$$R_t = -d \tag{5.1}$$

This ensures that the agent will receive higher reward values, if the distance tracking error (d in Eq. (4.38)) is smaller; the robotic arm's EE moves closer towards the target. The result of training is shown in Figure 5.3. The blue line is the cumulative reward from all timesteps in each episode, and the

red line is the moving average of the cumulative rewards of all episodes.

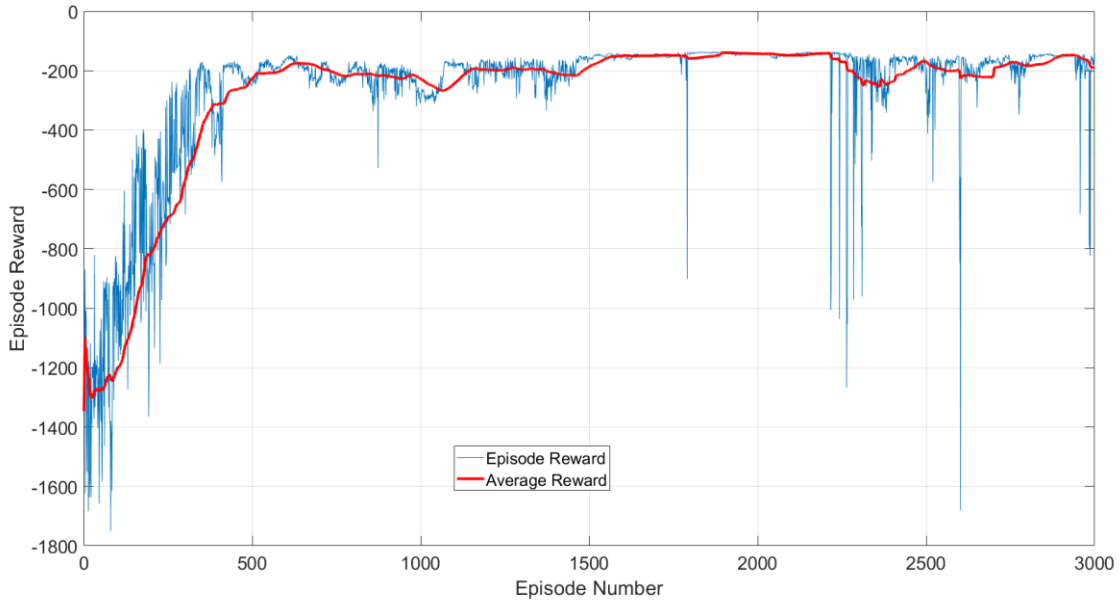


Figure 5.3 Case 1 Training result :3,000 episodes

We can see that the training was a success. Notably, the robotic manipulator was able to reach its target and achieve high rewards after about 500 episodes of training and spends the rest of the episodes exploring different paths to search for higher rewards. Figure 5.4 shows the snap shots of robotic manipulator capture mission after the agent is trained and implemented.

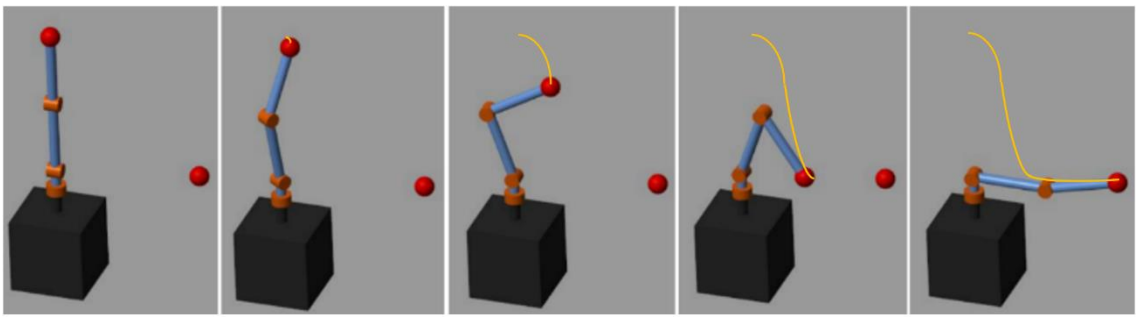


Figure 5.4 Motion of implemented trained agent from Case 1

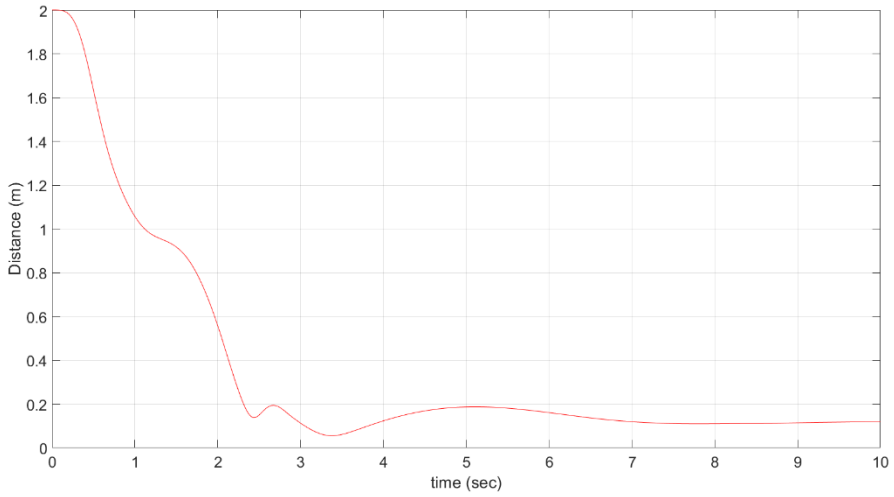
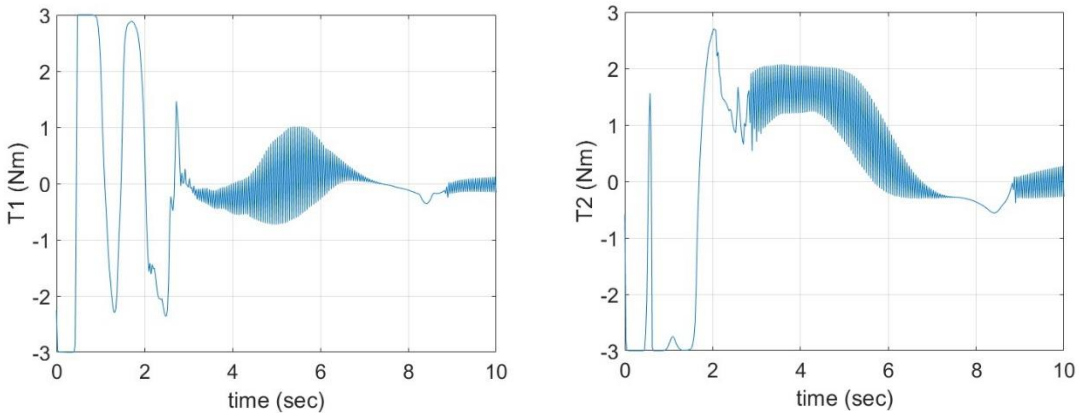


Figure 5.5 Case 1 position tracking error (d) over time

Figure 5.5 shows the tracking error in position between the EE and target over time. Evidently, the trained agent steers the end-effector towards target position (defined capture range) within 3s, and subsequently, the end-effector stays within that region (to achieve capture) for the remaining duration. Figure 5.6 shows the time histories of actions or output torques at each joint given by the agent.



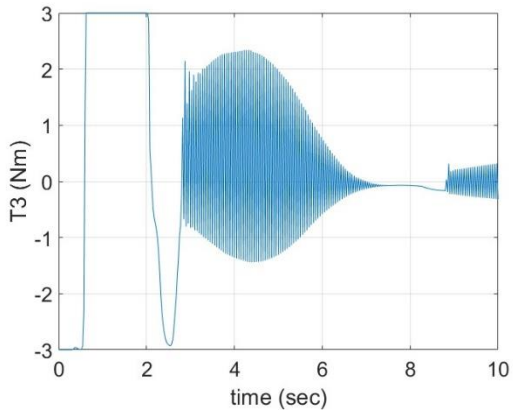


Figure 5.6 Case 1 time histories of joint torques given by the agent

Notably, the agent output (torque) has large oscillations. This phenomenon, referred to as action oscillation in [98], arises from the balance between exploration and exploitation during training. Deep reinforcement learning algorithms are particularly prone to this issue in discrete action settings, where an agent may select different actions in consecutive steps despite only slight differences in states. The problem is worsened by the presence of noise in the state observations. Although this oscillation is undesirable, the agent perceives it as the optimal strategy to maximize rewards, given that its decisions are driven by a reward only. Future work is needed to mitigate this phenomenon by incorporating a penalty for excessive changes in actions or torque.

5.2.2 Case 2: 3DOF Free-Floating Base Manipulator

The second simulation is done on a free-floating 3 DOF space robotic

manipulator, where is base is free to move in reaction to the arm motion. The task is to reach the target position and stay in that position as long as possible for later capture. The reward function at each timestep is given as:

$$R_t = -K_d d + r_d \tag{5.2}$$

where $K_d = 30$ is a weighted constant and $r_1 = 0.2$ is a large sparse reward given at each time step that the distance between EE and target $d_{capture} \leq 0.1m$. This will ensure that after the EE reaches the target, it will stay within the specified range of the target for future grasping and/or servicing. The result of training is shown in Figure 5.7.

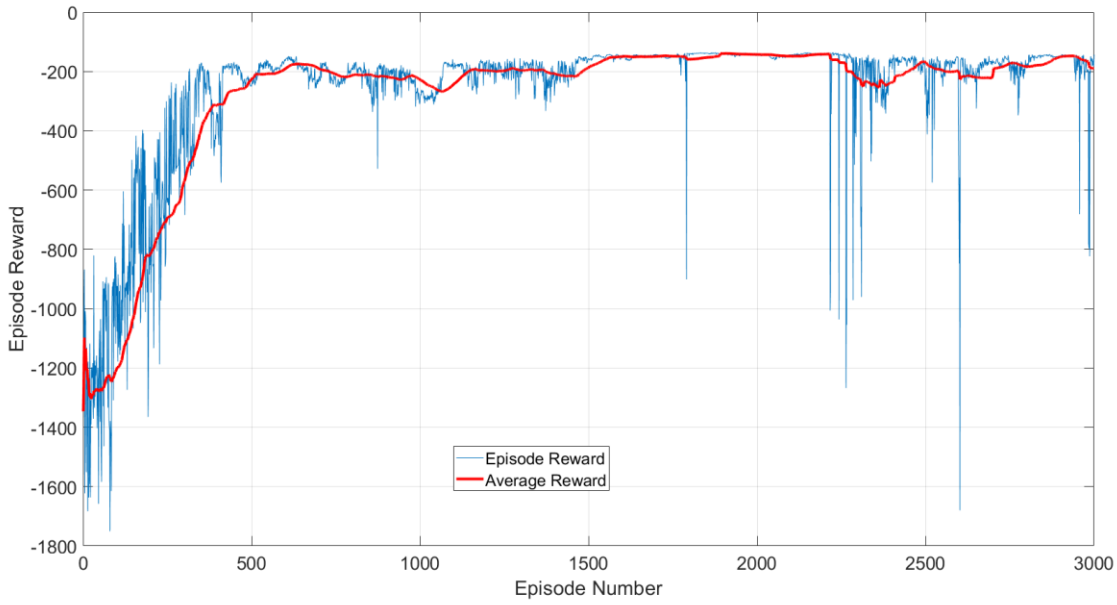


Figure 5.7 Case 2 Training result :3,000 episodes

We notice that the robotic manipulator's EE was able to get close to its target within the first 500 episodes of training, and after 1500 episodes started

to get the high reward (r_d) for staying within 0.2m of the target until the end of each episode.

What is interesting is that the Agent learned to reach the target faster to obtain the high reward (r_d) sooner and keep receiving it until the episode ends. That caused the robotic manipulator to collide with itself to do so. We can see this in Figure 5.8. That shows snap shots of robotic manipulator capture mission after the agent is trained.

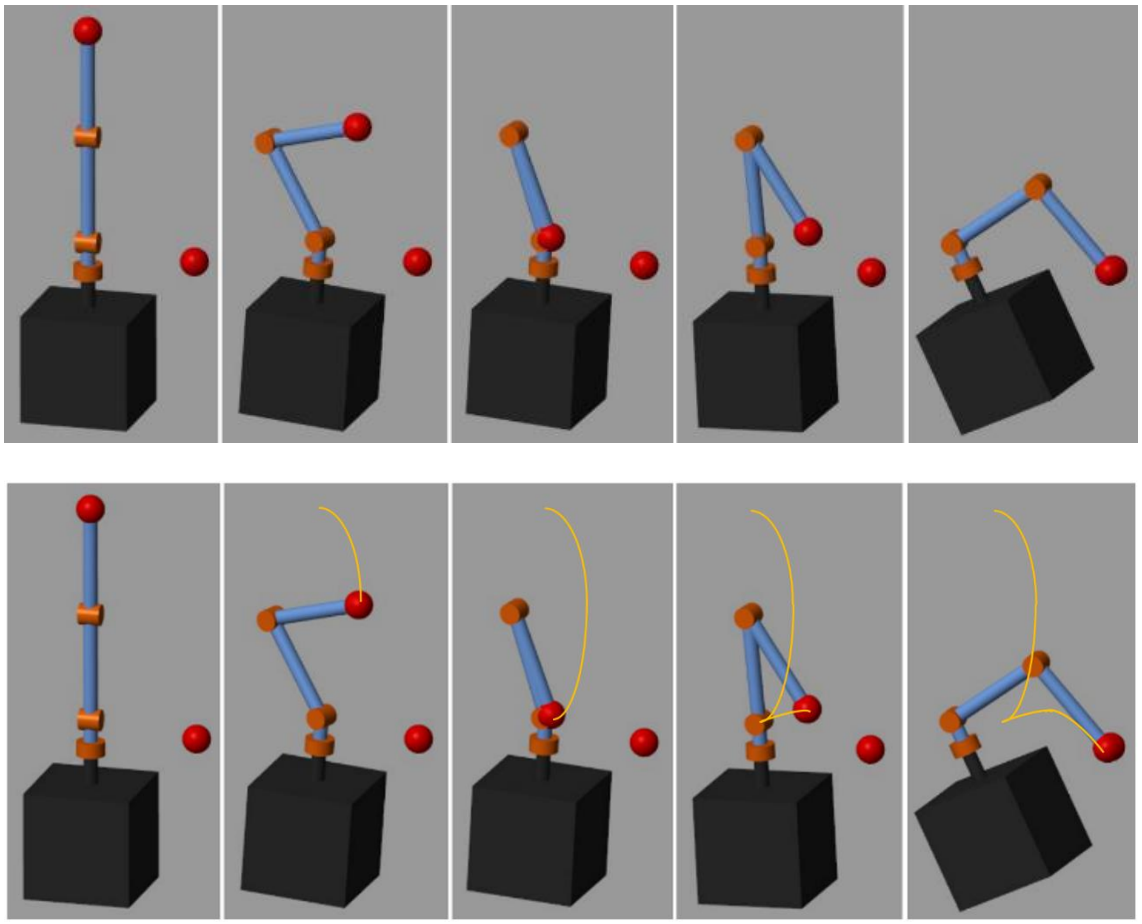


Figure 5.8 Motion of implemented trained agent from Case 2

Figure 5.9 shows the tracking error in position between the EE and target over time. In this case, the trained agent steers the end-effector towards the target position within 4s and remains within the sparse reward region.

Figure 5.10 shows the time histories of actions or output torques at each joint given by the agent.

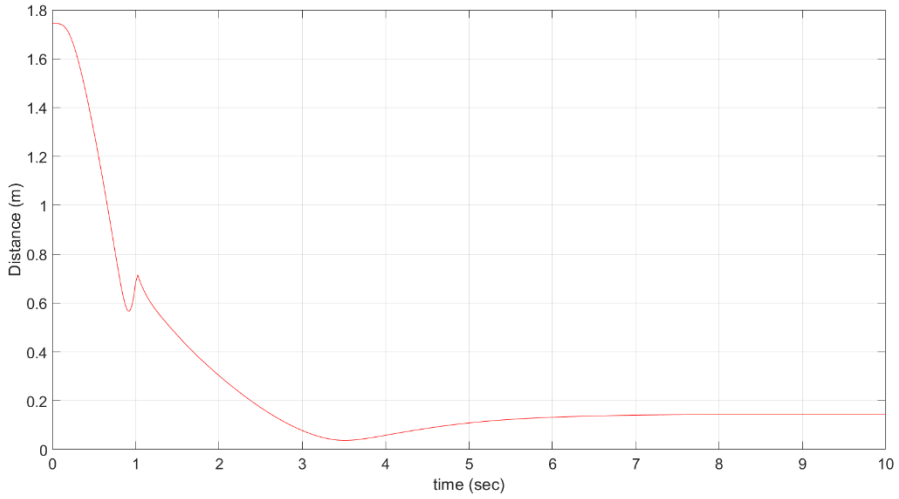
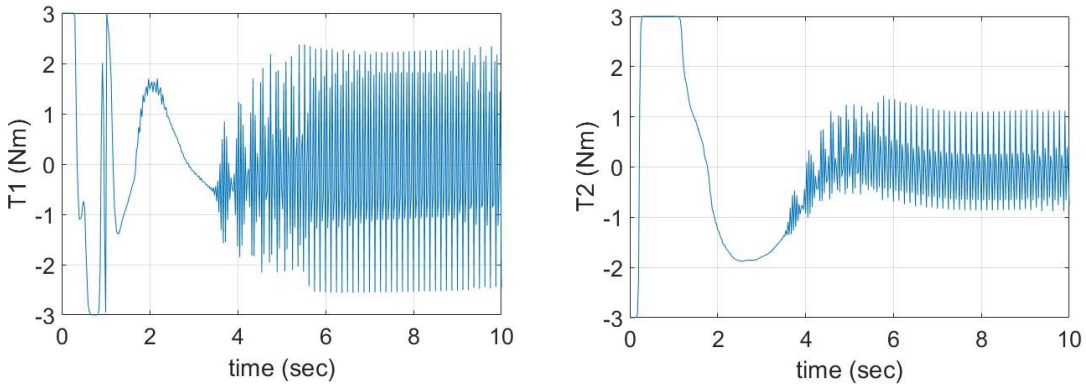


Figure 5.9 Case 2 position tracking error (d) over time



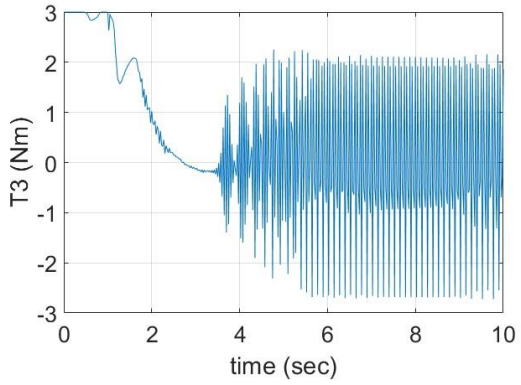


Figure 5.10 Case 2 time histories of joint torques given by the agent

5.2.3 Case 3: 3DOF Free-Floating Base Manipulator with Improved Reward

The third simulation is done on a free-floating 3 DOF space robotic manipulator, with the task of achieving target capture, staying within capture distance, while avoiding self-collision, and optimizing the path. The reward function at each timestep is given as:

$$R_t = -K_d\left(\Delta d\right) + r_d - p_l \tag{5.3}$$

where $\Delta d = d_t - d_{t-1}$ is the difference between current distance (d_t) and previous distance (d_{t-1}). This means that the agent will obtain a negative weighted reward if the robotic EE moves away from the target ($d_t > d_{t-1}$), and the agent will obtain a positive weighted reward if the robotic EE moved closer towards the target ($d_t < d_{t-1}$). Implying that the path is taken into consideration and improves, as opposed to trying to reach the target as fast as possible without

considering the path (as seen in case 2). This ensures the agent will learn to move only towards the target. The result of training is shown in Figure 5.11. What is more impressive is that with this improved reward function, we only needed 2000 episodes to fully train the agent.

Figure 5.12 shows the snap shots of robotic manipulator capture mission after the agent is trained. The path of the space robotic manipulator improved significantly, it reached its target position, smoothly and without any self-collision. Furthermore, it stayed at that target for future capture.

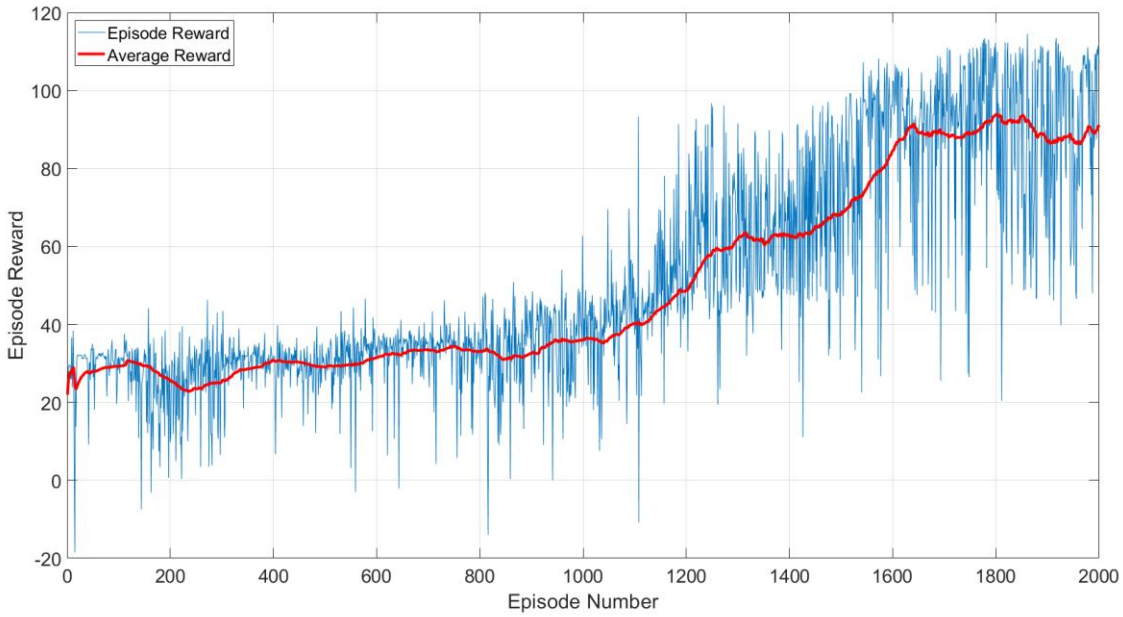


Figure 5.11 Case 3 Training result :2,000 episodes

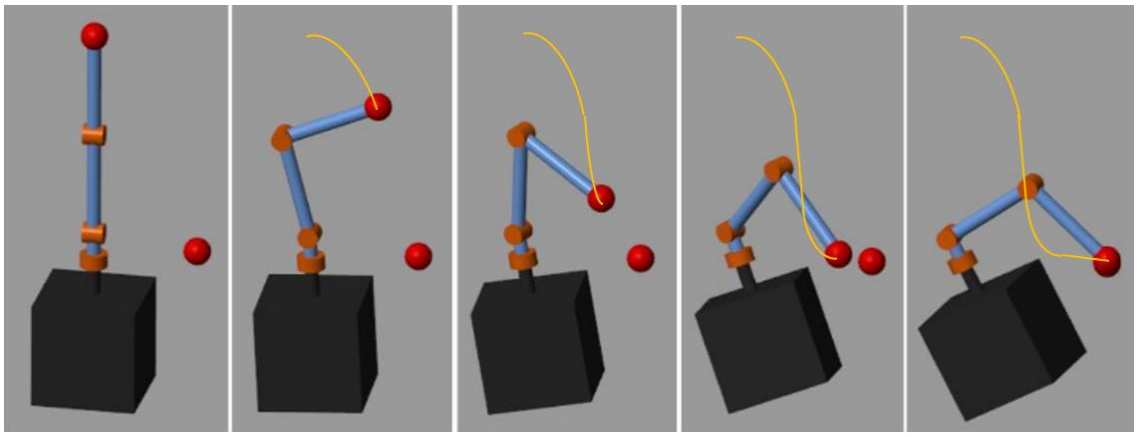


Figure 5.12 Motion of implemented trained agent from Case 3

Figure 5.13 shows the tracking error in position between the EE and target over time. Evidently, the trained agent steers the end-effector towards target position (defined capture range) within 2s, afterwards the end-effector stays within that region (to achieve capture) for the remaining duration.

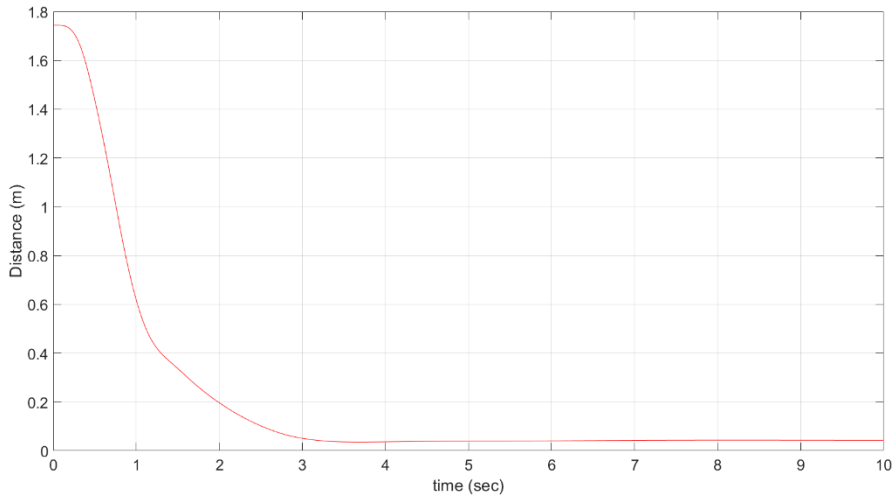


Figure 5.13 Case 3 position tracking error (d) over time

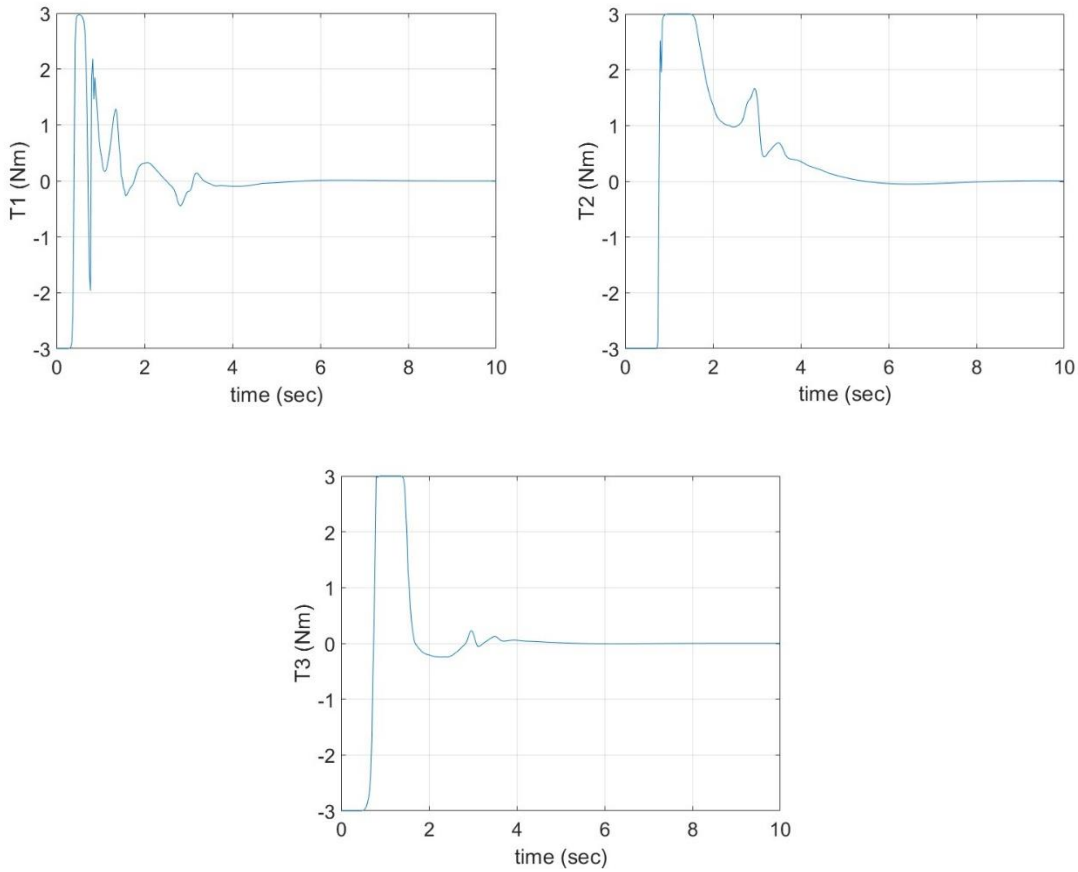


Figure 5.14 Case 3 time histories of joint torques given by the agent

Figure 5.14 shows the time histories of actions or output torques at each joint given by the agent. It is noticed that the torques are smoother and their values have less oscillation in comparison with the results from Figure 5.10.

5.3 Six DOF Space Robotic Manipulator

In this section, the numerical simulation is done for a 6DOF free-floating space robotic manipulator to verify the effectiveness of the proposed RL control

scheme. The manipulator is seen in Figure 5.15.

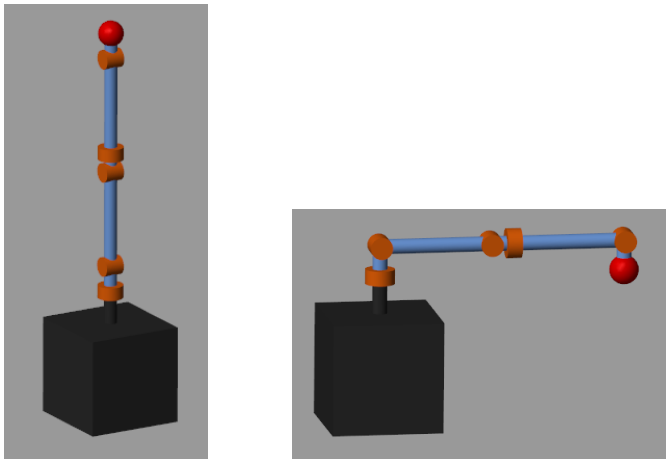


Figure 5.15 6 DOF Space robotic Manipulator system

The physical parameters of the space robotic manipulator are listed in Table 5.3 [82]. The initial pose of the end-effector is $(x, y, z, \xi)_e$ = (0, 0, 2.675, -1, 0, 0, 0) and the target's pose is $(x, y, z, \xi)_T$ = (1, 0, 1, -0.707, 0, -0.707, 0). The torque limits for the first three joints are set at ±6 Nm, while the remaining three joints have a constraint of ±3 Nm. The study considers two scenarios regarding the manipulator-base spacecraft mass ratio. The first scenario has a manipulator-base mass ratio of roughly at 7.4%, which is typical in known space robotic missions, along with the different cases of different reward function, and environmental factors. The second scenario has a manipulator-base mass ratio of 30% to assess the proposed reward function's performance in the presence of strong dynamic coupling between the robotic manipulator and the base spacecraft, which is prone to dynamic singularities. The

parameters used in the reward function are defined as $k_d = 0.1$, $k_\vartheta = 0.001$, $d_0 = 0.15m$, $\vartheta_0 = 15^\circ$, $r_d = 1$, $r_\vartheta = 2$, $p_j = 0.005$, $p_l = 0.5$, and $d_l = 0.3m$.

Table 5.3 Parameters of Six DOF Space Robotic Manipulator

Body	Mass (kg)	Length (m)	*Ix* (kg/m²)	*Iy* (kg/m²)	*Iz* (kg/m²)
Base	1700	0.75x0.75 x0.75	1434	1434	1434
Link 1	5	0.25	0.0292	0.0292	0.0063
Link 2	5	0.2	0.0292	0.0292	0.0063
Link 3	50	0.75	0.0625	26.1	26.1
Link 4	10	0.2	0.0125	0.215	0.215
Link 5	50	0.75	0.0625	26.1	26.1
Link 6	5	0.2	0.0063	0.0292	0.0292

The mass ratio of the robotic manipulator to base spacecraft is roughly 30%, which is high above average for commonly known space robotic missions [5]. This is done to demonstrate the effectiveness of the proposed RL control scheme in the presence of strong dynamic coupling between the robotic manipulator and the base spacecraft. The parameters for the training process and Actor-Critic networks are shown in Table 5.4.

Table 5.4 Reinforcement Learning Parameters

Parameter	Value

Learning rate	0.001
Replay Buffer	50,000
Discount factor for future reward	0.99
Smooth factor	0.001
Mini-batch size	128
Number of Episodes	3000
Episode timesteps	400
Timestep size	0.025 s
Episode time	10 s
Neural network architecture for Actor and Critic	Hidden layers (400, 300)
Lower & upper bounds for joint torques (1-3)	-3 Nm, 3 Nm
Lower & upper bounds for joint torques (4-6)	-6 Nm, 6 Nm

5.3.1 Case 1: Approach Target without Velocity Constraint

In this case, the free-floating space robotic manipulator has been trained using the reward function in Equation(4.45). The agent is trained to guide the end-effector towards the desired pose $(x, y, z, \xi)_T$ with a typical manipulator-base spacecraft mass ratio of 7.4% with and without imposing any constraints on velocity. The training process, derived from 5,000 episodes, is shown in Figure 5.16.

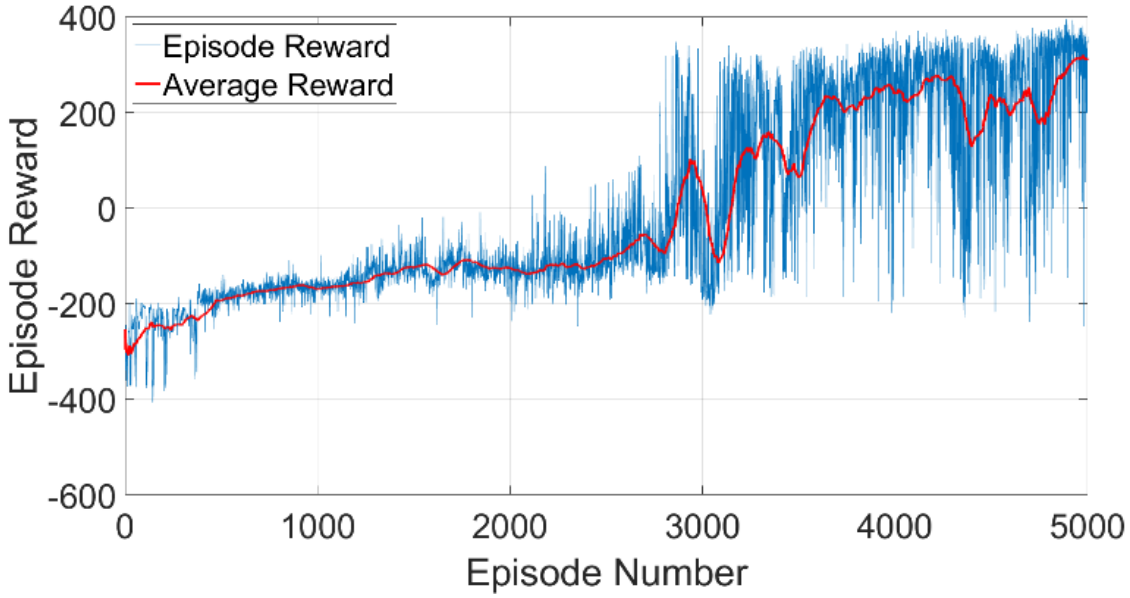


Figure 5.16 Case 1 Training result :5,000 episodes

Initially, the agent starts the learning process by searching for the highest reward, primarily focusing on the term $(-K_d d - K_\vartheta \vartheta)$, which allows the end-effector to rapidly approach the desired target pose. Around the halfway mark, after approximately 2,500 episodes, the end-effector starts to consistently reach the target's pose. Once the end-effector approaching to the target's pose with errors less than $d_0 = 0.15m$ and $\vartheta_0 = 15^\circ$, the agent relies on the activation of large rewards (r_d, r_ϑ) to get the end-effector closer to the target's pose and subsequently maintain in that pose throughout the remaining timesteps by concentrating on maximizing rewards within this specific region, thereby limiting its search space. The training process eventually approaches after 3,700 episodes, indicating the agent's ability to

effectively learn and perform the desired task. Figure 5.17 shows snap shots of robotic manipulator in the approaching process. It shows the base spacecraft moves translationally and rotationally because of the small mass ratio between the robotic manipulator and the base spacecraft.

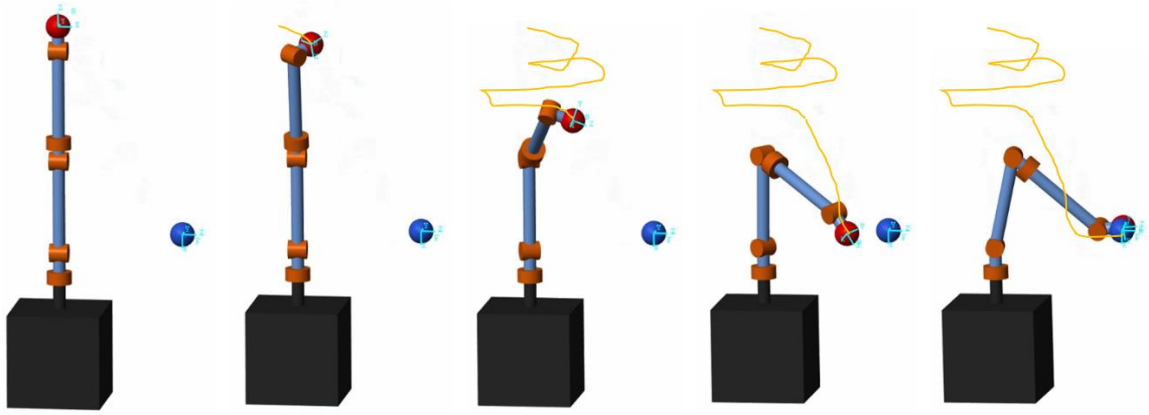


Figure 5.17 Motion of implemented trained agent from Case 1

The position (d) and orientation (ϑ) tracking errors between the end-effector and the target over time are shown in Figure 5.18. Evidently, the trained agent steers the end-effector toward the target's pose within the defined tolerance range in 6s, and subsequently, the end-effector stays within that region for the remaining duration. It is worth mentioning that the threshold $d_0 = 0.15m$ does not imply that the minimum distance error the trained agent can reach; instead, it signifies that crossing this threshold secures a large reward r_d .

In fact, the agent achieves a minimum distance error of 0.05m. The

threshold helps the agent to swiftly approach this region at an earlier stage, then spend the rest of the time steps searching for more rewards due to its greedy nature, effectively refining the search space for optimal solutions. Similarly, the same approach applies to the orientation error, achieving a minimum orientation difference of 5.0 degrees. Figure 5.19 shows the time histories of actions or output torques at each joint given by the agent.

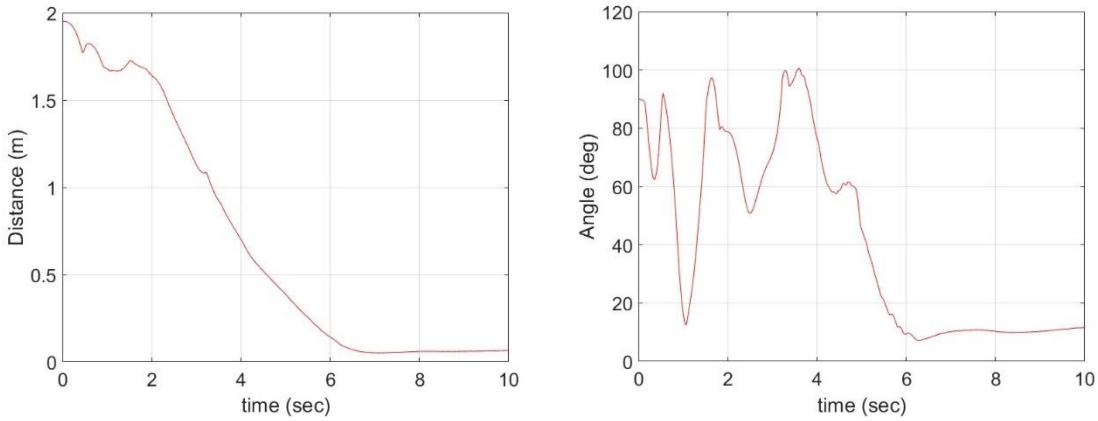
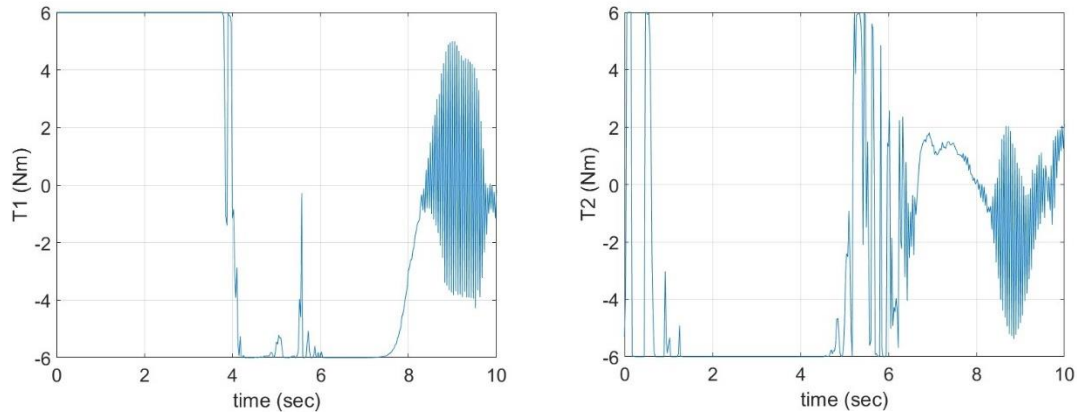


Figure 5.18 Case 1 Pose tracking errors (d, ϑ) over time



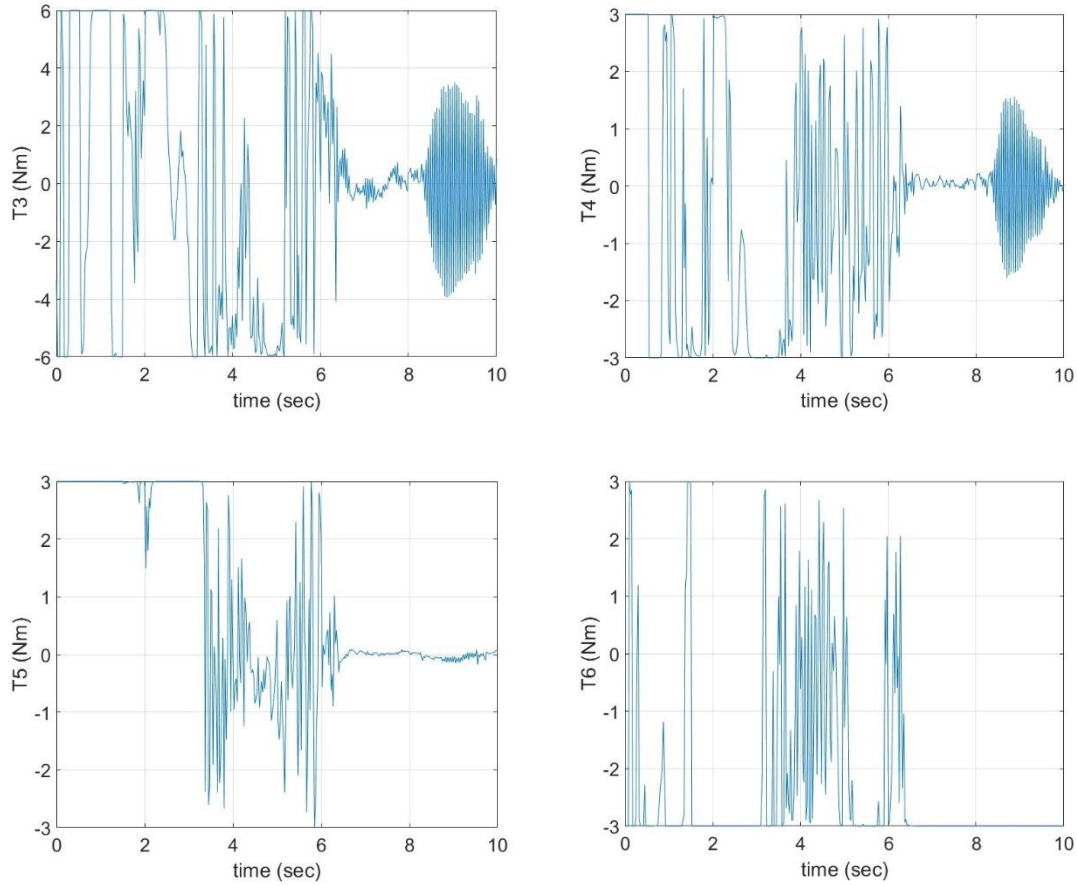


Figure 5.19 Case 1 time histories of joint torques given by the agent

Shows the squared sum of all joint angular velocities ($\sum(\dot{\varphi})^2$). Notably, it has a high peak initially, and still has high values throughout the motion.

5.3.2 Case 2: Approach Target with Velocity Constraint

To improve the path and avoid the undesired excess robotic manipulator joint velocities. The free-floating space robotic manipulator has been trained

with a penalty on joint angular velocities ($-p_j \sum(\dot{\phi})^2$), as shown:

$$R_t = -K_d d - K_\vartheta \vartheta + r_d + r_\vartheta - p_l - p_j \sum(\dot{\phi})^2 \tag{5.4}$$

The coefficient of the penalty term is $p_j = 0.005$, which determined by trial-and-error. The training result is shown in Figure 5.20.

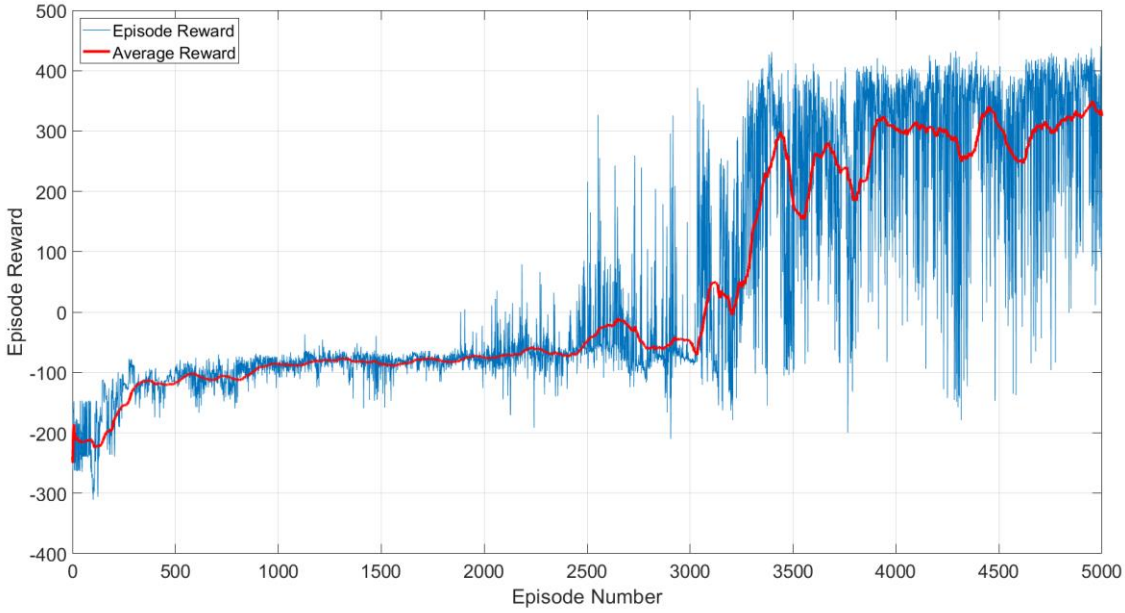


Figure 5.20 Case 2 Training result :5,000 episodes

Figure 5.21 shows the snap shots of robotic manipulator in the approaching process. Next, the tracking errors in position d and orientation ϑ between the end-effector and the target are shown in Figure 5.22. Notably, the errors progress into the specified tolerance range within the first 7s.

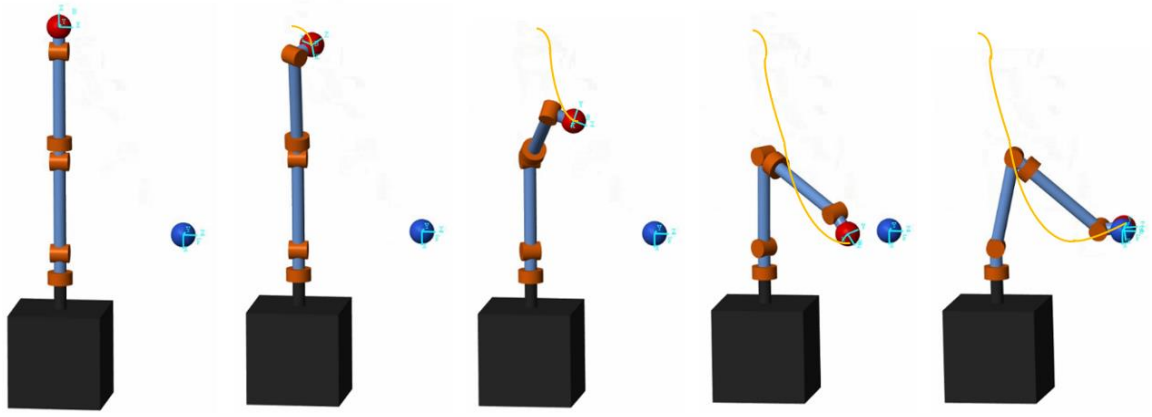


Figure 5.21 Motion of implemented trained agent from Case 2

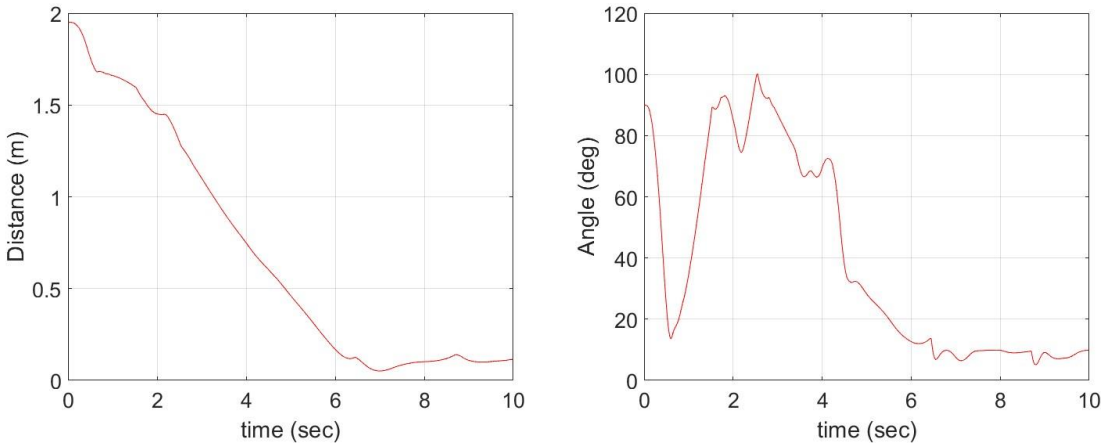


Figure 5.22 Case 2 Pose tracking errors (d, ϑ) over time

The impact of implementing a velocity constraint in the training process is shown Figure 5.24 by comparing the squared sums of all joint velocities ($\sum(\dot{\varphi})^2$) for scenarios with and without velocity constraint. The motion executed by the agent trained without any velocity constraints is represented by the red line, while the motion from the agent trained with the velocity constraint incorporated into the reward function is depicted by the blue line. It

is clearly shown that the velocity constraint added in the reward function had reduced the squared sum of all joint velocities by over three times. This evident decrease confirms that incorporating the velocity constraint into the reward function successfully smooths the robotic manipulator's motion.

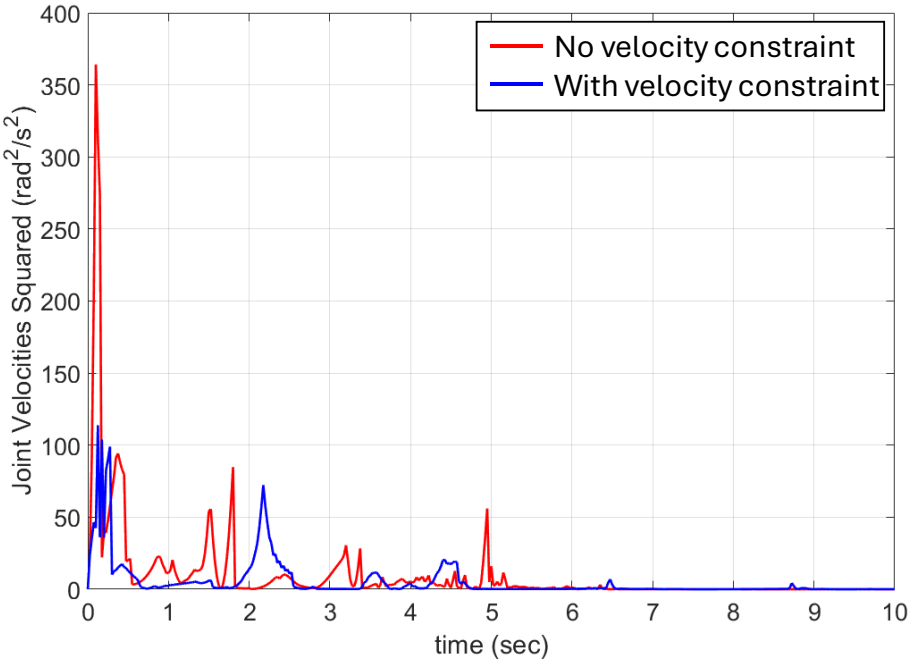


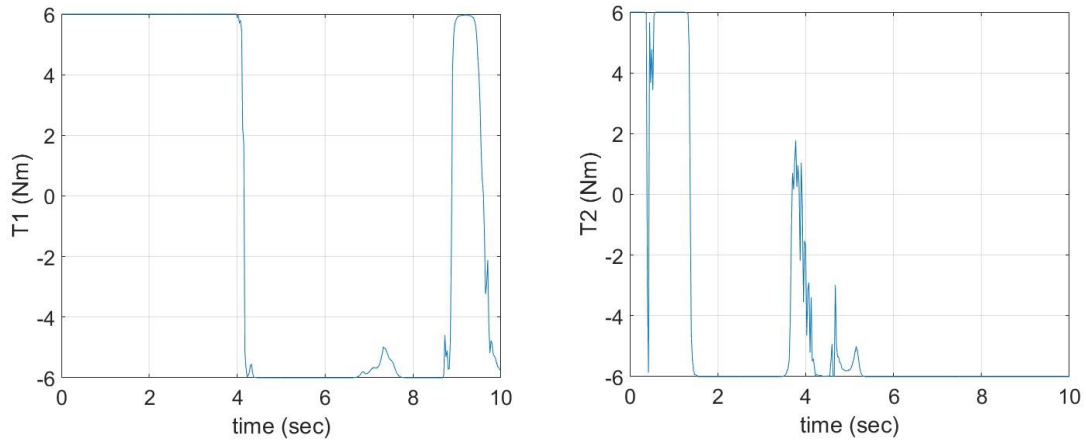
Figure 5.23 Squared sums of all joint velocities with and without velocity constraint

The ability to refine the motion of the agent through a customized reward function paves the way for further investigations into reward function design. Future studies could explore the inclusion of various criteria within the reward function that have been previously utilized in optimization efforts, such

as the minimization of torques, exerted forces, energy consumption, and disturbances to the base, among others.

Figure 5.24 shows the output torques given by the agent for all six joints for the second case, where the velocity penalty was used. It is noticed that the torques are smoother and their values are smaller compared to the results in Figure 5.19, without velocity penalty, meaning the velocity penalty works well in reducing joint velocities and the control torques.

This comparison shows clearly that the torques from the agent trained with the velocity constraint exhibit smoother characteristics and fewer oscillations in comparison with the torques generated by the agent without any velocity constraints. Additionally, the agent with the velocity constraint achieves stable torques at the end of training, indicating a more controlled and consistent performance throughout the operation.



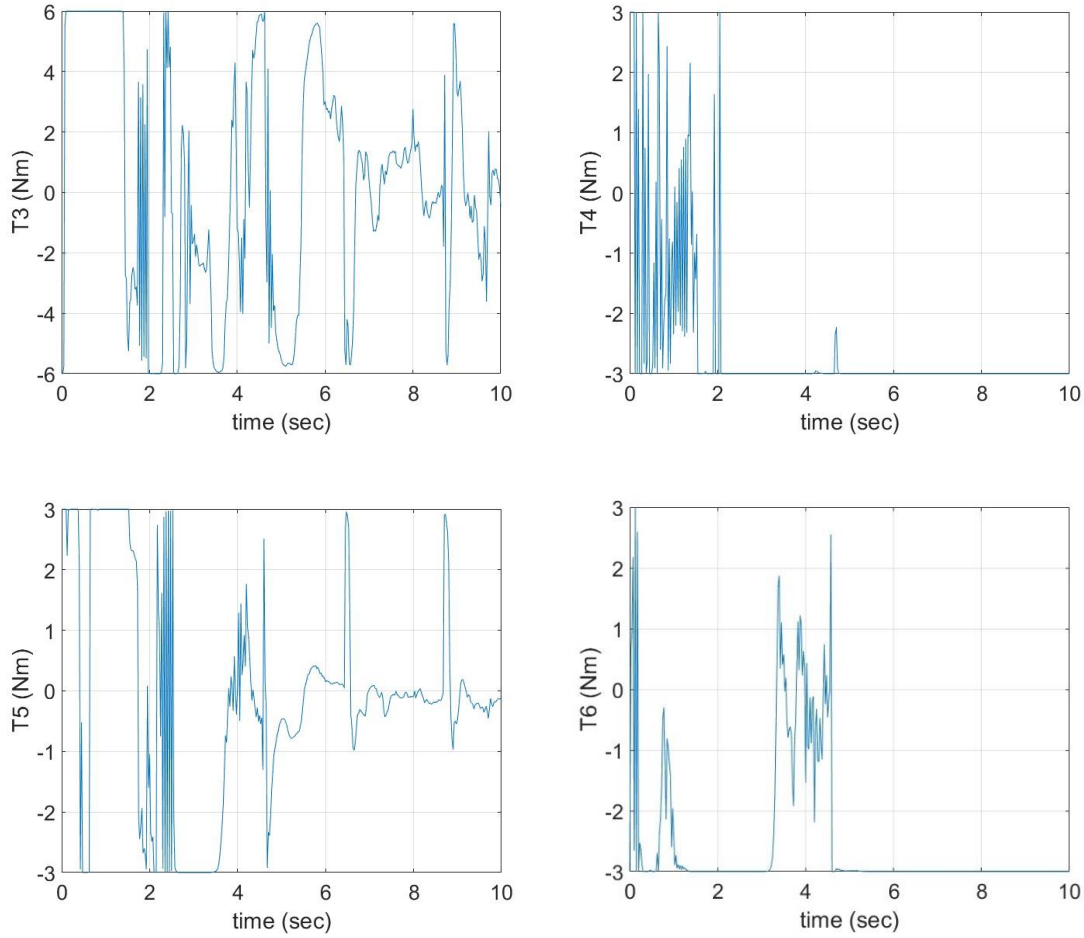


Figure 5.24 Case 2 time histories of joint torques given by the agent

5.3.3 Case 3: Approach Target with Larger Mass Ratio

In this particular case, the free-floating space robotic manipulator mounted on a smaller base spacecraft has been trained using the same reward function in Equation(5.4), which was previously used in Case 2. However, unlike the previous cases where the mass ratio between the robotic

manipulator and the base spacecraft was 7.4%, the mass ratio is increased to 30% in the current case. Consequently, it is expected that the dynamic coupling effect between the base spacecraft and the robotic manipulator will be intensified. Adjustments are made to the base mass and inertia values listed in Table 5.3, with the base mass becoming 425kg and the inertia components (*Ix, Iy, Iz*) revised to (358, 358, 434) kg·m^2, respectively. Aside from the altered mass ratio, the geometric parameters of the manipulator and other RL parameters remain unchanged, including the incorporation of the velocity constraint in the reward function.

The training results, as shown in Figure 5.25, clearly demonstrate that increasing the dynamic coupling presents additional challenges and prolongs the approach time towards the desired pose. For instance, it requires approximately 3,700 episodes to begin reaching the defined tolerance region, compared to the 3,200 episodes observed in Figure 5.20.

Despite the increased complexity introduced by the heightened dynamic coupling, the agent was able to reach the desired pose, highlighting the effectiveness of the proposed reward function to overcome the increased dynamic coupling effects.

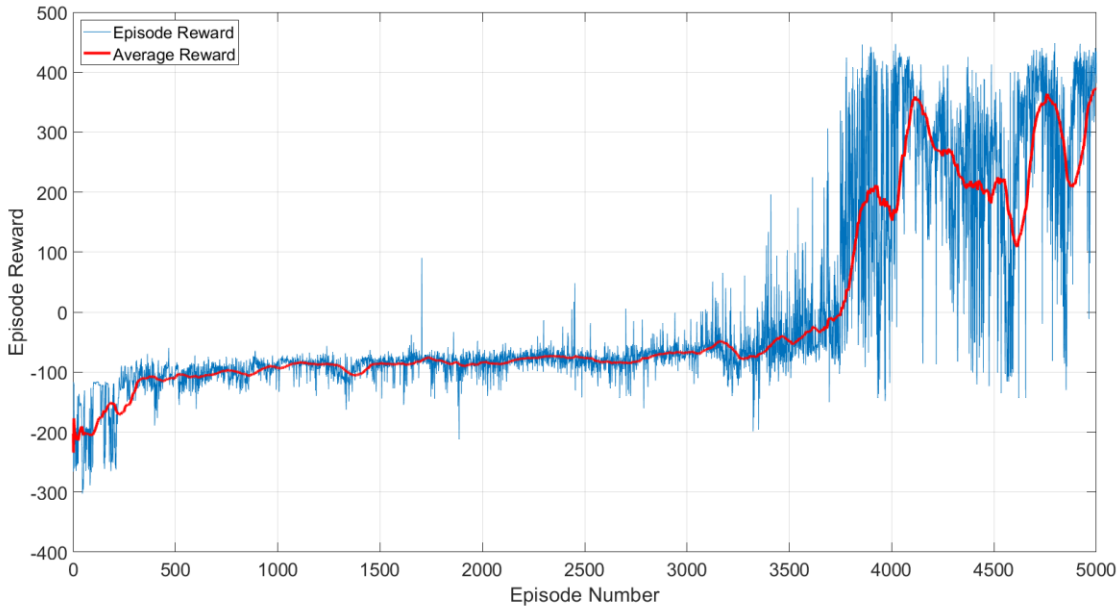


Figure 5.25 Case 3 Training result :5,000 episodes

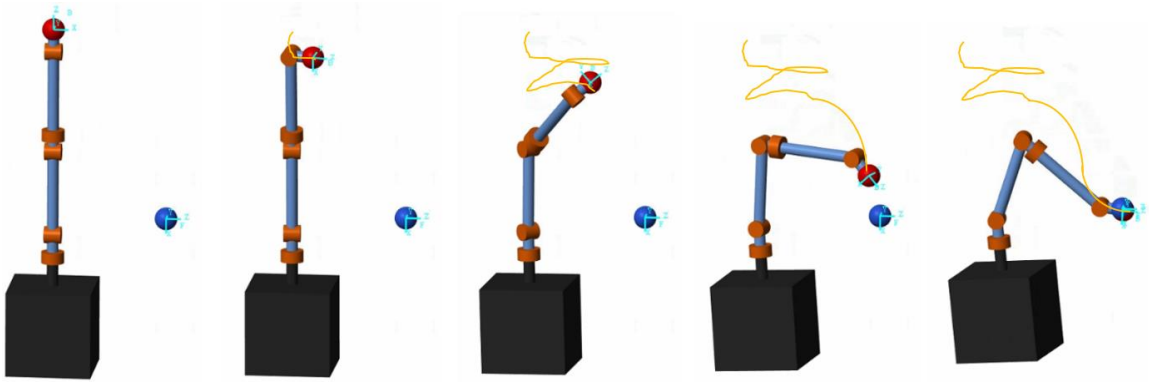


Figure 5.26 Motion of implemented trained agent from Case 3

Figure 5.26 presents snapshots of the robotic manipulator as it approaches its target. These images clearly show that the base spacecraft experiences significant disturbances in the process as a result of the dynamic coupling between the manipulator and base spacecraft due to the high mass

ratio. To further illustrate the effects of dynamic coupling, the translational and rotational movements /disturbances of the base spacecraft observed during the approaching process are shown in Figure 5.27. For a comparative analysis, the movements of the base spacecraft from the previous case with a smaller mass ratio of 7.4% are also plotted in Figure 5.27, where red lines represent the small mass ratio case, and blue lines denote the high mass ratio case.

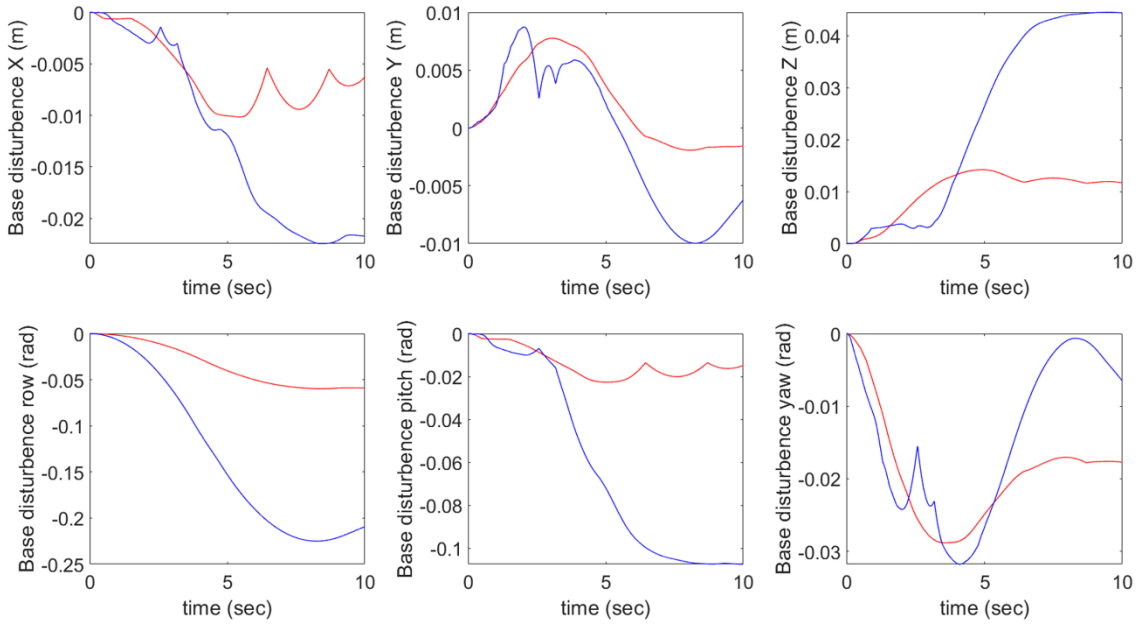


Figure 5.27 Position and attitude disturbances of base spacecraft: —small mass ratio, —high mass ratio

Figure 5.28 represents the tracking errors in position d and orientation ϑ between the end-effector and the target over time. Evidently these errors progress towards the specified range with similar times as the previous cases.

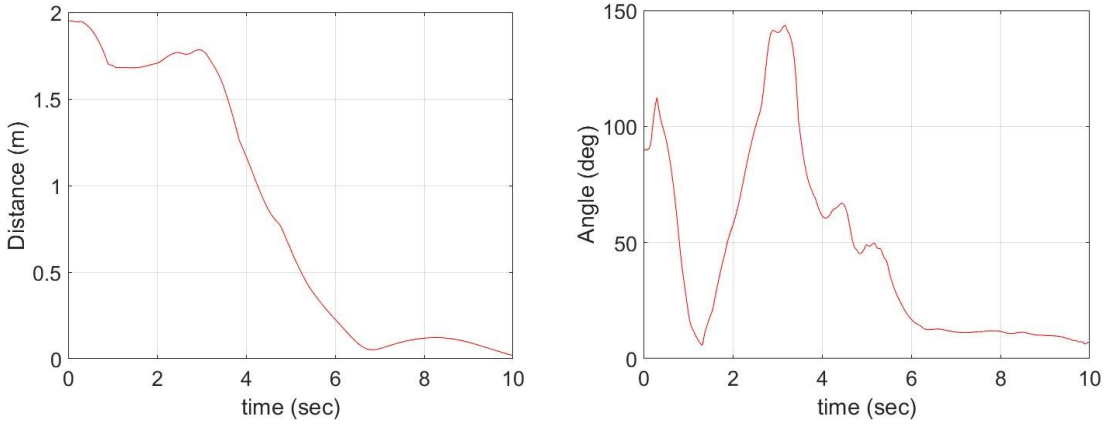


Figure 5.28 Case 3 Pose tracking errors (d, ϑ) over time

Figure 5.29 shows the time histories of output torques generated by the agent for all six joints. In comparison to Case 2, it is apparent that the torques are higher, which can be attributed to the significant dynamic coupling between the robotic manipulator and the base spacecraft. This indicates that the strong dynamic coupling necessitates greater torque inputs to ensure the end-effector achieves the desired pose. The achieved results clearly demonstrate the effectiveness of Reinforcement Learning for path planning in free-floating space robots even when applied to a system with a reduced mass base spacecraft, resulting in larger manipulator/base coupling. It implies that future space robotic missions will no longer need to design the base spacecraft with larger masses solely to minimize the dynamic coupling effect. The ability to utilize smaller base spacecraft without concerns about dynamic coupling revolutionizes the design and operation of space robotic systems. It facilitates increased efficiency, flexibility, and cost-effectiveness in space missions.

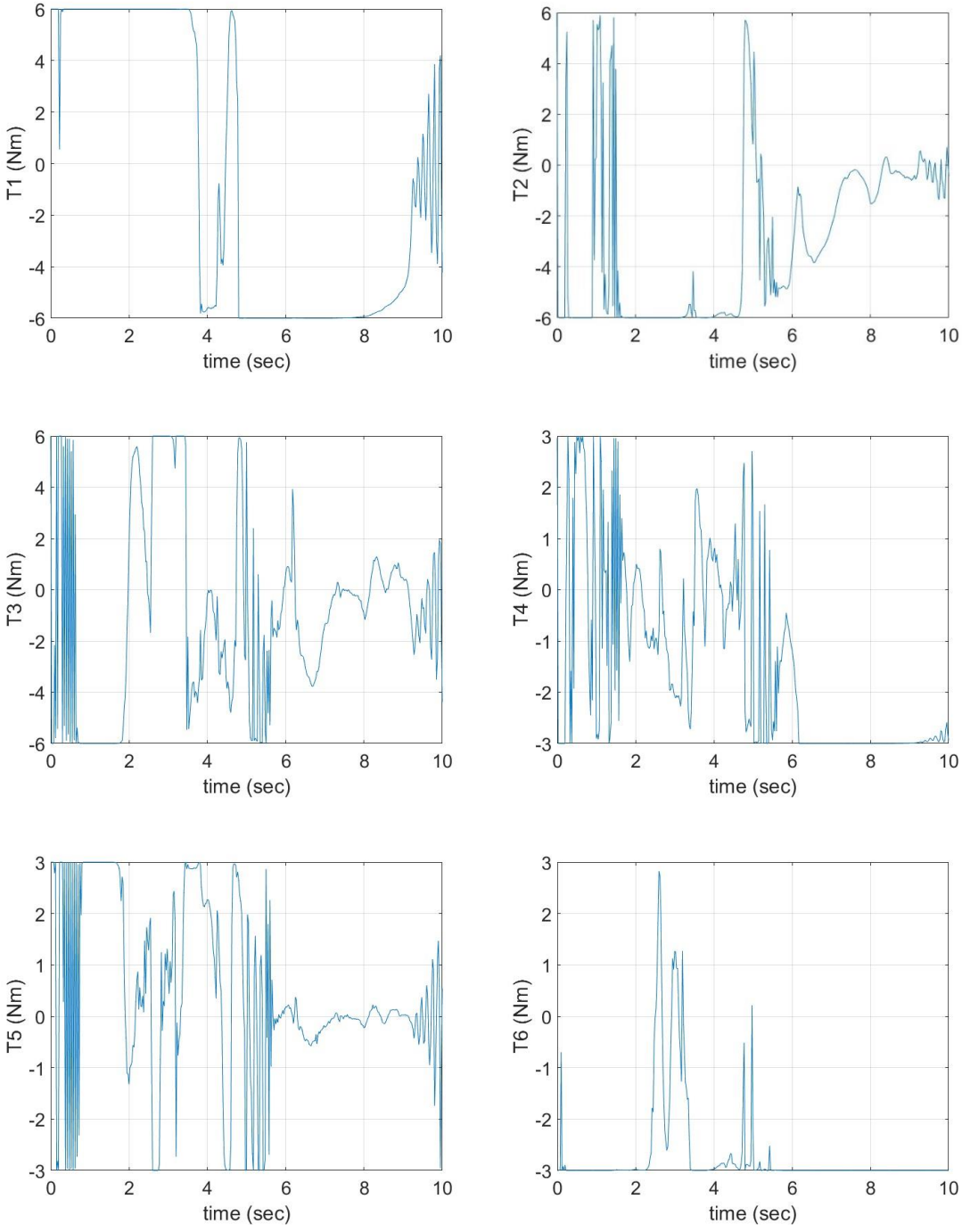


Figure 5.29 Case 3 time histories of joint torques given by the agent

5.3.4 Case 4: Approach Target with Noisy Observations

In this case, the influence of state measurement noise on RL is studied by introducing white noises into state observations at a signal-to-error ratio of 10/1. It is known that in real-world scenarios, perfect observations of the state are unattainable, particularly when estimating the target's pose using non-intrusive cameras in space.

To assess the robustness of the proposed methodology in handling noisy observations, in this case the agent has been trained while introducing a white noise to the target position, orientation, and errors $(\mathbf{p}_T, \xi_T, d, \vartheta)$ such that:

$$(\mathbf{p}_T, \xi_T, d, \vartheta)_{out} = (1 + R_n)(\mathbf{p}_T, \xi_T, d, \vartheta)_{in} \quad R_n \in \mathbb{N}(0,1) \tag{5.5}$$

where $(\mathbf{p}_T, \xi_T, d, \vartheta)_{in}$ is the ideal target pose and error, $(\mathbf{p}_T, \xi_T, d, \vartheta)_{out}$ is the noisy pose and error, and R_n is the signal-to-noise ratio (SNR), which is assumed 10/1 in the current study.

The mass ratio between the robotic manipulator and the base spacecraft is set to 7.4%. The same parameters of the manipulator and RL remains the same as those in Case 2 with the incorporation of joint velocity constraint in the reward functions. Figure 5.30 shows the result of training. It is noteworthy that the training process requires more episodes to approach, specifically 7,000 as compared to 5,000 in previous cases. Nevertheless, the agent effectively overcomes the challenges posed by noisy inputs, which demonstrates the

effectiveness and robustness of the proposed reward function. Figure 5.31 presents snapshots of the robotic manipulator as it approaches its target.

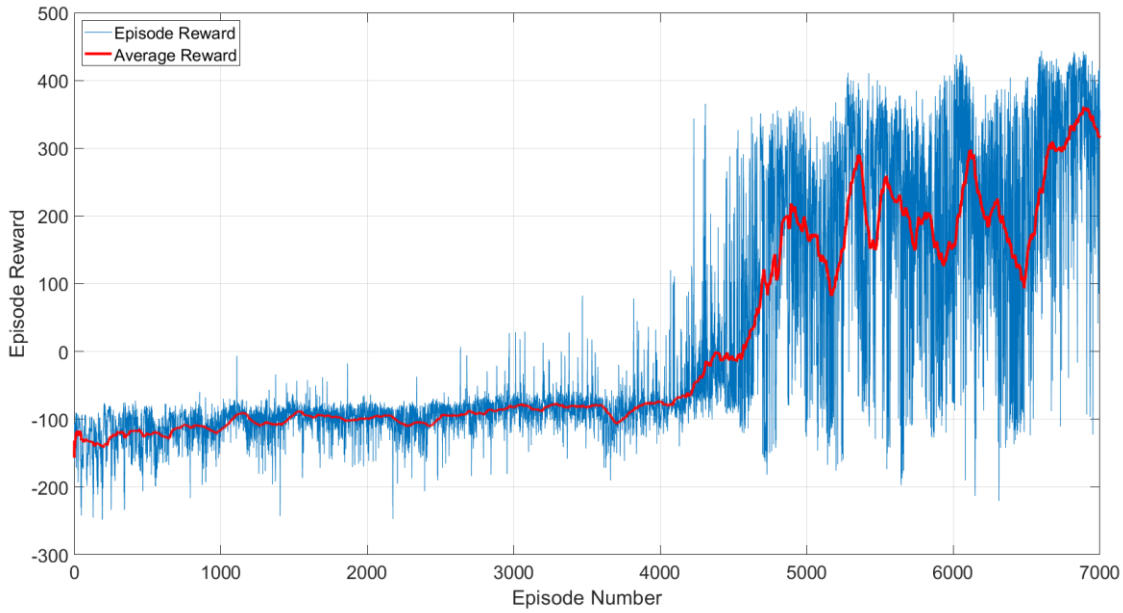


Figure 5.30 Case 4 Training result :7,000 episodes

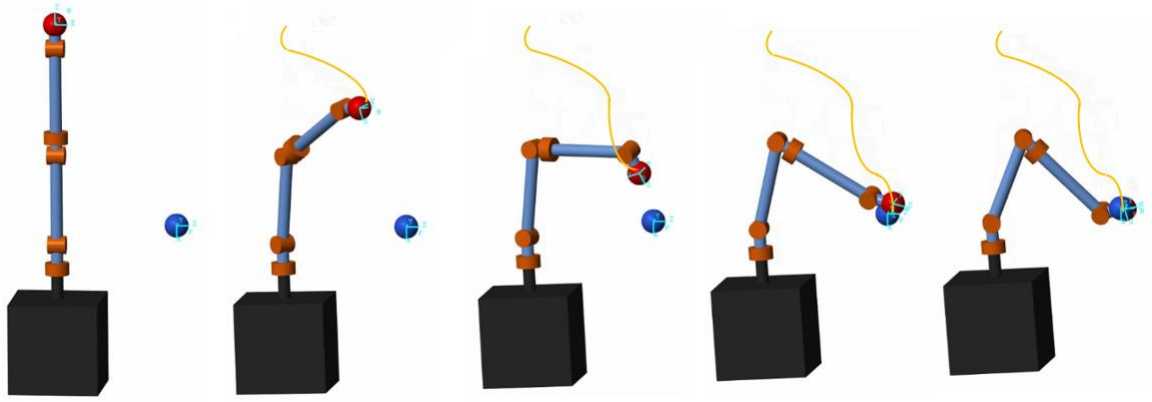


Figure 5.31 Motion of implemented trained agent from Case 4

The position (d) and orientation (ϑ) tracking errors between the end-effector and the target over time are shown in Figure 5.32. While these tracking errors exceed those observed in previous cases, the agent nonetheless successfully aligns the end-effector with the target's pose.

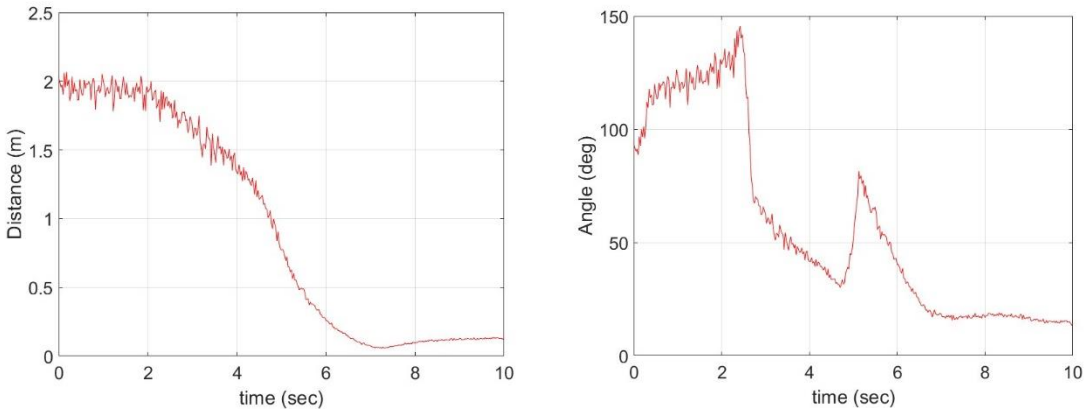


Figure 5.32 Case 4 Pose tracking errors (d, ϑ) over time

Furthermore, Figure 5.33 shows the time histories of joint torques generated by the agent, which are required to reach the target's pose. Compared to the torque patterns in Figure 5.24, a significant increase in torque oscillations is observed. This phenomenon can be attributed to the impact of noisy observations. These torque oscillations do not disappear even when the pose of the end-effector is aligned with the target's pose, the agent continuously adjusts the torques in order to maintain a stable pose of the end-effector amidst the perturbations induced by noise in the observed stat errors, as shown in Figure 5.32.

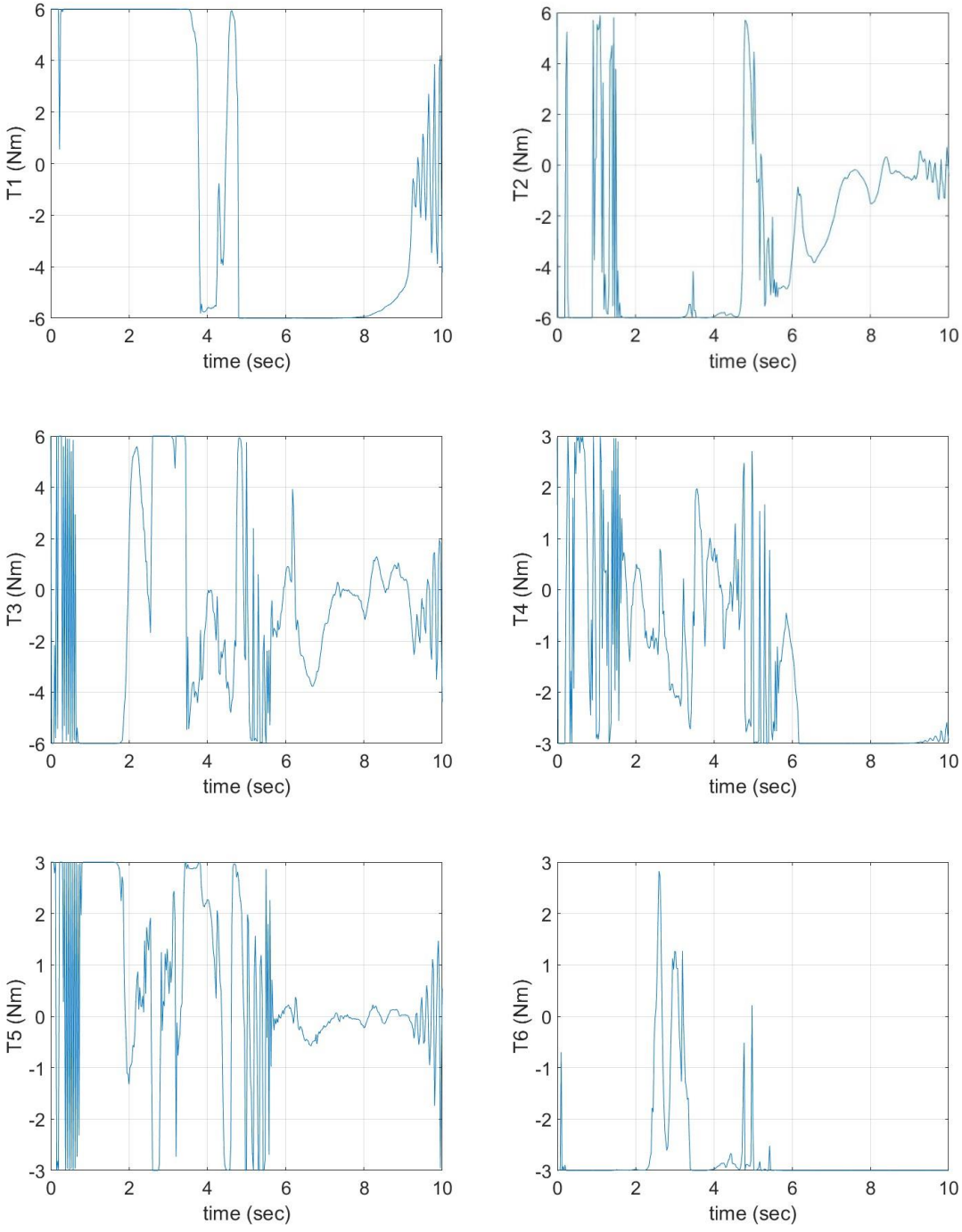


Figure 5.33 Case 4 time histories of joint torques given by the agent

The outcomes of all simulations conducted in Sec.5.3 are summarized in Table 5.5, which includes a comparison based on several key metrics: the required training episodes necessary for approach towards large rewards, the total CPU time elapsed during training, and the minimum recorded tracking errors for both position and orientation of the end-effector that were successfully achieved.

Table 5.5 Comparison between Simulation Cases in Sec.5.3

Comparison metrics / Simulation case	Episodes starting to approach	CPU times (second)	Position tracking error (meter)	Orientation tracking error (deg)
Case 1 – no velocity constraint	2,869	58,136	0.0513	7.0795
Case 2 – with velocity constraint	3,072	52,302	0.0509	5.0899
Case 3 – high mass ratio + velocity constraint	3,751	51,549	0.0192	5.8835
Case 4 – noisy observations + velocity constraint	4,567	103,510	0.0567	13.3055

5.4 Six DOF Space Robotic Manipulator with Obstacle

In this section, the numerical simulation is done for a 6DOF free-floating space robotic manipulator with an obstacle located between manipulator and target, in order to verify the effectiveness of the proposed RL control scheme. The manipulator, target and obstacle are shown in Figure 5.34. Three different cases were conducted to demonstrate the capabilities of the proposed RL approach. The first case employs the reward function without velocity constraints. In the second case the reward function was used with the velocity constraints in order to improve the path. And for the third case, the same reward function as the second case was used however, it included noisy observations, thereby closely reflecting realistic sensor observations.

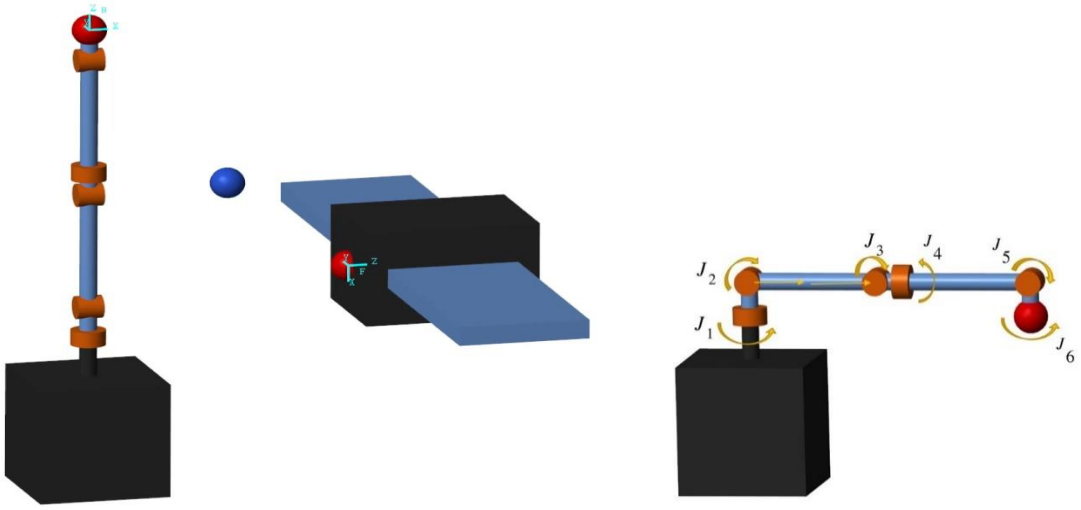


Figure 5.34 6 DOF Space robotic Manipulator system

The physical parameters of the space robotic manipulator, the target and the obstacle are given in Table 5.6. Table 5.7 provides the reinforcement learning training hyperparameters and the initial conditions used in the environment for all simulation cases.

Table 5.6 Parameters of Six DOF Space Robotic Manipulator

Body	Mass (kg)	Length (m)	*Ix* (kg/m^2)	*Iy* (kg/m^2)	*Iz* (kg/m^2)
Base	1700	0.75x0.75 x0.75	1434	1434	1434
Link 1	5	0.25	0.0292	0.0292	0.0063
Link 2	5	0.2	0.0292	0.0292	0.0063
Link 3	50	0.75	0.0625	26.1	26.1
Link 4	10	0.2	0.0125	0.215	0.215
Link 5	50	0.75	0.0625	26.1	26.1
Link 6	5	0.2	0.0063	0.0292	0.0292
Target Sphere	0	radius 0.1	0	0	0
Obstacle Sphere	0	radius 0.1	0	0	0

Table 5.7 Reinforcement Learning Parameters

Parameter	Value
Learning rate	0.001
Replay Buffer	50,000
Discount factor for future reward	0.99

Smooth factor	0.001
Mini-batch size	128
Number of Episodes	3000
Episode timesteps	400
Timestep size	0.025 s
Episode time	10 s
Neural network architecture for Actor and Critic	Hidden layers (400, 300)
Bounds for joint torques (1-3)	-3 Nm, 3 Nm
Bounds for joint torques (4-6)	-6 Nm, 6 Nm
Initial EE pose $(x, y, z, \xi)_e$	(0, 0, 2.675, -1, 0, 0, 0)
Target pose $(x, y, z, \xi)_T$	(0.8, 0, 1.6, 0.707, 0, 0.707, 0)
Obstacle position $(x, y, z)_{Ob}$	(1.5, 0, 1)

It is worth pointing out that the RL hyperparameters play a pivotal role in governing various aspects of the learning process, including learning dynamics, exploration versus exploitation trade-offs, convergence, adaptation to changing environments, and computational resource utilization. The training is greatly influenced by the learning rate, replay buffer and discount factor. The learning rate dictates the pace at which the RL agent updates its estimates based on new information, influencing both the speed and stability of learning. Similarly, a larger replay buffer enhances sample efficiency and stability by enabling the agent to learn from a diverse array of past experiences while mitigating overfitting risks. Moreover, the discount factor is essential for

the agent's long-term decision-making, allowing it to balance immediate rewards with future gains. This factor significantly influences the progress and stability of RL algorithms and aids in handling uncertainty and delayed rewards, thus facilitating effective exploration and exploitation strategies.

5.4.1 Case 1: Obstacle Avoidance without Velocity Constraints

In this case, the free-floating space robotic manipulator has been trained using the reward function in Equation(4.47). The parameters in this reward function are defined as $k_d = 0.1$, $k_\vartheta = 0.001$, $d_0 = 0.2m$, $\vartheta_0 = 30^\circ$, $r_d = 1$, $r_\vartheta = 2$, $p_{Ob} = 0.5$, $d_{Ob} = 0.3$. These parameters are used in all simulation cases described in the next sections.

The training progress is shown in Figure 5.35. These curves show that the agent learns to reach the desired pose by searching for the highest reward. The term $(-K_d d - K_\vartheta \vartheta)$ in the reward function efficiently guides the EE towards the capture range within the first 5000 episodes, while the penalty term $(-p_l - p_{Ob})$ in the reward function effectively prevents collisions of the manipulator with the obstacle, the target and itself.

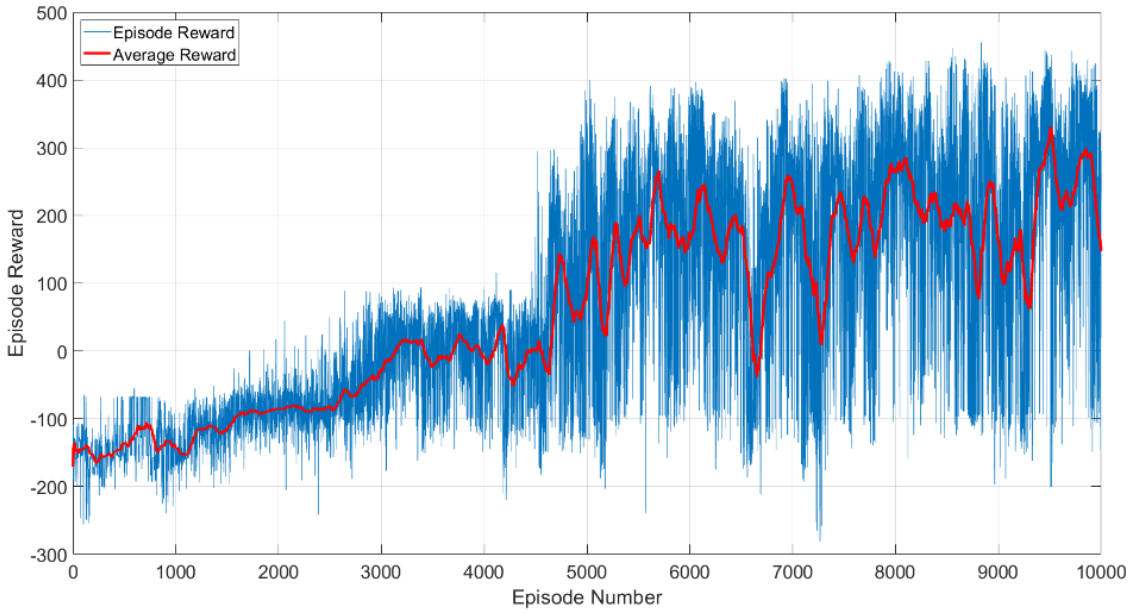


Figure 5.35 Case 1 Training result :10,000 episodes

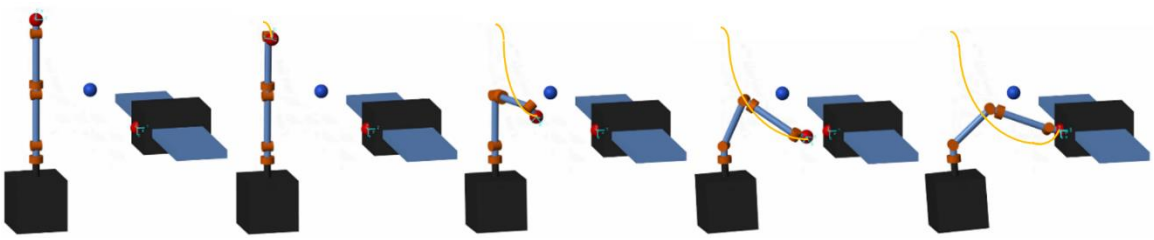


Figure 5.36 Motion of implemented trained agent from Case 1

Upon entering the capture range, the sparse rewards $(r_d + r_g)$ in the reward function are activated to keep the EE within the capture range while simultaneously refining the EE's pose throughout the remaining timesteps to prepare for capture. The large variation of the episode reward reflects the activation and deactivation of the large rewards $(r_d + r_g)$ in this process. Figure

5.36 shows snapshots of the output path for the space robotic manipulator towards the target.

Figure 5.37 shows the pose error between the EE and the target over time with the left subfigure representing the distance error (d) and the right subfigure depicting the orientation difference (ϑ). The trained agent guides the EE towards the desired target pose range (d_0, ϑ_0) within approximately 6 seconds. Subsequently, the EE stays within that region to achieve capture for the remaining duration of the simulation. Notably, the threshold ($d_0 = 0.20m$) does not imply that the minimum distance error the trained agent can reach; instead, it signifies that crossing this threshold secures a large positive reward (r_d). As a result, the agent achieves a minimum distance error of d = 0.058m and orientation error of ϑ = 4.85 degrees.

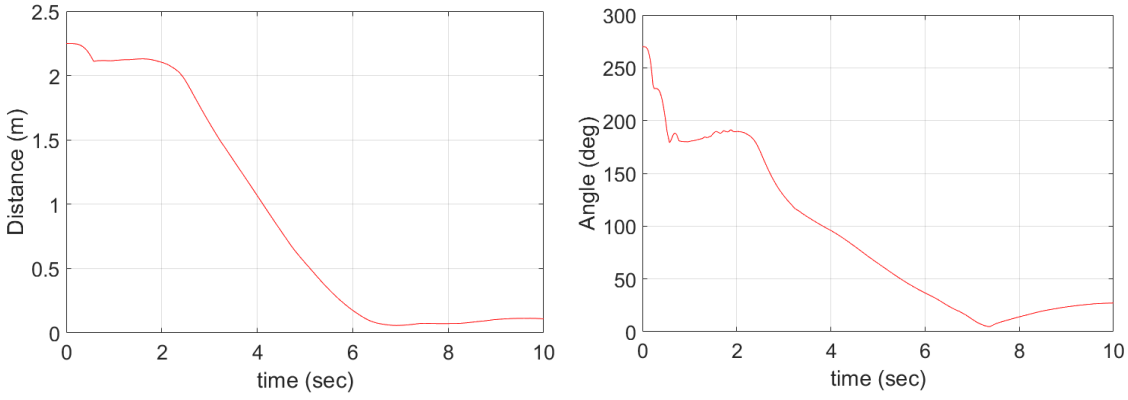
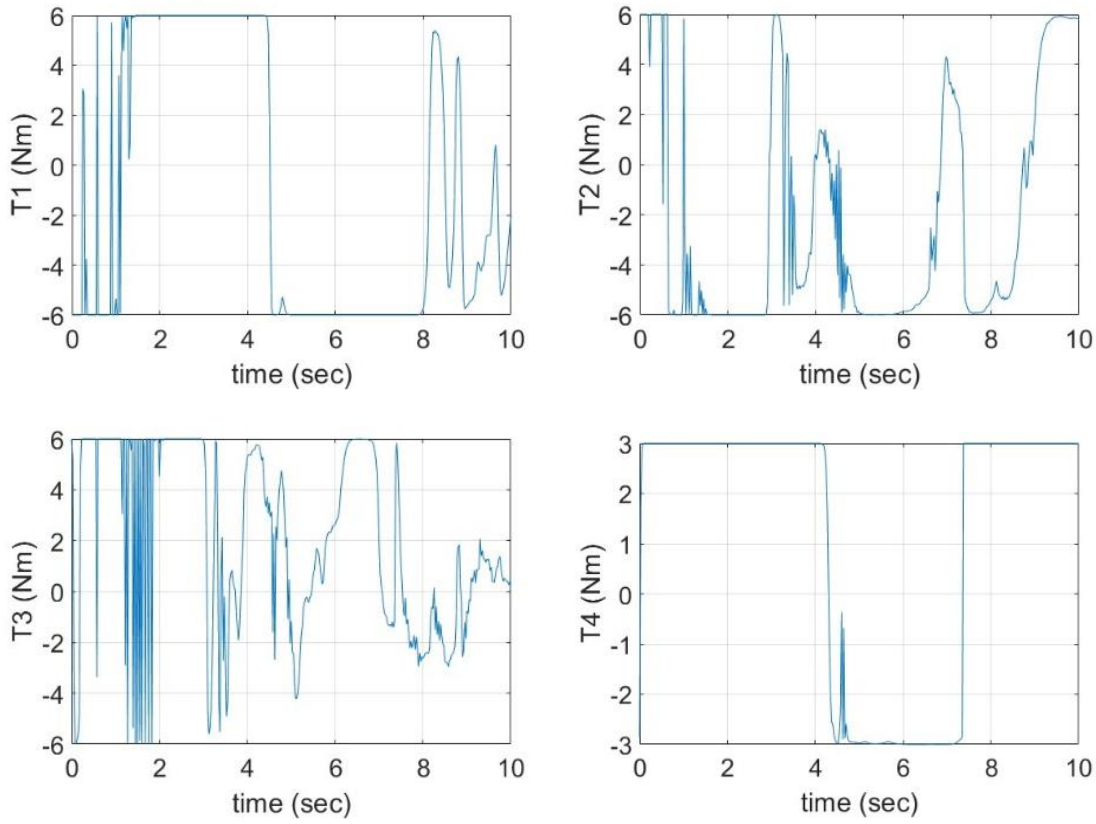


Figure 5.37 Case 1 Pose tracking errors (d, ϑ) over time

Figure 5.38 shows the time histories of actions (control torques) at all six joints as given by the agent. The agent effectively provides the necessary torques to achieve a valid path for the manipulator to move the EE towards the target, ensuring no self-collision occurs between manipulator links, and most importantly, avoiding the obstacle placed between the EE and the target.



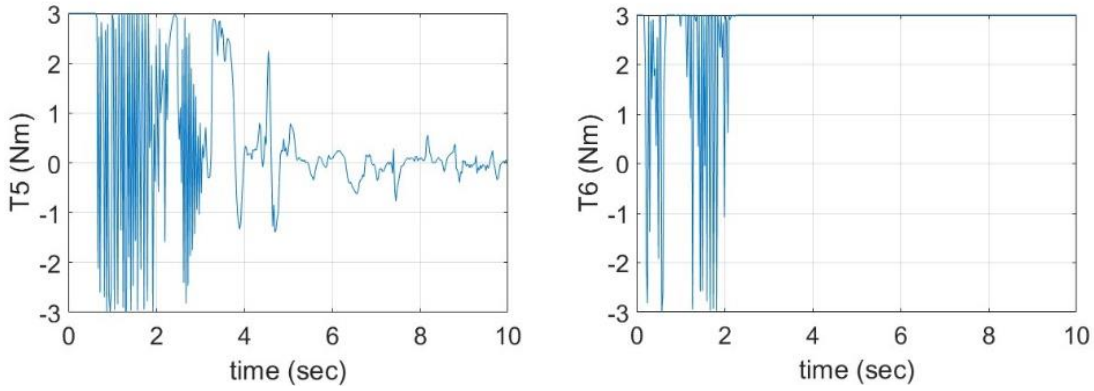


Figure 5.38 Case 1 time histories of joint torques given by the agent

5.4.2 Case 2: Obstacle Avoidance with Velocity Constraint

To achieve smoother motion and eliminate undesired sudden changes in the path, often associated with high joint angular velocities, a velocity penalty term for all six joint velocities ($-p_j\sum(\dot{\varphi})^2$) has been added to the reward function, as shown in Equation(4.51). The parameters for this simulation case are the same as those in case 1, with the inclusion of an additional parameter of $p_j = 0.005$. The training result is shown in Figure 5.39. along with the progress of the reward function. Figure 5.40 shows the pose error between the EE and the target over time, and Figure 5.41 shows the time histories of actions (control torques) for all six joints given by the agent.

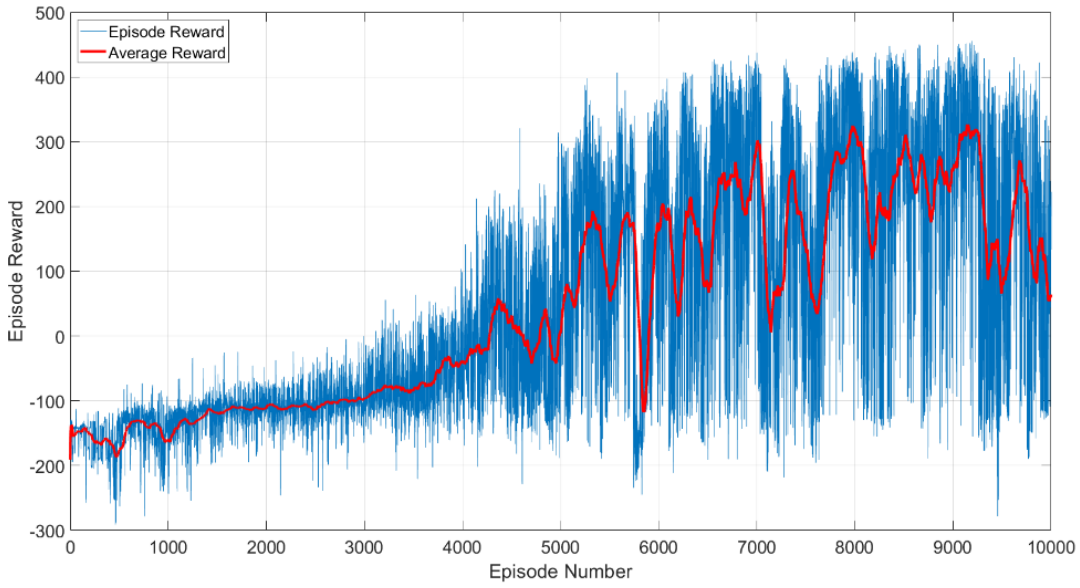


Figure 5.39 Case 2 Training result :10,000 episodes

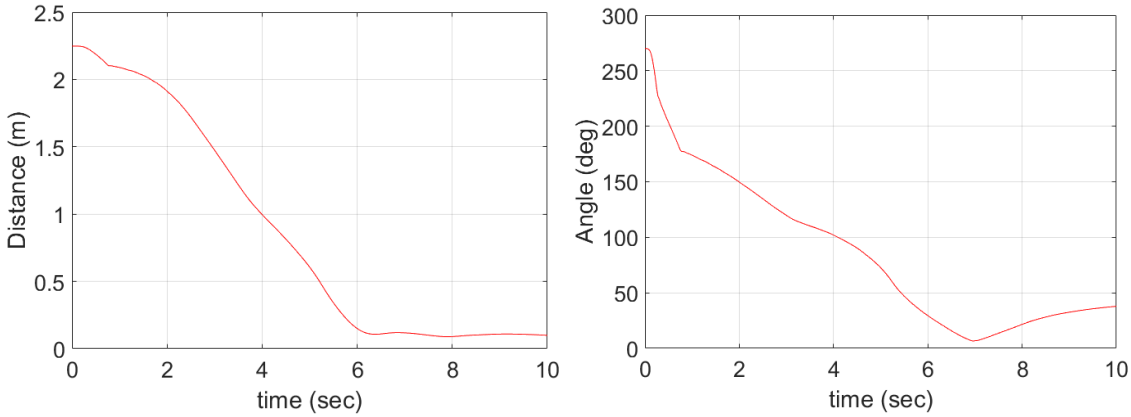


Figure 5.40 Case 2 Pose tracking errors (d, ϑ) over time

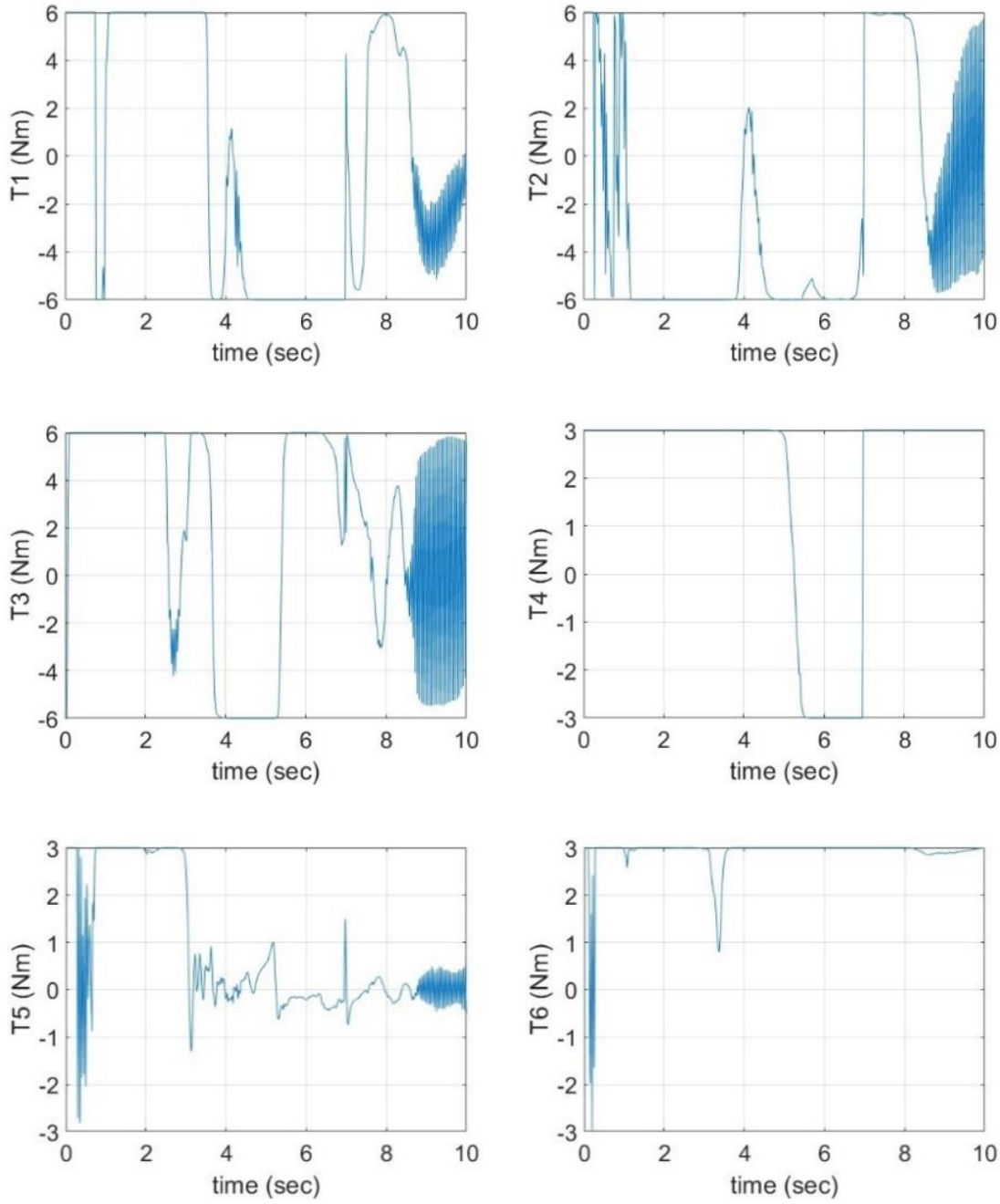


Figure 5.41 Case 1 time histories of joint torques given by the agent

In comparison to Figure 5.35, Figure 5.39 shows that the training process is similar to Case 1. The training approaches the capture range in

approximately 5,000 episodes. Upon entering the capture range, the large rewards ($r_d + r_\vartheta$) in the reward function are activated to keep the EE within the capture range while simultaneously refining the EE's pose throughout the remaining timesteps to prepare the gripper for capture. However, it is noted that in Case 2, the introduction of a velocity penalty term ($-p_j \sum(\dot{\varphi})^2$) to the reward function results in a smoother path. Comparing Figure 5.40 with Figure 5.37 reveals an improvement in the approach of the path in the first 3 seconds. Furthermore, the end-effector successfully approaches towards the capture range. The high rewards ($r_d + r_\vartheta$) helps the agent staying in this region for future capture and servicing with a minimum distance error of d = 0.088m and orientation difference of ϑ = 6.61 degrees.

To further examine the effect of the velocity constraint term on agent training, the square sum of all six joint velocities ($\sum(\dot{\varphi})^2$) along the path is shown in Figure 5.42. It is observed that the square sum of velocities obtained from the path in Case 2 (right plot) barely surpasses 80 in comparison to the 160 reached in Case 1 (left plot).

These results clearly demonstrate the efficacy of modifying the reward function on the solution. Such findings open the door for future research to enhance path optimization by designing better reward functions, potentially

incorporating torque penalties, energy consumption penalties, and the integration of optimal path constraints commonly used in optimization to Reinforcement Learning.

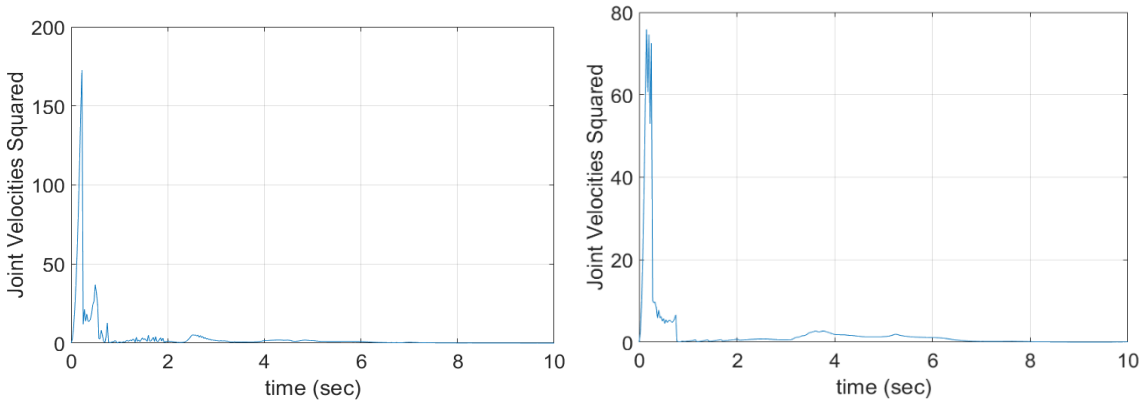


Figure 5.42 Square sum of all joint angular velocities in Case 1 (left) and Case 2 (right).

5.4.3 Case 3: Obstacle Avoidance with Noisy Observations

In real missions, assuming perfect observations of the state is unrealistic, particularly in estimating the target's pose using vision sensors like cameras. To evaluate the robustness of the proposed reward function in handling noisy observations, training of the agent was proceeded with the same reward function and parameters in Case 2, except introducing white noise to the observations that would be obtained from cameras in real mission. The noise is added to the target pose and obstacle position: $\boldsymbol{p}_T, \alpha_T, \beta_T, \gamma_T, \boldsymbol{p}_{Ob}$ with a signal-to-noise ratio (SNR) of 10/1 to simulate realistic conditions:

$$Signal_{out} = Signal_{in}\left[1 + R_n\right] \tag{5.6}$$

where $R_n \in [-0.1, 0.1]$ is a random number with *Mean=0* and *Variance=1*.

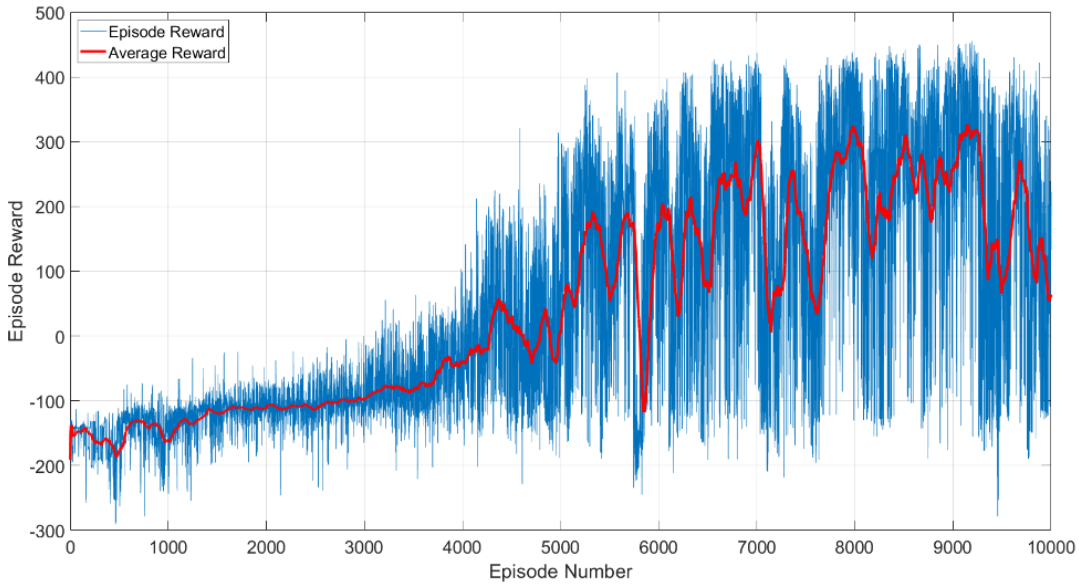


Figure 5.43 Case 3 Training result :10,000 episodes

The training outcome is shown in Figure 5.43. It is noted that the agent is able to obtain high rewards despite the presence of observational noise. Figure 5.44 shows the pose error between the EE and the target over time. Despite the presence of noise in the target position and orientation observations and obstacle position, the agent effectively manages to align the EE and the target for capture while avoiding the obstacle along the path. The agent achieves a minimum distance error of Δd = 0.048m and orientation difference of $\Delta\theta = 7.79$ degrees.

Notably, as the EE approaches the target, the absolute value of the noise

decreases, imitating the characteristic behavior observed in real camera observations. The agent successfully overcomes the noisy inputs, which proves the effectiveness and robustness of the solution.

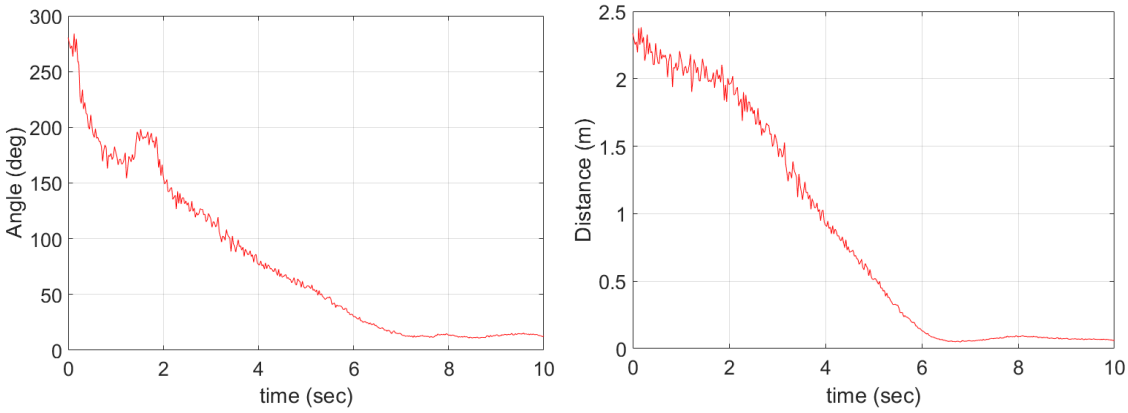


Figure 5.44 Case 3 Pose tracking errors (d, ϑ) over time

Figure 5.45 shows the time histories of actions (control torques) for all six joints given by the agent. It reveals that the torques required for capture exhibit more oscillations compared to the previous solution in Figure 5.41. This increase in oscillation results from the noisy observations that add uncertainties during training.

It is worth mentioning that the agent trained in Case 3 (noise on observations) still manages to achieve the desired task when implemented in a noise-free environment. Conversely, the agent trained in Case 2 (no noise on observations) fails its task in a noisy environment.

This highlights the importance of RL training under realistic and challenging conditions. Exposing the agent to noise during training equips it with the ability to effectively achieve its task in real-world applications.

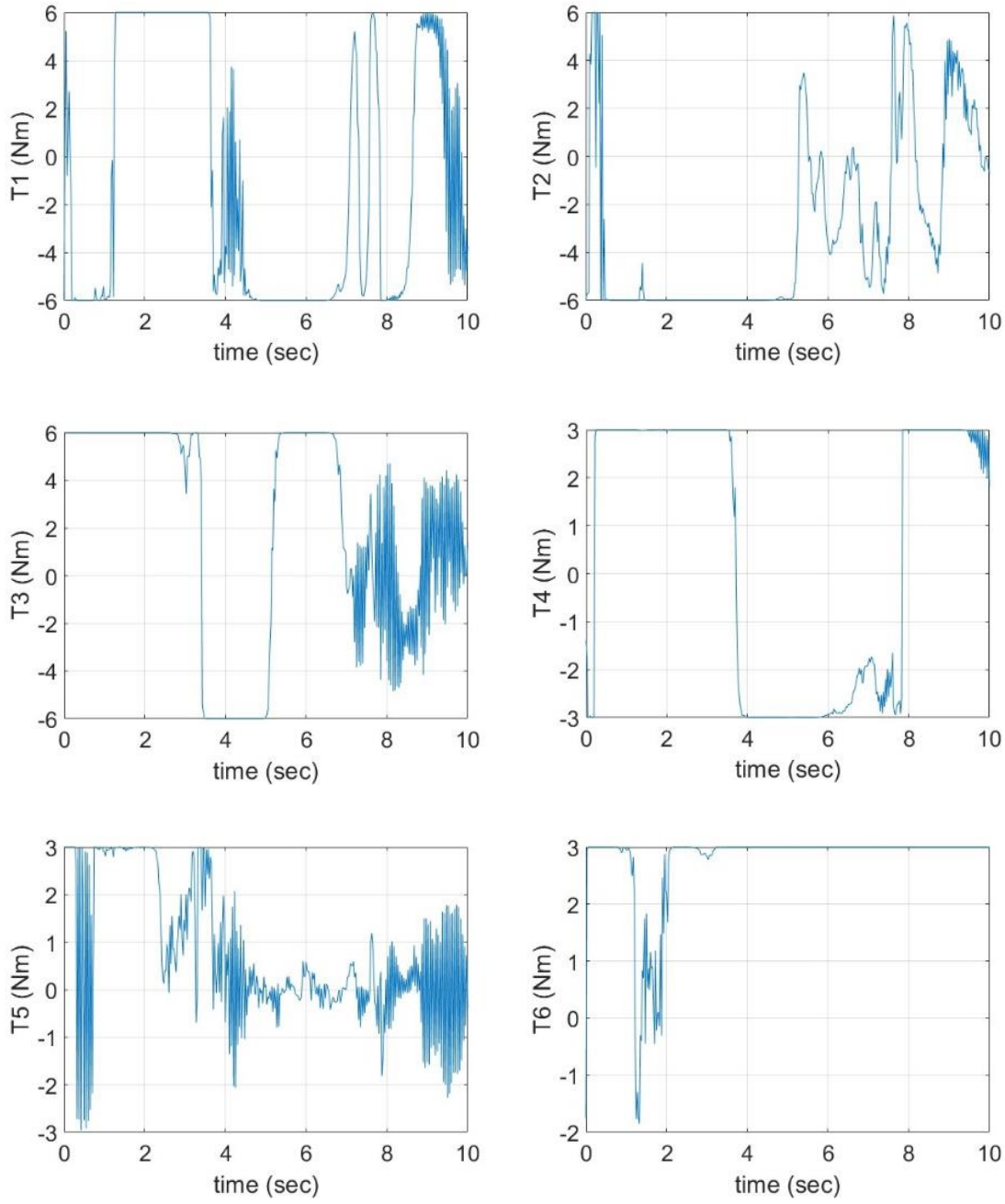


Figure 5.45 Case 3 time histories of joint torques given by the agent

Chapter 6 MULTI-SCENARIO REINFORCEMENT LEARNING

Summary: In this chapter, the agent will be trained using multi-scenario reinforcement learning, a new approach where the training environment changes dynamically with each training episode and its efficacy has been demonstrated by numerical simulation.

Part of theatrical and simulation results will be published in the reference paper I. All variable symbols in this chapter refer to the definitions in Section 4.3

6.1 Introduction

Unlike the RL in previous Chapter, the agent will be trained using multi-scenario reinforcement learning, an approach where the training environment changes dynamically with each training episode. This includes varying initial conditions such as random manipulator joint angles, target positions/ orientations, and obstacle placements. This method aims to create a versatile agent capable of achieving target capture under any circumstances, meaning it can reach any target from any position, regardless of obstacles in its path. The goal is to achieve human-level capture efficiency.

A review of the previous literature, as summarized in Table 2.3, reveals a notable absence of multi-scenario reinforcement learning (RL) training.

Traditionally, RL has been employed to train agents for single mission scenarios. The next step in advancing this field is to develop an agent capable of achieving target capture for any mission without the need for retraining. It is my belief that this objective can be accomplished through multi-scenario reinforcement learning.

The pseudo code that describes the multi-scenario RL training is shown in Table 6.1. It is based of the DDP procedure in Table 4.1 but with randomly selected initial conditions for the environment at the beginning of every episode.

Table 6.1 Multi-scenario RL Algorithm Procedure

Randomly initialize online actor $\mu(\mathbf{s}|\theta^{\mu})$ and critic $Q(\mathbf{s},\mathbf{a}|\theta^{Q})$ networks by respective weights $(\theta^{\mu},\theta^{Q})$.

Reset target actor μ' and critic Q' networks with weights $\theta^{\mu}\rightarrow\theta^{\mu'}$, $\theta^{Q}\rightarrow\theta^{Q'}$.

Reset replay buffer (RAM).

For each episode $\boldsymbol{M}$ do:

Randomly select initial six robotic joint angles $(\Phi_1,\Phi_2,\Phi_3,\Phi_4,\Phi_5,\Phi_6)$

Randomly choose target position (x_T,y_T,z_T) and orientation $(\alpha_T,\beta_T,\gamma_T)$

Randomly choose obstacle position (x_{Ob},y_{Ob},z_{Ob})

For each timestep:

Select an action $\mathbf{a}=\mu(\mathbf{s}|\theta^{\mu})+\mathbb{N}$, where $\boldsymbol{s}$ and $\mathbb{N}$ are current state and random process exploration noise.

Execute action $\boldsymbol{a}$, observe reward R & new state $\mathbf{s}_{i+1}$.

Store the experience $(\mathbf{s}, \mathbf{a}, R_i, \mathbf{s}_{i+1})$ in the replay buffer.

Sample random minibatch of experiences $(\mathbf{s}, \mathbf{a}, R_i, \mathbf{s}_{i+1})$ from the replay buffer.

If $\mathbf{s}_{i+1}$ is the terminal state, set the value function target as

y = R

Else, Set the value function target as

$$y = R + \gamma Q'(\mathbf{s}_{i+1}, \mu'(\mathbf{s}_{i+1} \mid \theta^{\mu'}) \mid \theta^{Q'})$$

Update critic parameters by minimizing the loss L on all sampled experiences,

$$L = \frac{1}{M} \sum_{i=1}^{M} \left[y - Q\left(s, a \middle| \theta^{Q}\right) \right]^2$$

Update actor parameters to maximize the expected discounted reward by the sampled policy gradient,

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{M} \sum_{i=1}^{M} \nabla_a Q\left(s, a \middle| \theta^{Q}\right) \nabla_{\theta^{\mu}} \mu\left(s \middle| \theta^{\mu}\right)$$

Update target actor and critic parameters by:

$$\theta^{Q'} = \tau\theta^{Q} + \left(1-\tau\right)\theta^{Q'} \text{ and } \theta^{\mu'} = \tau\theta^{\mu} + \left(1-\tau\right)\theta^{\mu'}$$

End for

End for

At the beginning of each episode, initial conditions are randomly selected using the MATLAB function Rand, while adhering to specific constraints to avoid self-collisions between robotic links and stay within joint limits. The initial conditions that are changed include the initial joint angles,

the target pose, and an obstacle positioned between the robot and target.

6.2 Six DOF Space Robotic Manipulator

6.2.1 Random Target Positions

In this section the Environment for a six degrees-of-freedom free floating space manipulator is used to train a dynamic agent. The environment previously built in Section 5.3 is used for training. However, the environment is rebuilt to have different, randomly chosen target positions at the start of each training episode within the ranges in shown in Table 6.2.

Table 6.2 Multi-scenario RL Initial Conditions 1

Parameter	Random Value for Training	For Testing
Target x	Between (0.5 to 1.5) meters	1.163 m
Target y	Between (-0.5 to 0.5) meters	0.023 m
Target z	Between (0.5 to 1.5) meters	0.76 m

The reward function used was the same as previously designed.

$$R_t = -K_d d - K_\vartheta \vartheta + r_d + r_\vartheta - p_l - p_j \sum \left(\dot{\varphi}\right)^2 \tag{6.1}$$

The training was conducted for 10,000 episodes and the result of training is shown in Figure 6.1. Notably, the training process requires a larger number of episodes to achieve progress. Despite this, it successfully approaches

high rewards associated with successful capture. The figure shows considerable oscillation, which results from the random placement of the target; in some episodes, the target may be unreachable, resulting in lower rewards, while in others, successful captures lead to high rewards.

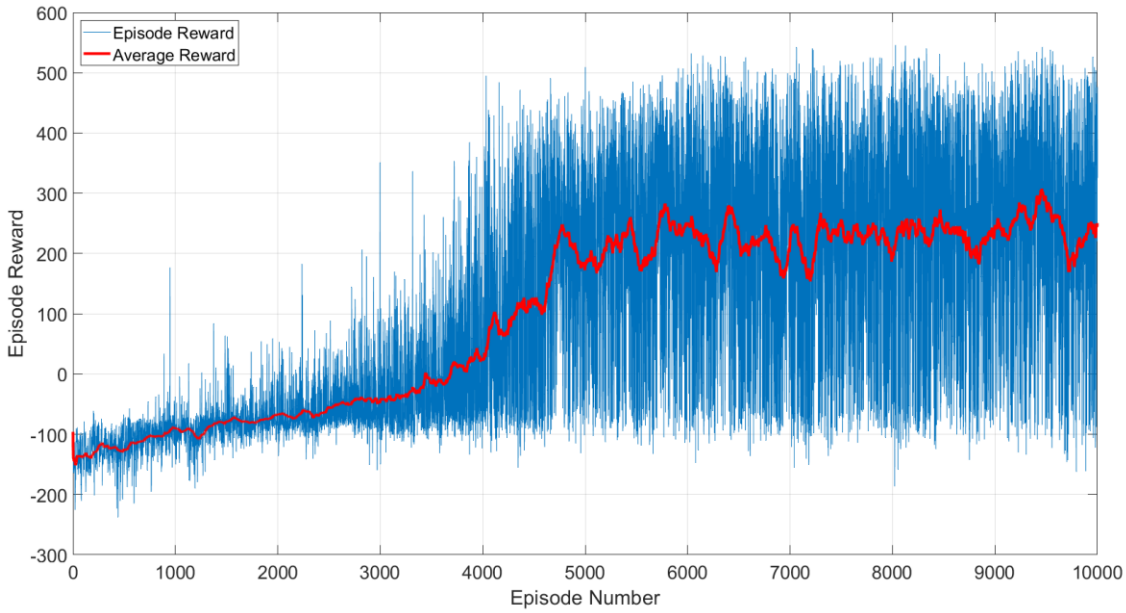


Figure 6.1 Multi-scenario RL Training result :10,000 episodes

Although the training took 45 hours, once the agent was fully trained it was merely 3.7megabytes and had the ability to achieve successful capture of different target positions with a 72% success rate. It did not require re-training every time the initial conditions of the capture mission change, as seen in optimization techniques. The minimum distance error achieved by the agent was of d = 0.0214m and orientation error of ϑ = 0.898 degrees.

Furthermore, the fully trained agent does not need any computation, it

is made of a combination of wights and bias, it can be implemented on any small chip as the ones used in small satellites.

For testing the target position is chosen randomly, see Table 6.2. Figure 6.2 shows the pose error between the EE and the target over time. The trained agent guides the EE towards the desired target pose range within approximately 7 seconds. The agent achieves a minimum distance error of d = 0.118m and orientation error of ϑ = 1.5 degrees.

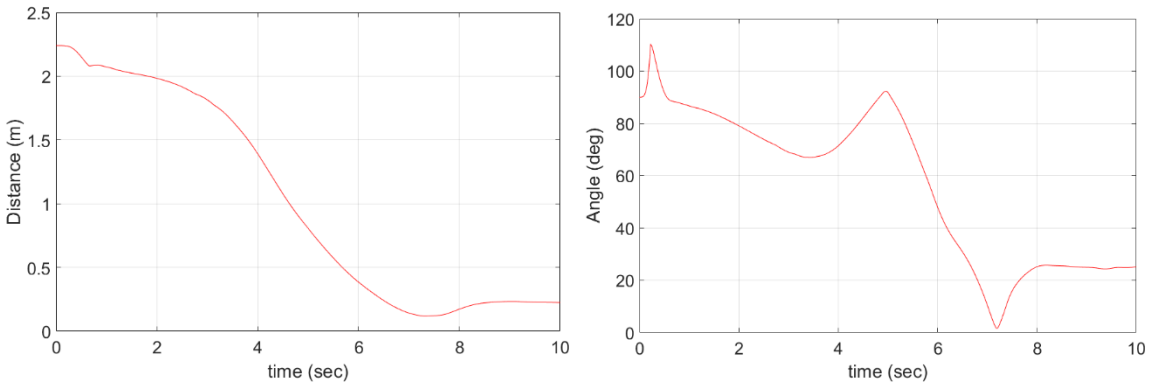


Figure 6.2 Capture Test Pose tracking errors (d, ϑ) over time

Figure 6.3 shows the time histories of actions (control torques) at all six joints as given by the agent. The agent effectively provides the necessary torques to achieve a valid path for the manipulator to move the EE towards the target, ensuring no self-collision occurs between manipulator links.

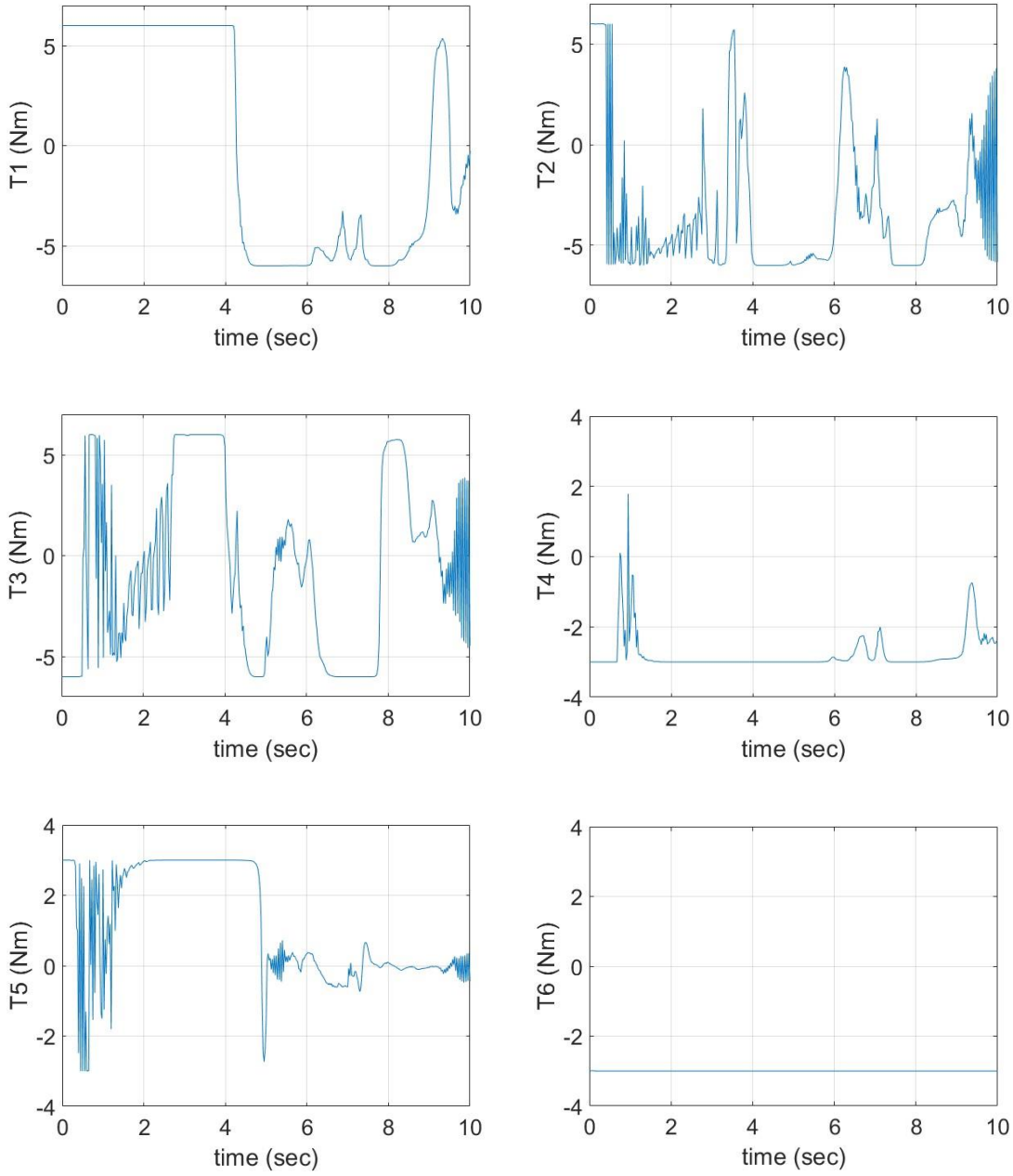


Figure 6.3 Capture Test joint torques time histories

6.2.2 Random Target Positions and Orientations

In this section the Environment for a six degrees-of-freedom free floating

space manipulator is used to train a dynamic agent. The environment previously built in Section 5.3 is used for training. However, the environment is rebuilt to have different, randomly chosen target positions and orientations at the start of each training episode within the ranges in shown in Table 6.3.

Table 6.3 Multi-scenario RL Initial Conditions 2

Parameter	Random Value for Training	For Testing
Target x	Between (0.5 to 1.5) meters	1.13 m
Target y	Between (-0.5 to 0.5) meters	0.296 m
Target z	Between (0.5 to 1.5) meters	1.191 m
Target roll	Between (0 to +30) degrees	10.36 deg
Target pitch	Between (0 to +30) degrees	28.405 deg
Target yaw	Between (0 to +30) degrees	15.606 deg

The reward function used was the same as previously designed.

$$R_t = -K_d d - K_\vartheta \vartheta + r_d + r_\vartheta - p_l - p_j \sum \left(\dot{\varphi}\right)^2 \tag{6.2}$$

The training was conducted for 10,000 episodes, and the results are shown in Figure 6.4. Notably, even after ten thousand episodes, the agent barely learns to achieve capture, indicating that the training process requires a larger number of episodes. Consequently, the training was extended to 15,000 episodes. The results, shown in Figure 6.5, demonstrate successful progress towards high rewards associated with successful position and orientation alignment. The dynamic agent achieves capture across various

target positions and orientations scenarios with a success rate of 71%. The minimum distance error achieved by the agent was of d = 0.0161m and orientation error of ϑ = 3.2838 degrees.

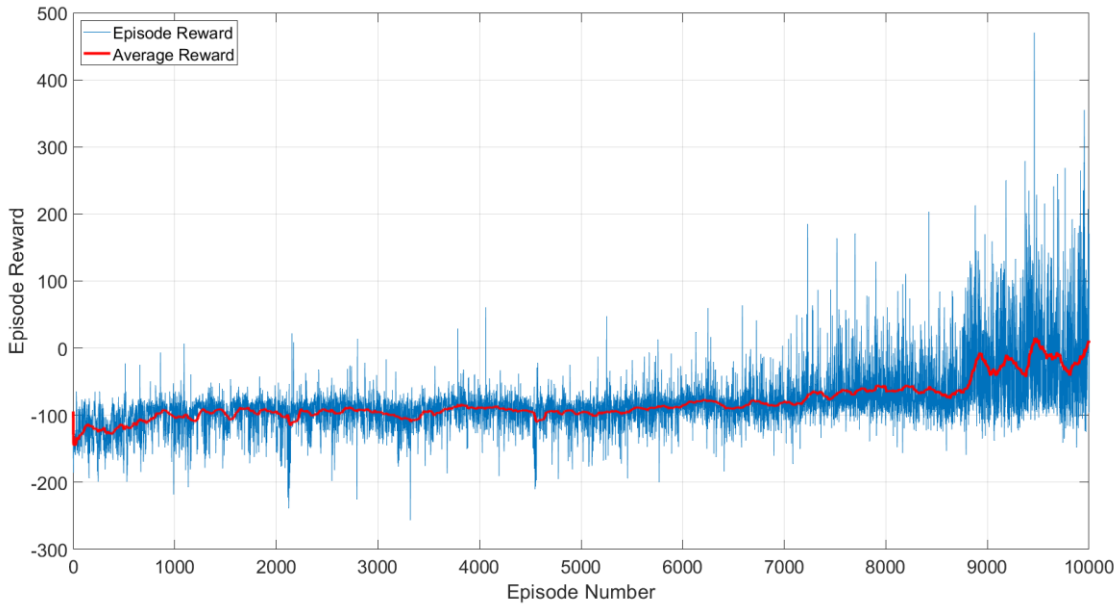


Figure 6.4 Multi-scenario RL Training result :10,000 episodes

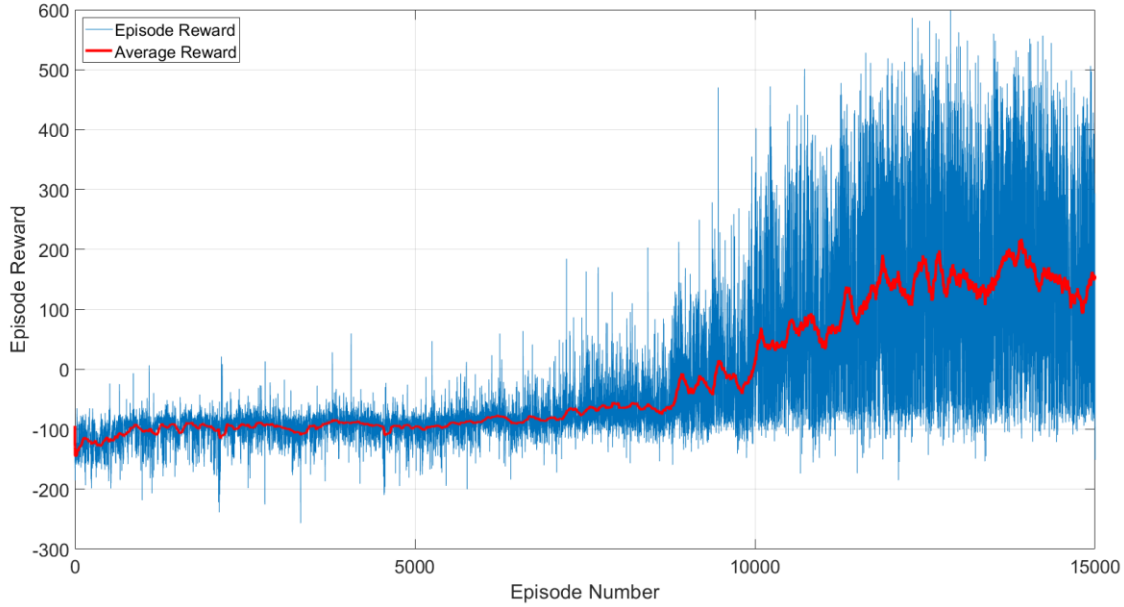


Figure 6.5 Multi-scenario RL Training result :15,000 episodes

For testing the target position is chosen randomly, see Table 6.3. Figure 6.6 shows the pose error between the EE and the target over time. The trained agent guides the EE towards the desired target pose range within approximately 7 seconds. The agent achieves a minimum distance error of d = 0.0401m and orientation error of ϑ = 7.0898 degrees. Figure 6.7 shows the time histories of actions (control torques) at all six joints as given by the agent.

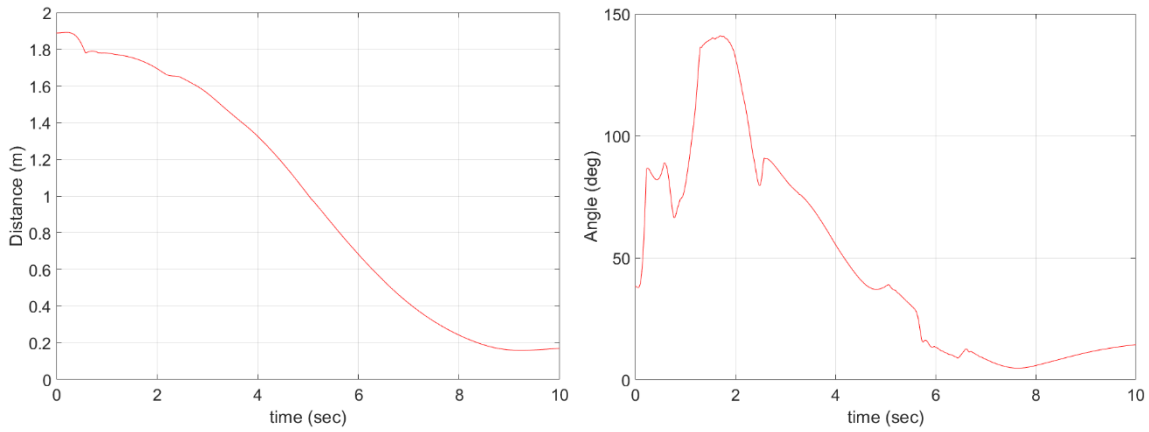
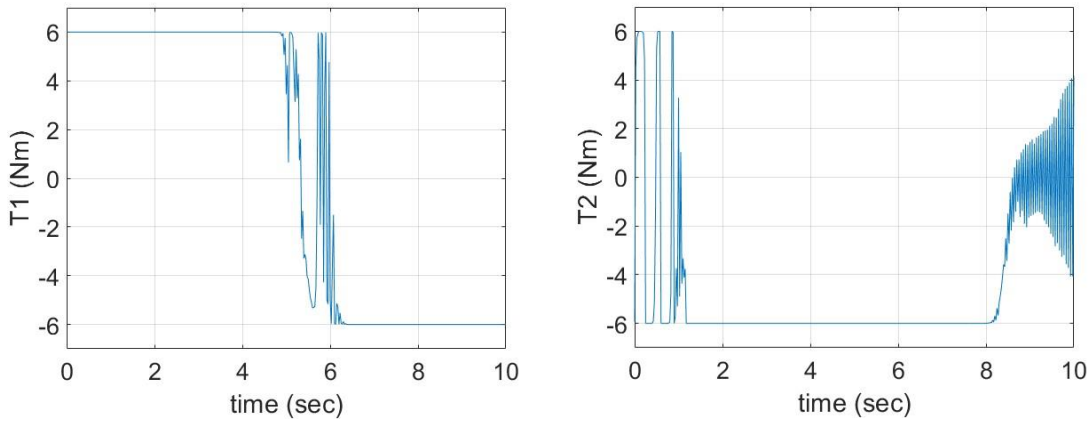


Figure 6.6 Capture Test Pose tracking errors (d, ϑ) over time



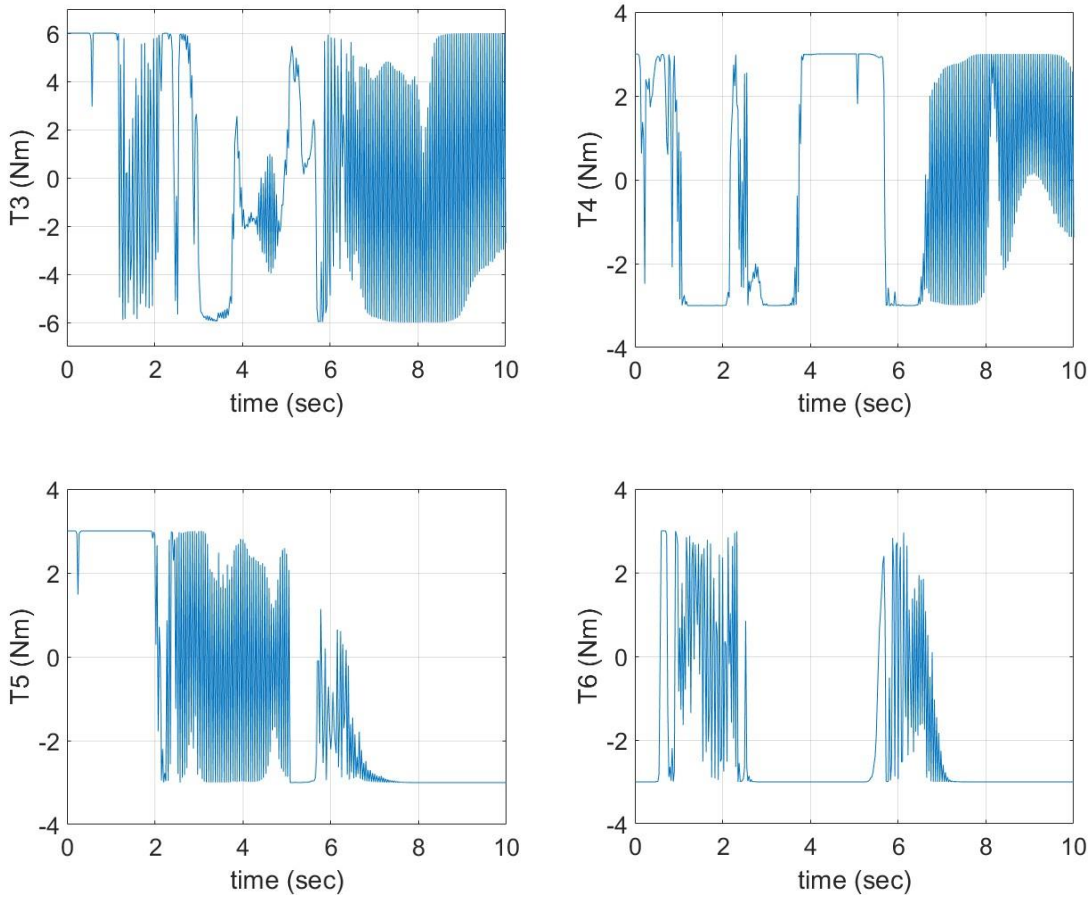


Figure 6.7 Capture Test joint torques time histories

6.3 Six DOF Space Robotic Manipulator with Obstacle

In this section the Environment for a six degrees-of-freedom free floating space manipulator with obstacle avoidance is used to train a dynamic agent. The environment previously built in Section 5.4 is used for training. However, the environment is rebuilt to have different, randomly chosen target positions,

obstacle positions, and initial robotic joint angles at the start of each episode.

In each subsection the multi-scenario training will be conducted with one or more changing initial conditions. Furthermore, each subsection will be dynamically trained for three different cases, The first case is trained without any optimization term in the reward function, the second case will be trained with a velocity constraint term, and the third case will be trained under noisy observations. These three cases are exactly similar to the cases in Section 5.4. However, the initial conditions change at each episode.

This will create a robust dynamic agent that has the ability to smoothly achieve position and orientation alignment with any randomly positions target, from any initial joint positions, while avoiding any obstacle placed in its path, and under noisy observations.

6.3.1 Random Initial Joint Positions

The training in this section will be done on the same environment, and reward functions previously described in Section 5.4, with random initial joint positions at the start of each episode within the ranges in shown in Table 6.4.

Table 6.4 multi-scenario RL Initial Conditions 1

Parameter	Random Value for Training	For Testing		
		Case 1	Case 2	Case 3
Joint1	between (60 to 120) deg	115.032	78.673	62.581
Joint2	between (0 to 30) deg	8.575	15.856	5.069
Joint3	between (0 to 30) deg	22.716	4.969	19.473
Joint4	between (0 to 30) deg	22.612	18.059	21.951
Joint5	between (-30 to 30) deg	-7.173	14.221	8.864
Joint6	between (-30 to 30) deg	4.069	9.244	-2.944
Target x	1.6 m	1.6 m	1.6 m	1.6 m
Target y	0.0 m	0.0 m	0.0 m	0.0 m
Target z	1.0 m	1.0 m	1.0 m	1.0 m
Obstacle x	0.9 m	0.9 m	0.9 m	0.9 m
Obstacle y	0.0 m	0.0 m	0.0 m	0.0 m
Obstacle z	1.5 m	1.5 m	1.5 m	1.5 m

6.3.1.1 Random Joint Positions Case 1: without Velocity Constraints

This case is similar to case 1 in Section 5.4.1. The same reward function without the velocity constraint term is used. However, it is trained with using multi-scenario RL. The training result is shown in Figure 6.8 where the training convergences nicely. Within the first 5,000 episodes it gains the high rewards from position alignment, after 7,000 episodes it learns to achieve position and orientation alignment.

The fully trained agent has the ability to achieve successful capture of

the target pose from different initial joint positions with a 94% success rate. It can also achieve capture of the target pose while avoiding the obstacle with a 40% success rate. The minimum distance error achieved by the agent was of d = 0.0031m and orientation error of ϑ = 4.4002 degrees.

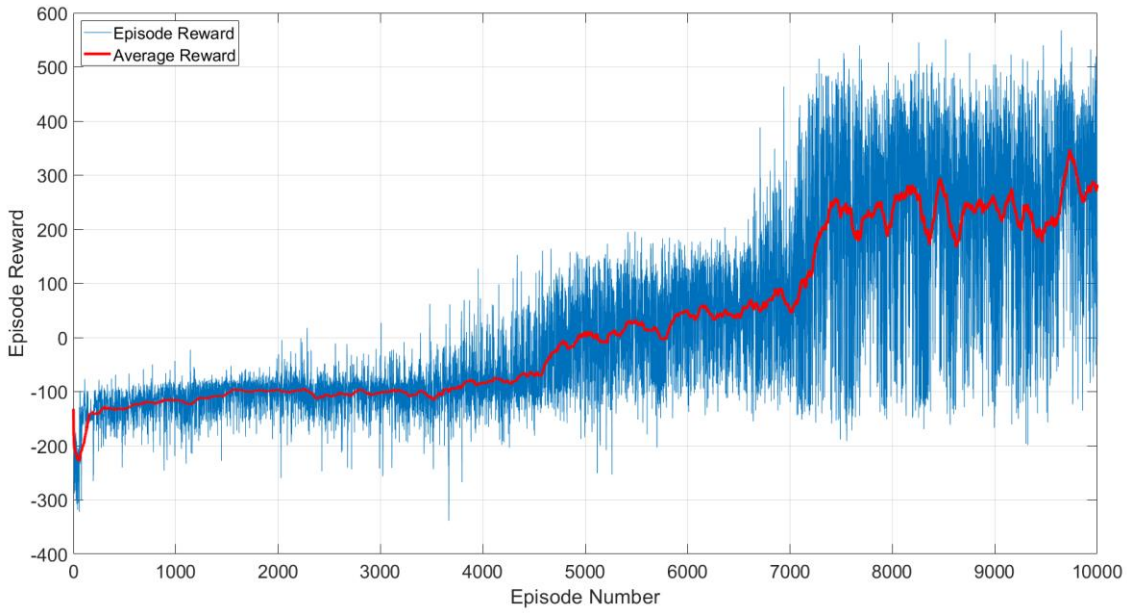


Figure 6.8 Multi-scenario RL Case 1 Training result :10,000 episodes

For testing case 1 the initial joint positions are chosen randomly, see Table 6.4. Figure 6.9 shows the pose error between the EE and the target over time. The trained agent guides the EE towards the desired target pose range within approximately 7 seconds. The agent achieves a minimum distance error of d = 0.0378m and orientation error of ϑ = 11.69 degrees. Figure 6.10 shows the time histories of actions (control torques) at all six joints as given by the agent.

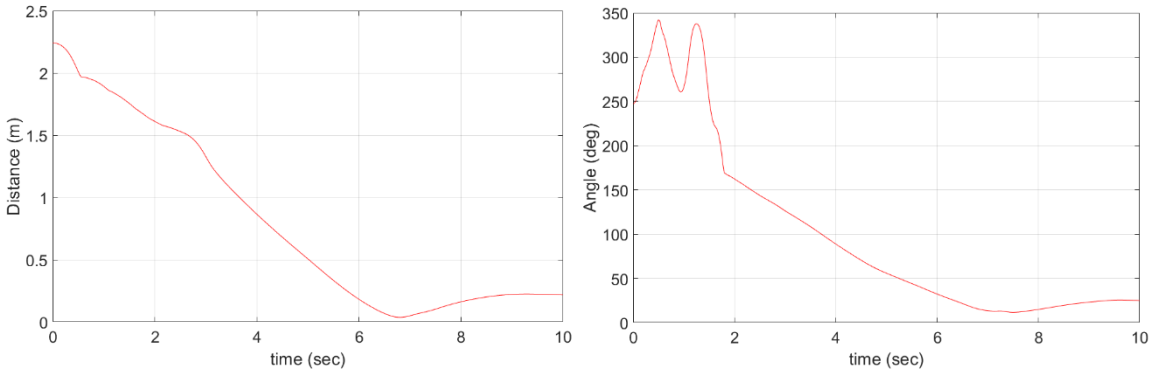
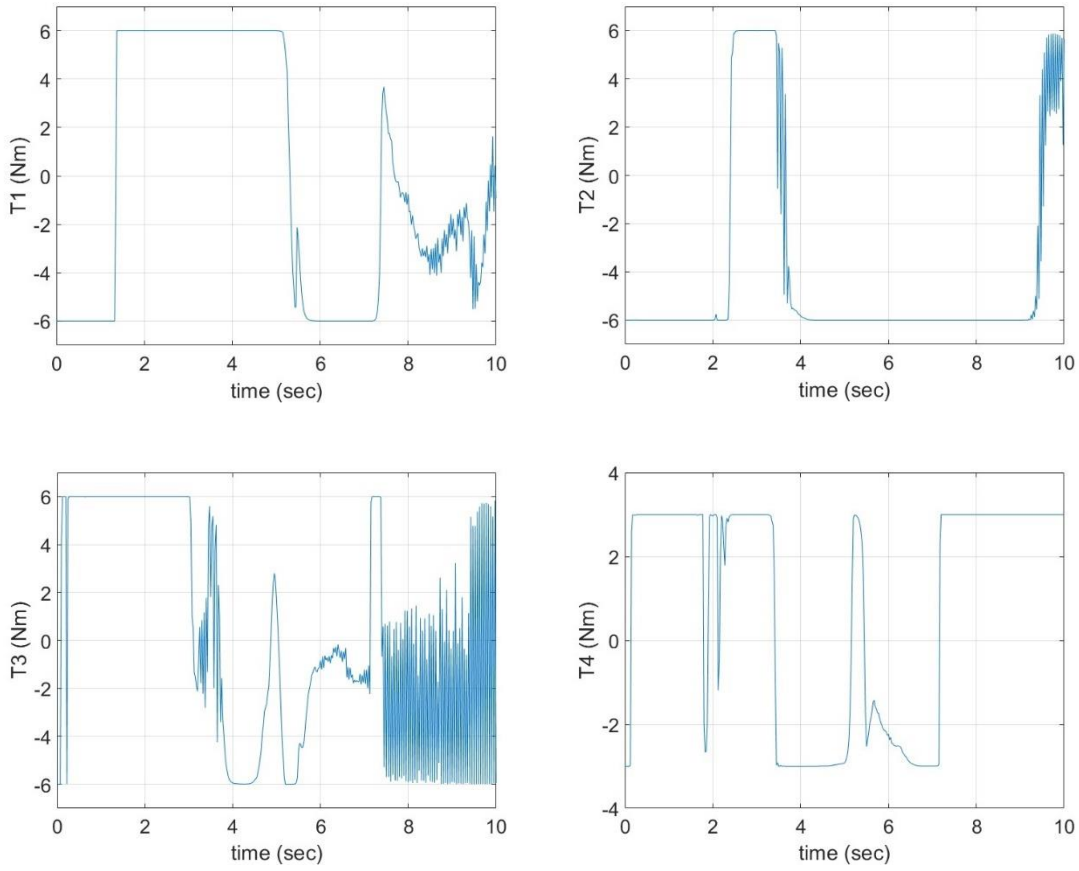


Figure 6.9 Case 1 test pose tracking errors (d, ϑ) over time



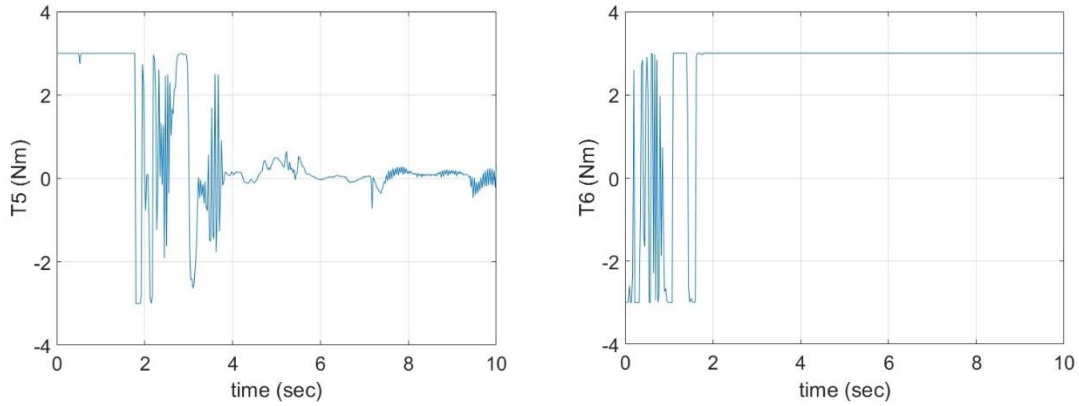


Figure 6.10 Case 1 test joint torques time histories

6.3.1.2 Random Joint Positions Case 2: with Velocity Constraints

This case is similar to case 2 in Section 5.4.2. The same reward function with the velocity constraint term is used. However, it is trained with using multi-scenario RL. The training result is shown in Figure 6.11.

The fully trained agent has the ability to achieve successful capture of the target pose from different initial joint positions with a 98% success rate. It can also achieve capture of the target pose while avoiding the obstacle with a 63% success rate. In addition to improving the motion using velocity constraint.

The minimum distance error achieved by the agent was of d = 0.0111m and orientation error of ϑ = 4.2993 degrees.

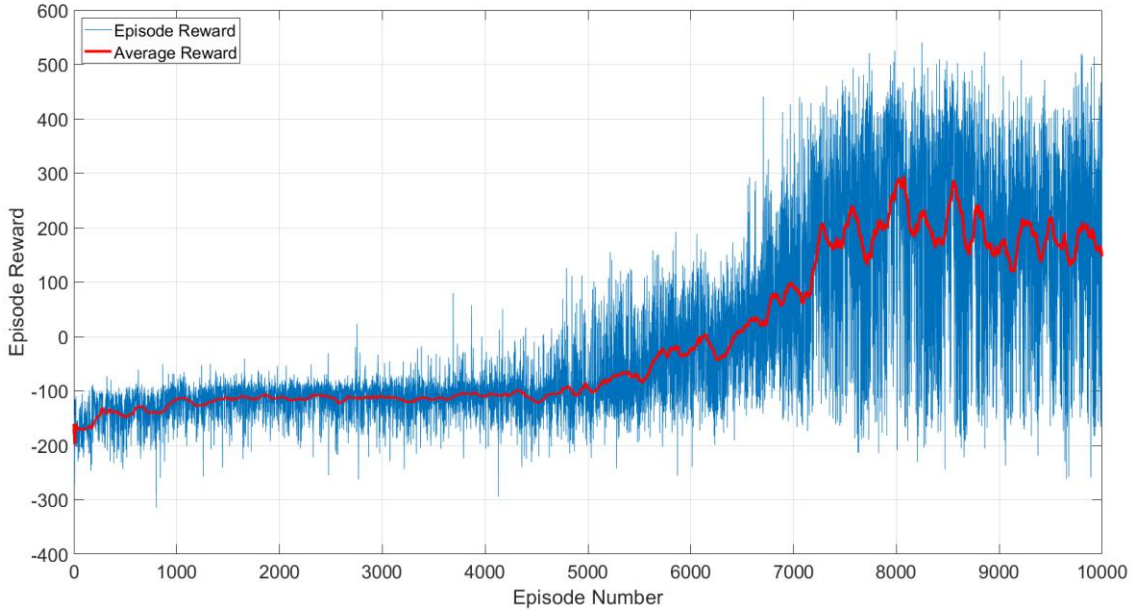


Figure 6.11 multi-scenario RL Case 2 Training result :10,000 episodes

For testing case 2 the initial joint positions are chosen randomly, see Table 6.4. Figure 6.12 shows the pose error between the EE and the target over time. The trained agent guides the EE towards the desired target pose range within approximately 8 seconds.

The agent achieves a minimum distance error of d = 0.0559 m and orientation error of ϑ = 6.3404 degrees. Figure 6.13 shows the time histories of actions (control torques) at all six joints as given by the agent.

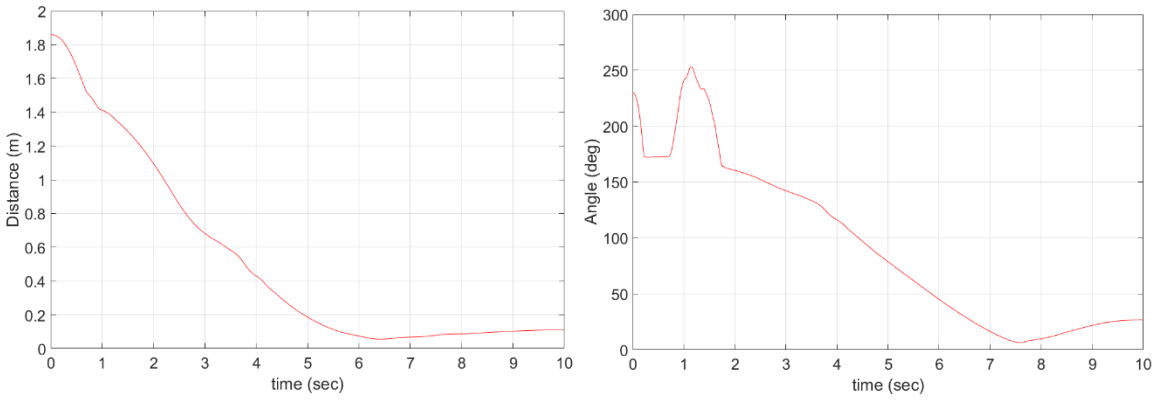
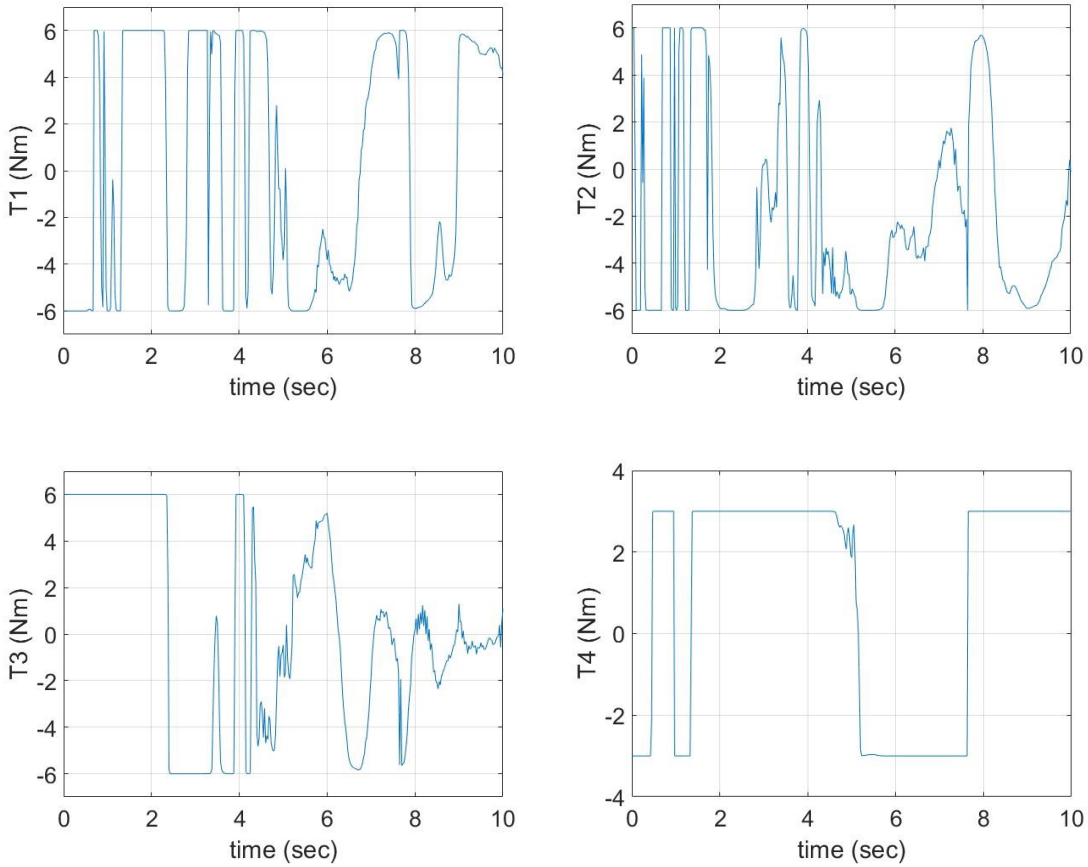


Figure 6.12　Case 2 test pose tracking errors (d, ϑ) over time



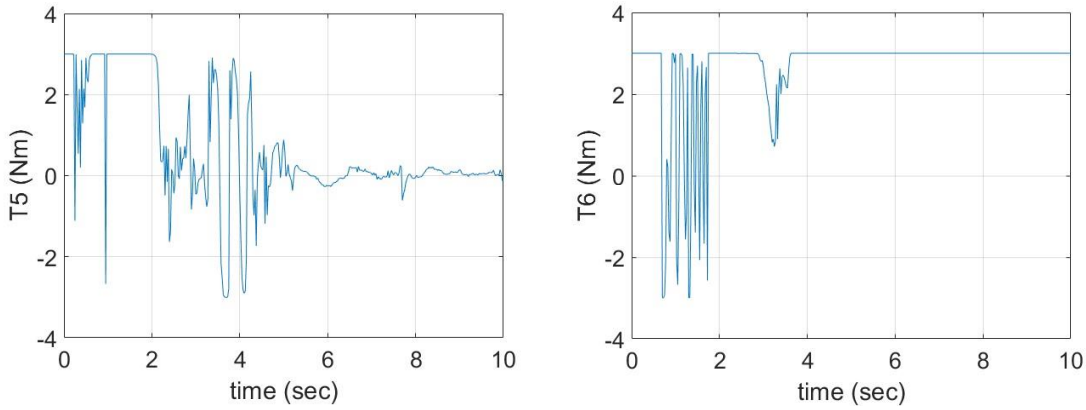


Figure 6.13 Case 2 test joint torques time histories

6.3.1.3 Random Joint Positions Case 3: with Noisy Observations

This case is similar to case 3 in Section 5.4.3. The same reward function with the velocity constraint term is used. However, it is trained with using multi-scenario RL and with noisy observations, such as target positions, orientations, and obstacle positions, The training result is shown in Figure 6.14. As revealed in previous sections, the addition of noisy observations required more episodes of training. Therefore, in case 3 the training was done for 15,000 episodes.

The fully trained agent can achieve successful capture of the target pose from different initial joint positions with an 87% success rate. It can also achieve capture of the target pose while avoiding the obstacle with a 50% success rate. In addition to improving the motion using velocity constraint and under noisy observations. The minimum distance error achieved by the agent

was of d = 0.0038m and orientation error of ϑ = 2.1337 degrees.

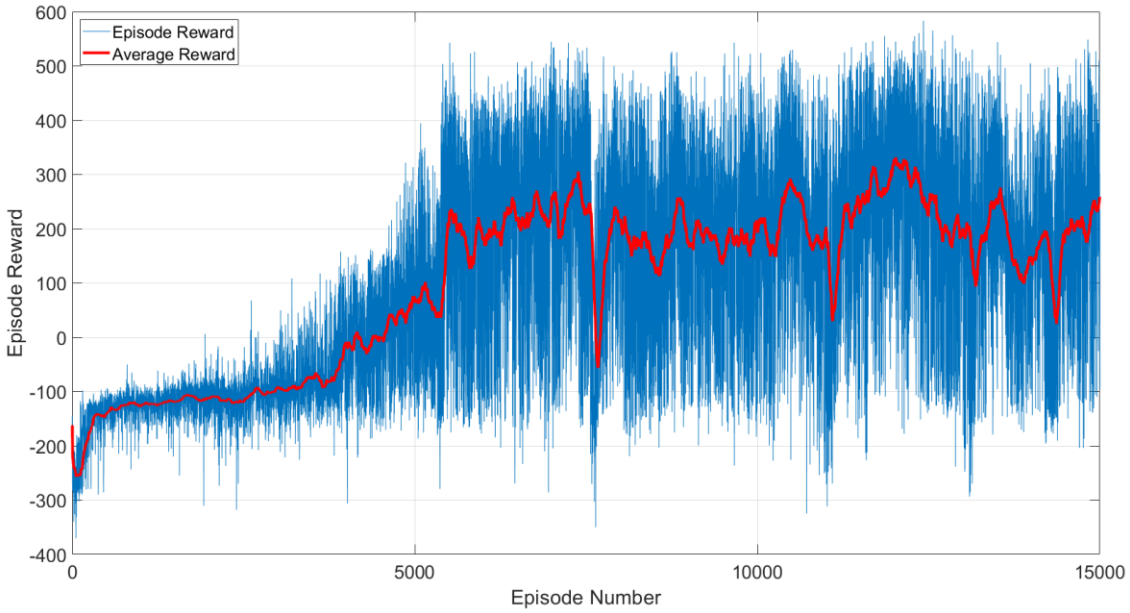


Figure 6.14 Multi-scenario RL Case 3 Training result :15,000 episodes

For testing case 3 the initial joint positions are chosen randomly, see Table 6.4. Figure 6.15 shows the pose error between the EE and the target over time. The trained agent guides the EE towards the desired target pose range within approximately 8 seconds.

The agent achieves a minimum distance error of d = 0.0254m and orientation error of ϑ = 5.8576 degrees. Figure 6.16 shows the time histories of actions (control torques) at all six joints as given by the agent.

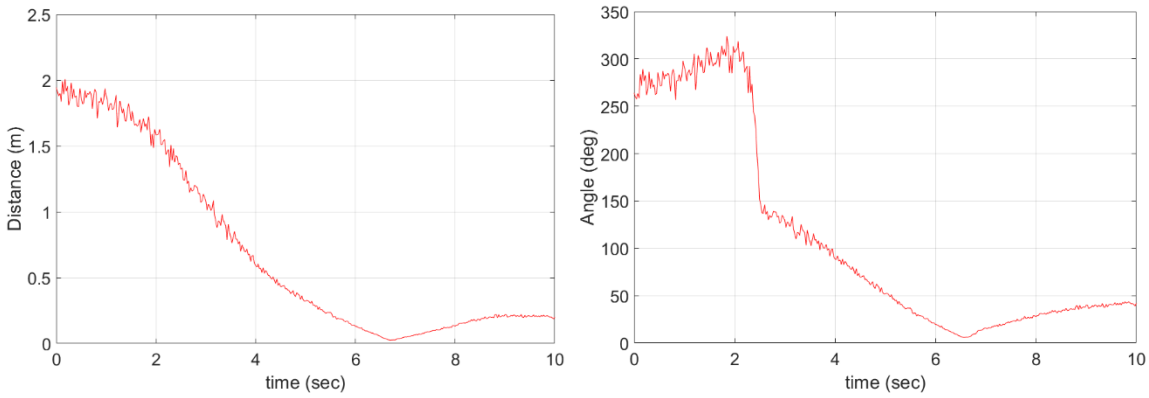
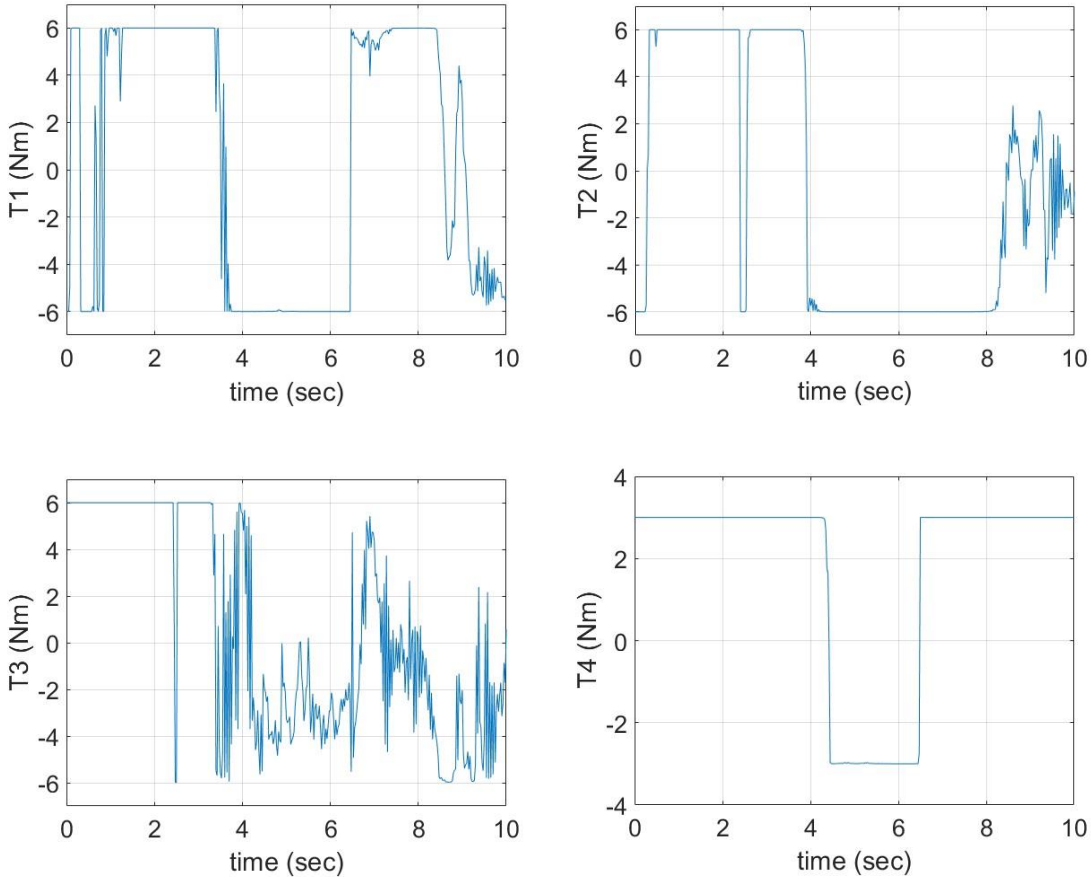


Figure 6.15 Case 3 test pose tracking errors (d, ϑ) over time



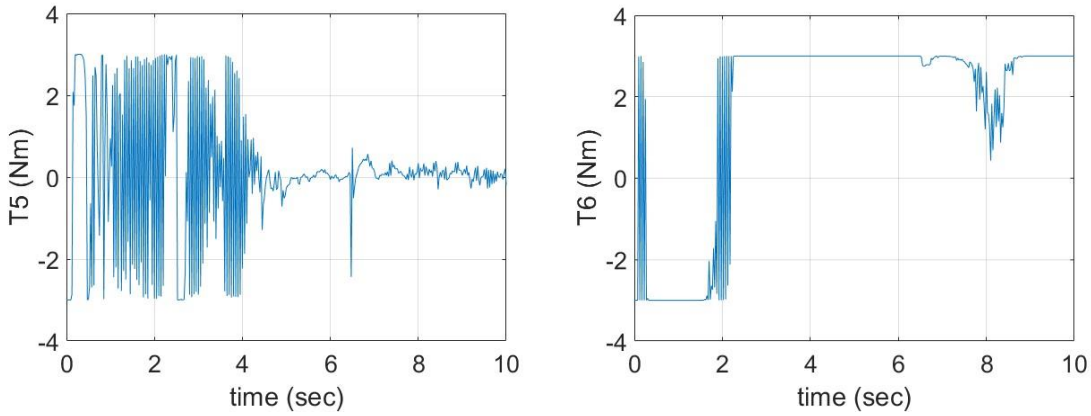


Figure 6.16 Case 3 test joint torques time histories

6.3.2 Random Initial Joint Positions & Obstacle Positions

The training in this section will be done on the same environment, and reward functions previously described in Section 5.4, with random initial joint positions, and randomly placed obstacles at the start of each episode within the ranges in shown in Table 6.5.

Table 6.5 Multi-scenario RL Initial Conditions 2

Parameter	Random Value for Training	For Testing		
		Case 1	Case 2	Case 3
Joint1	between (60 to 120) deg	97.323	66.405	84.969
Joint2	between (0 to 30) deg	10.528	19.612	25.257
Joint3	between (0 to 30) deg	15.397	14.825	24.987
Joint4	between (0 to 30) deg	12.054	23.371	7.693
Joint5	between (-30 to 30) deg	-25.442	12.902	6.807

Joint6	between (-30 to 30) deg	-15.605	24.223	4.934
Target x	1.6 m	1.6 m	1.6 m	1.6 m
Target y	0.0 m	0.0 m	0.0 m	0.0 m
Target z	1.0 m	1.0 m	1.0 m	1.0 m
Obstacle x	Between (0.5 to 1.0) m	0.562	0.945	0.770
Obstacle y	Between (-1.0 to 1.0) m	-0.632	-0.332	0.739
Obstacle z	Between (0.5 to 2.5) m	0.979	1.897	1.029

6.3.2.1 Random Joint & Obstacle Positions Case 1: without Velocity Constraints

This case is similar to case 1 in Section 5.4.1. The same reward function without the velocity constraint term is used. However, it is trained with using multi-scenario RL. The training result is shown in Figure 6.17.

The fully trained agent has the ability to achieve successful capture of the target pose from different initial joint positions with a 97% success rate. It can also achieve capture of the target pose while avoiding the randomly placed obstacles with an 86% success rate. The minimum distance error achieved by the agent was of d = 0.0061m and orientation error of ϑ = 3.0358 degrees.

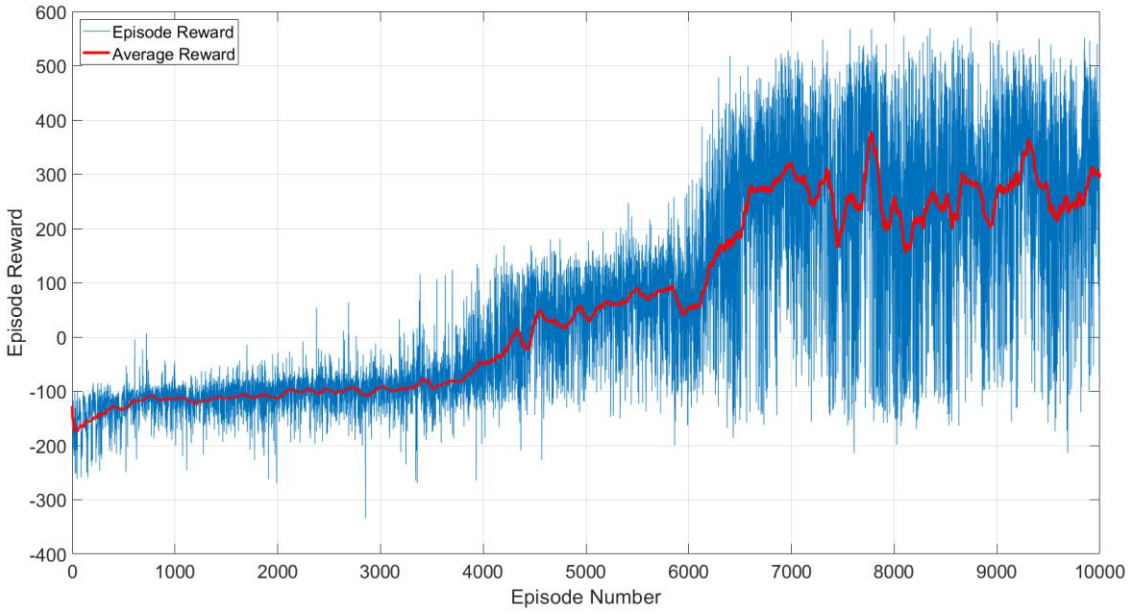


Figure 6.17 Multi-scenarioRL Case 1 Training result :10,000 episodes

For testing case 1 the initial joint positions are chosen randomly, see Table 6.5. Figure 6.18 shows the pose error between the EE and the target over time. The trained agent guides the EE towards the desired target pose range within approximately 6 seconds.

The agent achieves a minimum distance error of d = 0.0358 m and orientation error of ϑ = 6.371 degrees. Figure 6.19 shows the time histories of actions (control torques) at all six joints as given by the agent.

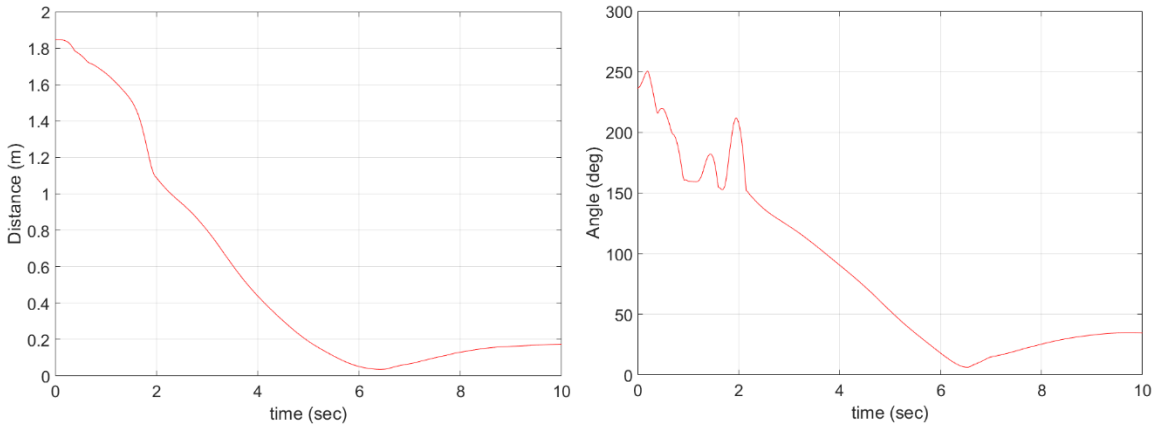
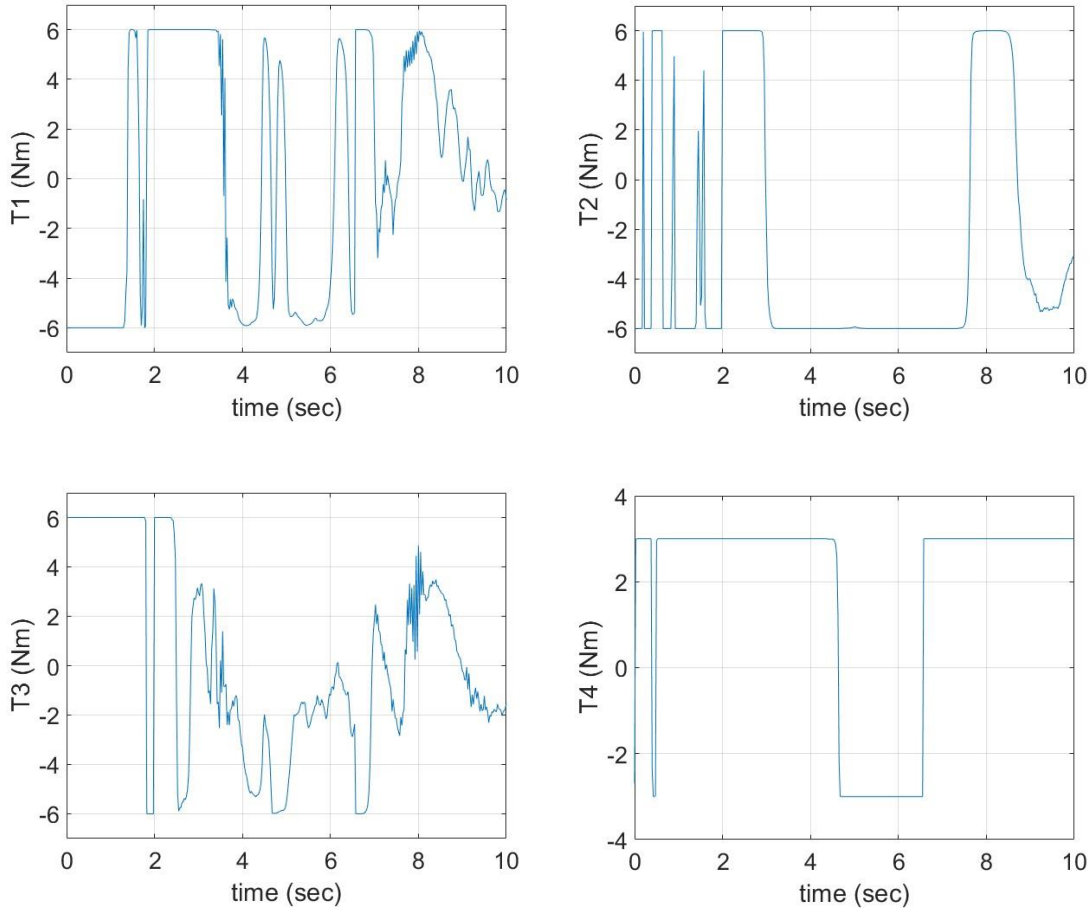


Figure 6.18 Case 1 test pose tracking errors (d, ϑ) over time



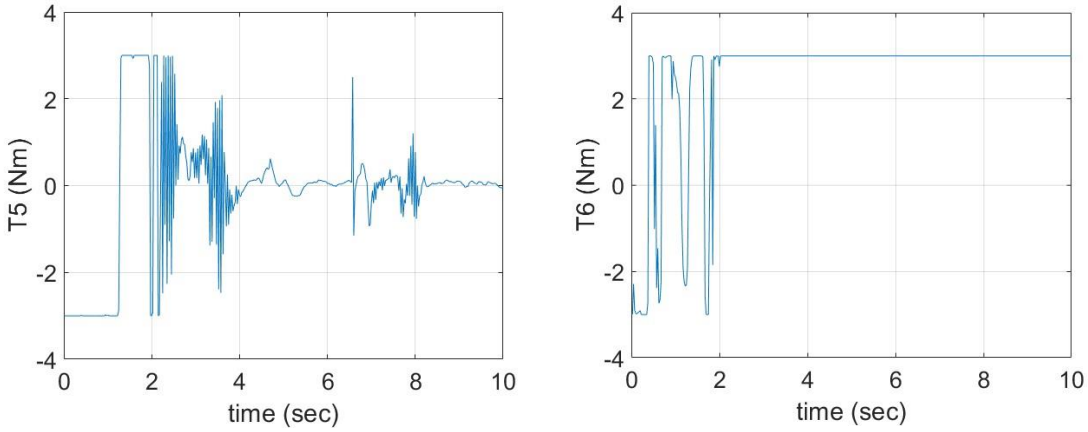


Figure 6.19 Case 1 test joint torques time histories.

6.3.2.2 Random Joint & Obstacle Positions Case 2: with Velocity Constraints

This case is similar to case 2 in Section 5.4.2. The same reward function with the velocity constraint term is used. However, it is trained with using multi-scenario RL. The training result is shown in Figure 6.20.

The fully trained agent has the ability to achieve successful capture of the target pose from different initial joint positions with a 97% success rate. It can also achieve capture of the target pose while avoiding the randomly placed obstacles with an 88% success rate. In addition to improving the motion using velocity constraint. The minimum distance error achieved by the agent was of d = 0.0067m and orientation error of ϑ = 2.0934 degrees.

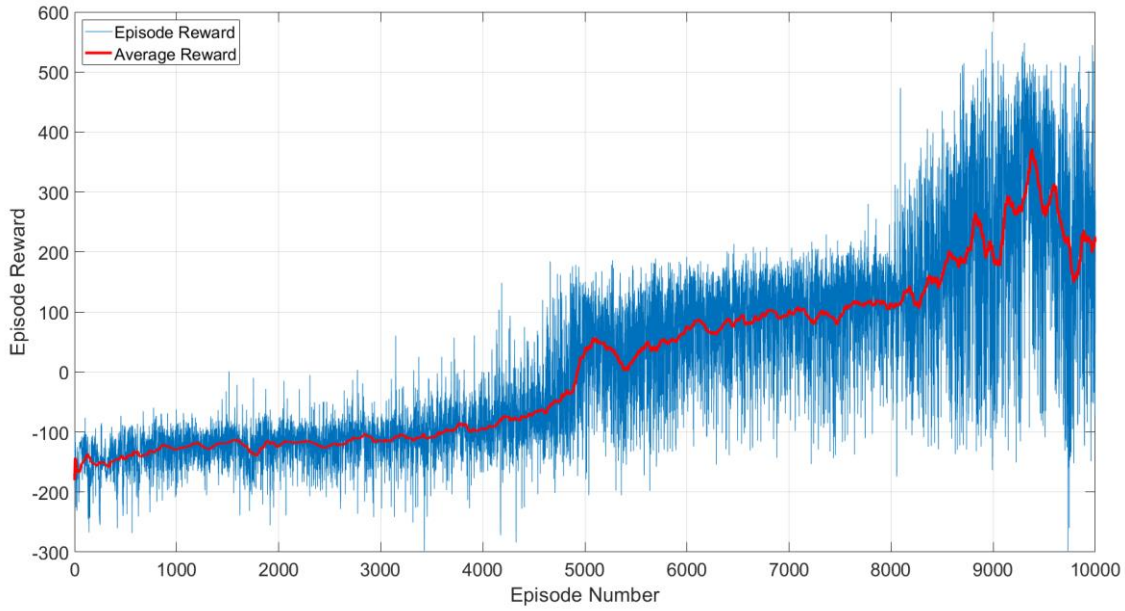


Figure 6.20 Multi-scenario RL Case 2 Training result :10,000 episodes

For testing case 2 the initial joint positions are chosen randomly, see Table 6.5. Figure 6.21 shows the pose error between the EE and the target over time. The trained agent guides the EE towards the desired target pose range within approximately 5 seconds.

The agent achieves a minimum distance error of d = 0.0620m and orientation error of ϑ = 8.2814 degrees. Figure 6.22 shows the time histories of actions (control torques) at all six joints as given by the agent.

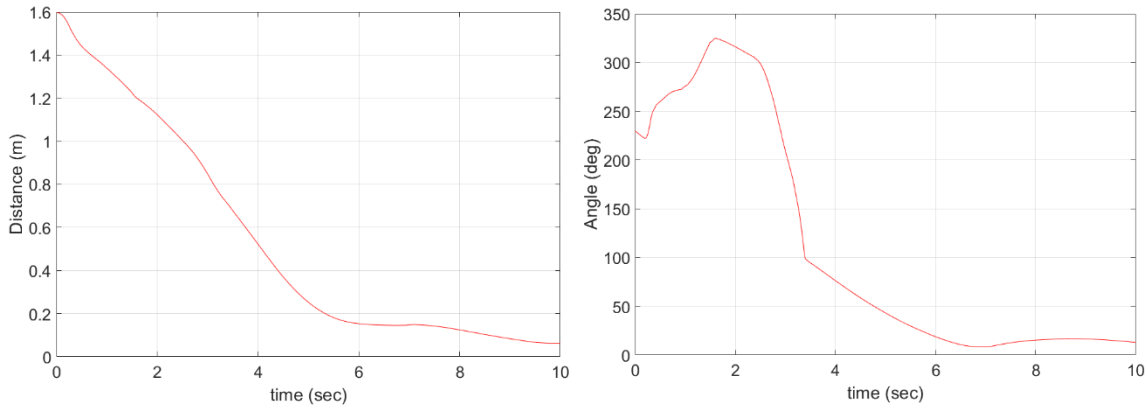
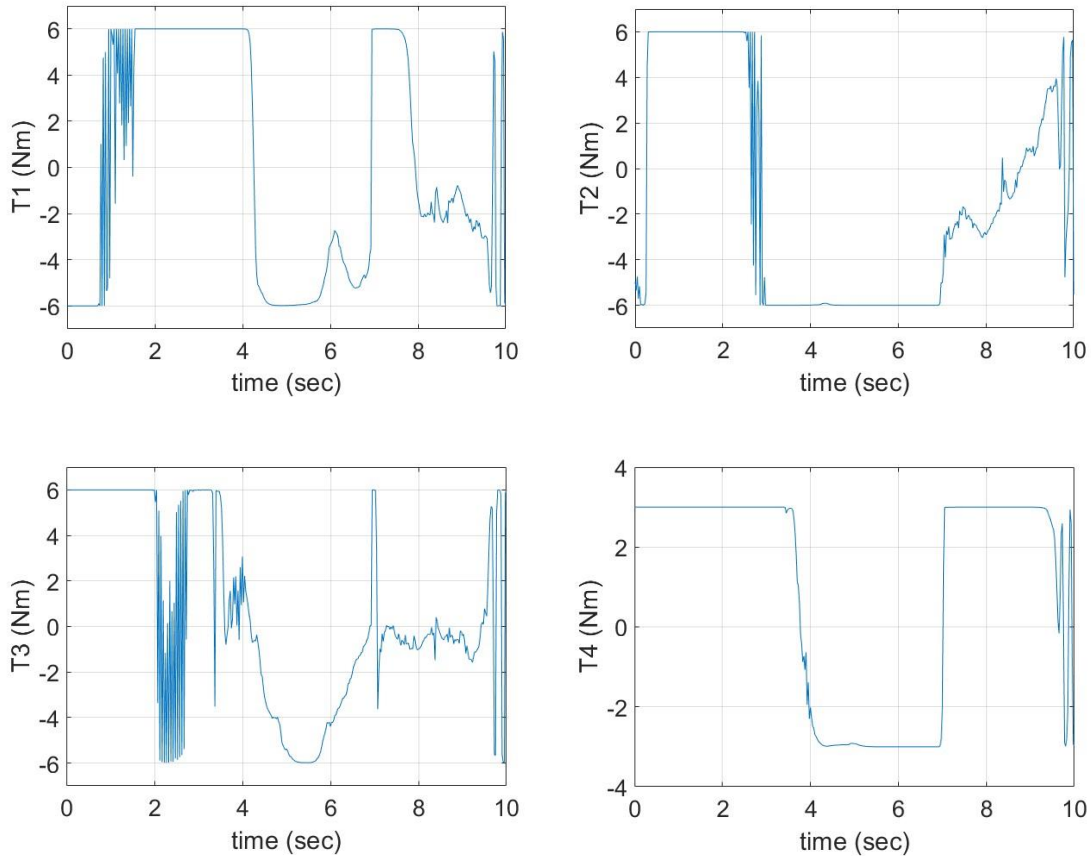


Figure 6.21 Case 2 test pose tracking errors (d, ϑ) over time



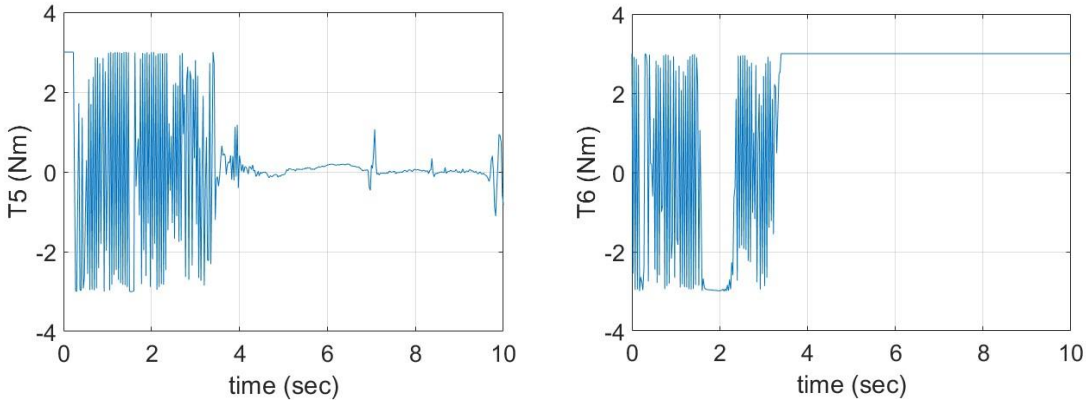


Figure 6.22 Case 2 test joint torques time histories

6.3.2.3 Random Joint & Obstacle Positions Case 3: with Noisy Observations

This case is similar to case 3 in Section 5.4.3. The same reward function with the velocity constraint term is used. However, it is trained with using multi-scenario RL and with noisy observations, such as target positions, orientations, and obstacle positions, The training result is shown in Figure 6.23. As revealed in previous sections, the addition of noisy observations required more episodes of training. Therefore, in case 3 the training was done for 15,000 episodes.

The fully trained agent has the ability to achieve successful capture of the target pose from different initial joint positions with a 100% success rate. It can also achieve capture of the target pose while avoiding the randomly placed obstacles with an 89% success rate. In addition to improving the motion

using velocity constraint and under noisy observations. The minimum distance error achieved by the agent was of d = 0.0072m and orientation error of ϑ = 2.5995 degrees.

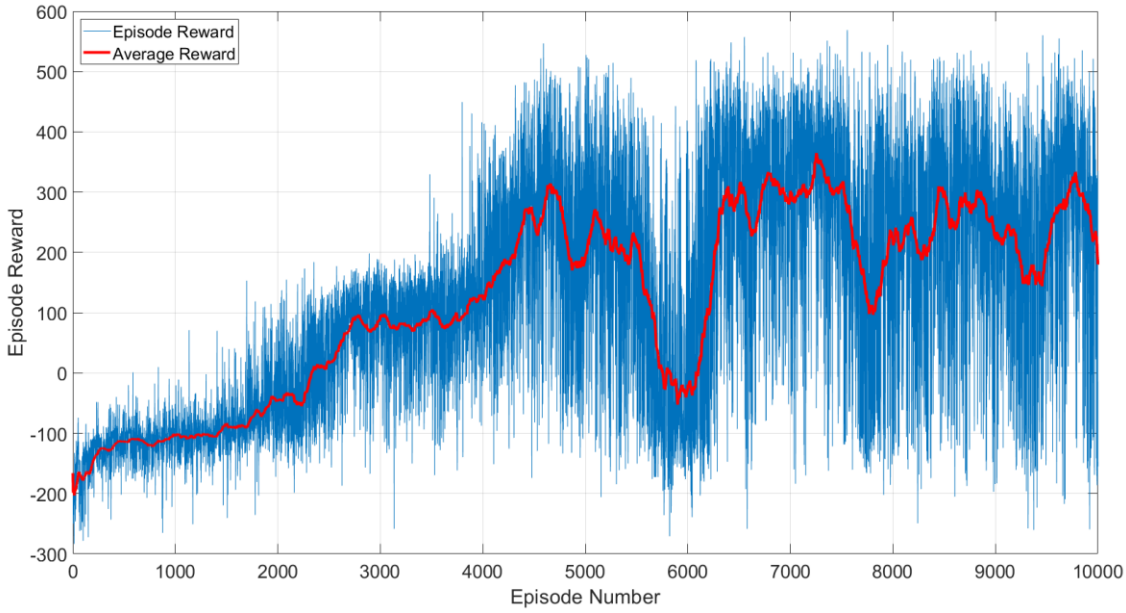


Figure 6.23 Multi-scenario RL Case 3 Training result :15,000 episodes

For testing case 3 the initial joint positions are chosen randomly, see Table 6.5. Figure 6.24 shows the pose error between the EE and the target over time. The trained agent guides the EE towards the desired target pose range within approximately 8 seconds.

The agent achieves a minimum distance error of d = 0.0567m and orientation error of ϑ = 3.9955 degrees. Figure 6.25 shows the time histories of actions (control torques) at all six joints as given by the agent.

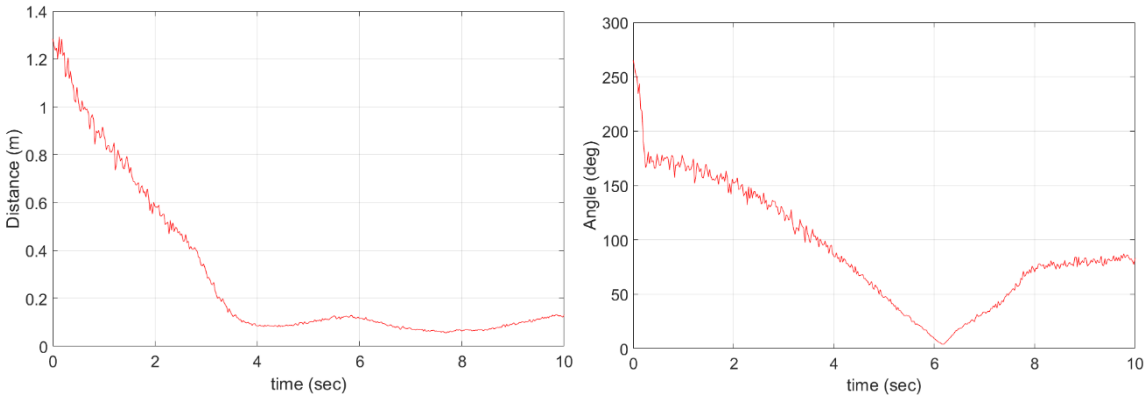
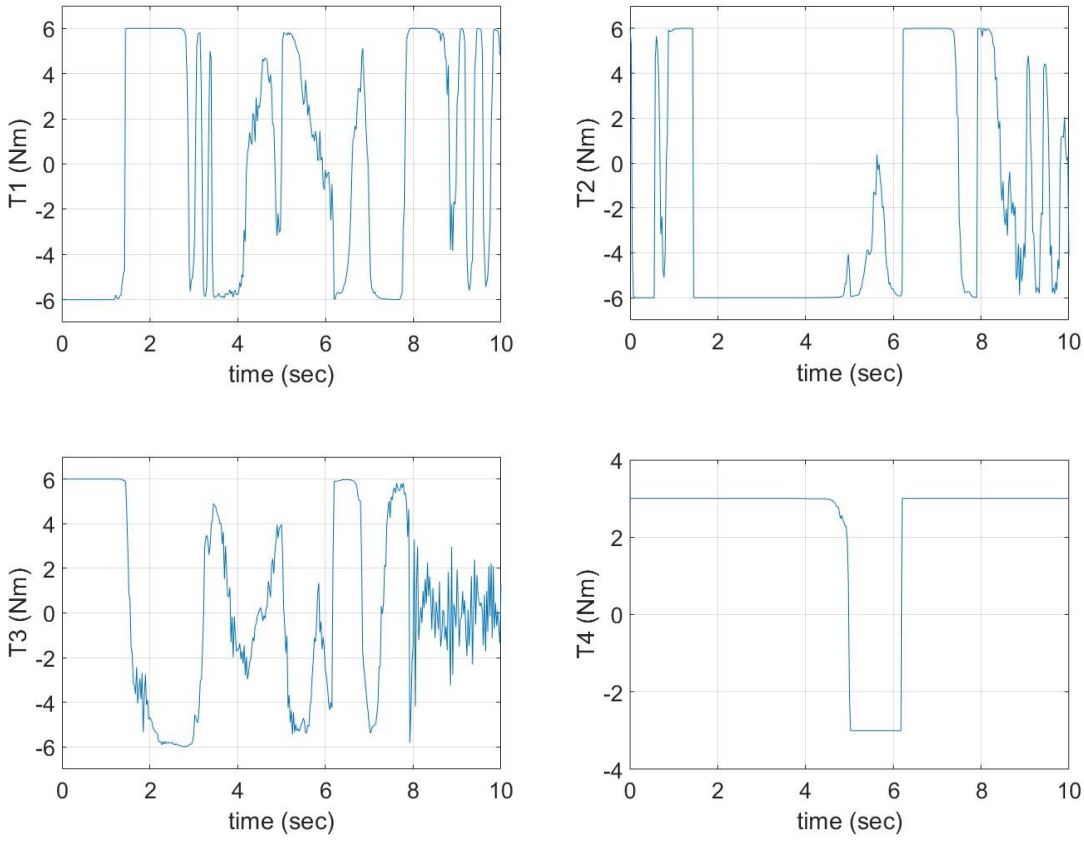


Figure 6.24 Case 3 test pose tracking errors (d, ϑ) over time



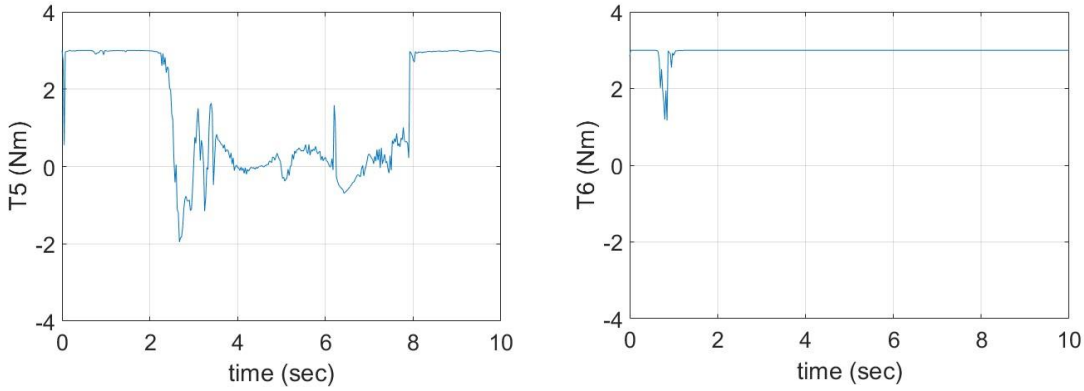


Figure 6.25 Case 3 test joint torques time histories

6.3.3 Random Initial Joint Positions, Obstacle Positions & Target Positions

The training in this section will be done on the same environment, and reward functions previously described in Section 5.4, with random initial joint positions, randomly placed targets, and random obstacle locations at the start of each episode within the ranges in shown in Table 6.6.

Table 6.6 Multi-scenario RL Initial Conditions 3

Parameter	Random Value for Training	For Testing		
		Case 1	Case 2	Case 3
Joint1	between (60 to 120) deg	105.304	112.435	105.68
Joint2	between (0 to 30) deg	11.321	8.108	18.932
Joint3	between (0 to 30) deg	6.480	6.253	2.696
Joint4	between (0 to 30) deg	3.712	16.949	2.425
Joint5	between (-30 to 30) deg	26.958	8.418	16.634

Joint6	between (-30 to 30) deg	-10.346	-4.978	24.308
Target x	between (1.6 to 2.1) m	1.984	1.661	1.749
Target y	between (-0.5 to 0.5) m	-0.103	-0.009	-0.453
Target z	between (1.0 to 1.5) m	1.404	1.4264	1.252
Obstacle x	Between (0.5 to 1.0) m	0.835	0.602	0.766
Obstacle y	Between (-1.0 to 1.0) m	-0.122	0.895	-0.781
Obstacle z	Between (0.5 to 2.5) m	2.167	0.664	2.151

6.3.3.1 Random Joint, Obstacle & Target Positions Case 1: without Velocity Constraints

This case is similar to case 1 in Section 5.4.1. The same reward function without the velocity constraint term is used. However, it is trained with using multi-scenario RL. The training result is shown in Figure 6.26.

The fully trained agent has the ability to achieve successful capture of the randomly placed target pose from different initial joint positions with an 86% success rate. It can also achieve capture of the target pose while avoiding the randomly placed obstacles with an 82% success rate.

The minimum distance error achieved by the agent was of d = 0.0081m and orientation error of ϑ = 3.1301 degrees.

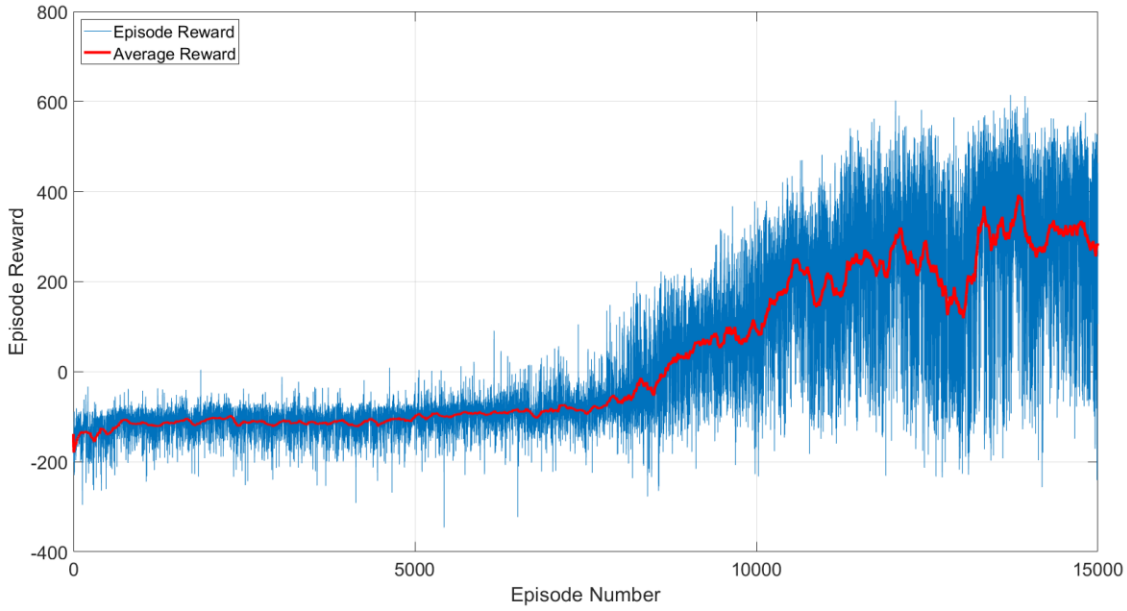


Figure 6.26 Multi-scenario RL Case 1 Training result :15,000 episodes

For testing case 1 the initial joint positions are chosen randomly, see Table 6.6. Figure 6.27 shows the pose error between the EE and the target over time. The trained agent guides the EE towards the desired target pose range within approximately 6 seconds.

The agent achieves a minimum distance error of d = 0.0275m and orientation error of ϑ = 8.3201 degrees. Figure 6.28 shows the time histories of actions (control torques) at all six joints as given by the agent.

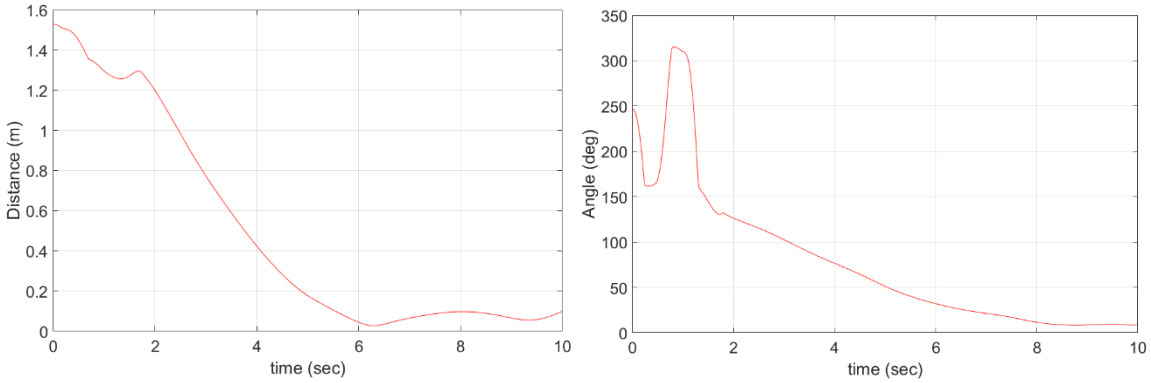
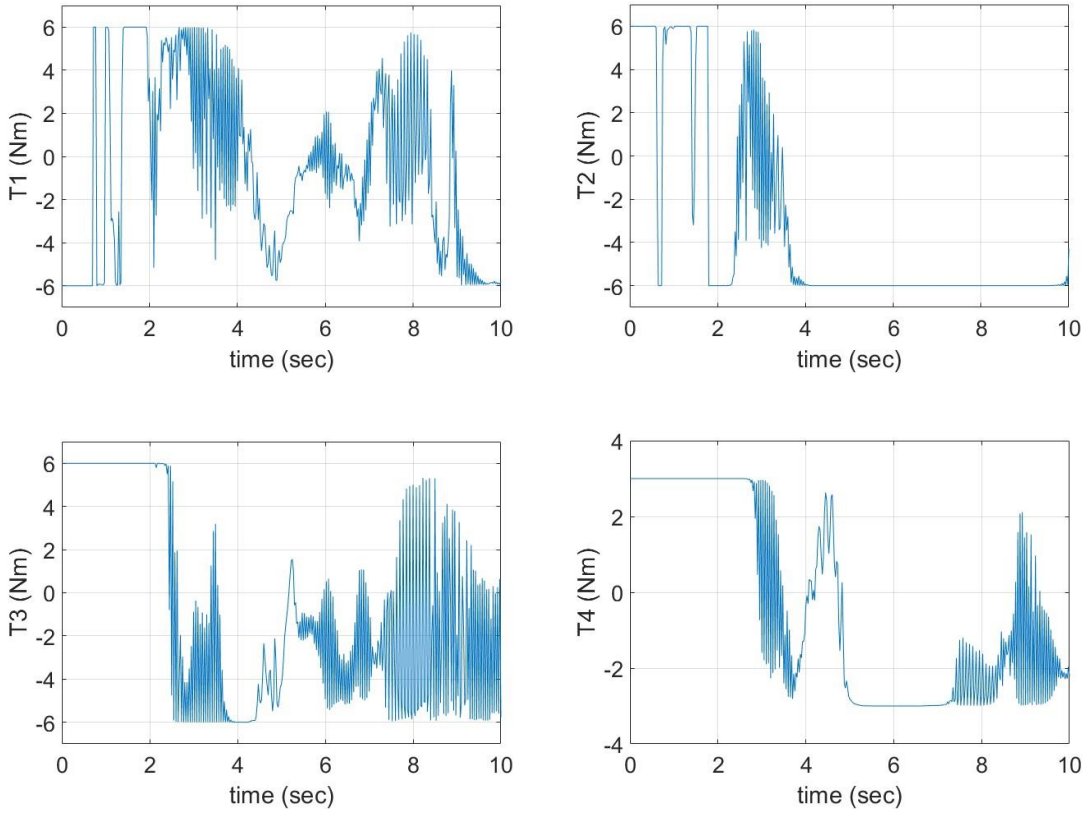


Figure 6.27 Case 1 test pose tracking errors (d, ϑ) over time



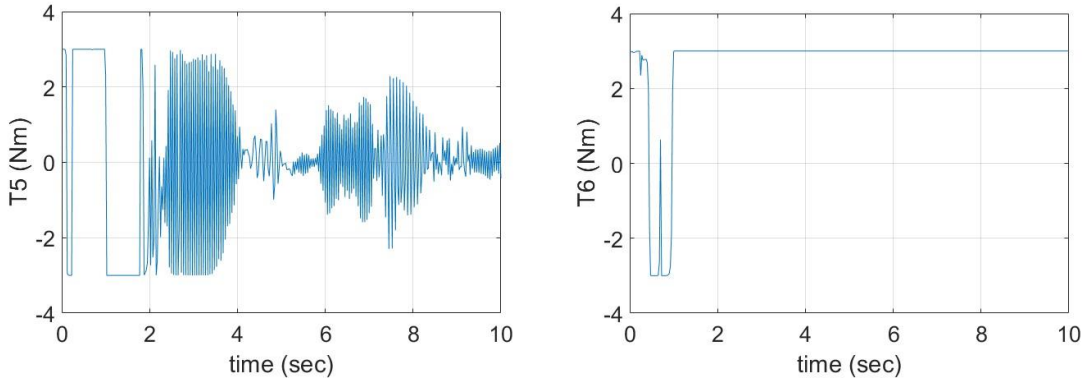


Figure 6.28 Case 1 test joint torques time histories.

6.3.3.2 Random Joint, Obstacle & Target Positions Case 2: with Velocity Constraints

This case is similar to case 2 in Section 5.4.2. The same reward function with the velocity constraint term is used. However, it is trained with using multi-scenario RL. The training result is shown in Figure 6.29.

The fully trained agent has the ability to achieve successful capture of the randomly placed target pose from different initial joint positions with an 97% success rate. It can also achieve capture of the target pose while avoiding the randomly placed obstacles with an 86% success rate. In addition to improving the motion using velocity constraint.

The minimum distance error achieved by the agent was of d = 0.0094m and orientation error of ϑ = 6.98 degrees.

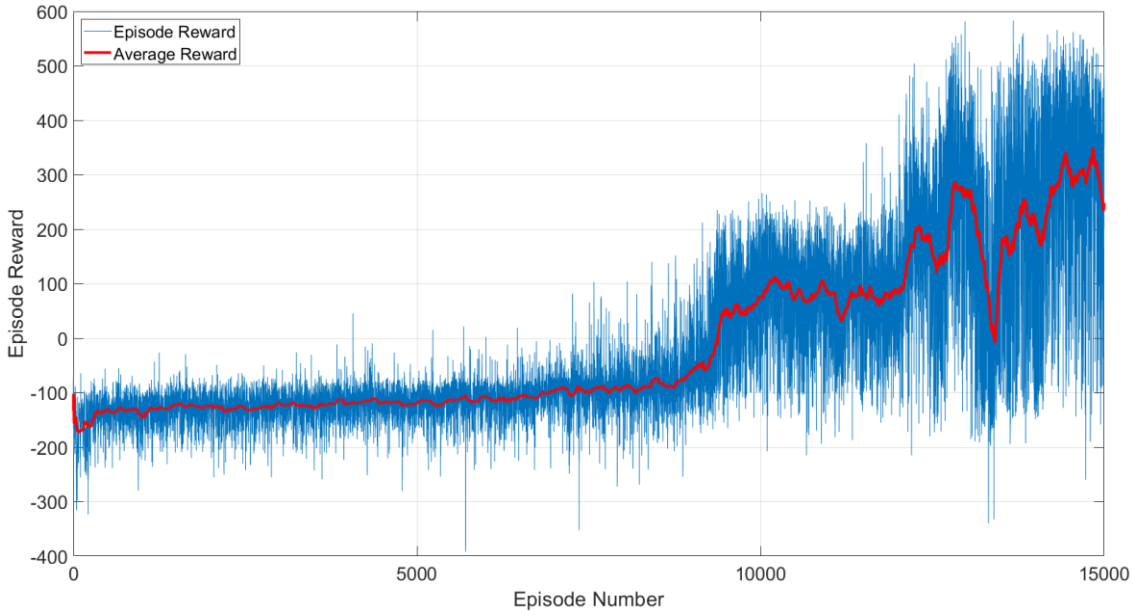


Figure 6.29 Multi-scenario RL Case 2 Training result :15,000 episodes

For testing case 2 the initial joint positions are chosen randomly, see Table 6.6. Figure 6.30 shows the pose error between the EE and the target over time. The trained agent guides the EE towards the desired target pose range within approximately 5 seconds.

The agent achieves a minimum distance error of d = 0.0203m and orientation error of ϑ = 14.842 degrees. Figure 6.31 shows the time histories of actions (control torques) at all 6 joints as given by the agent.

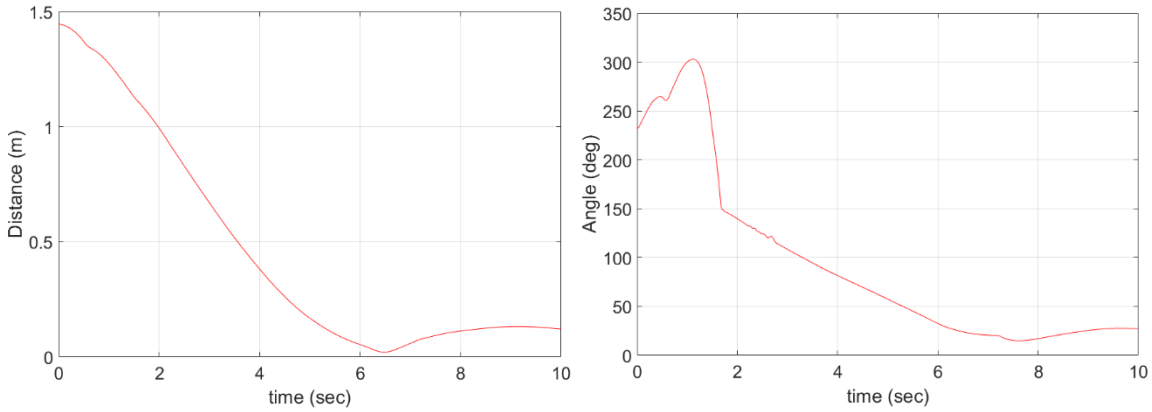
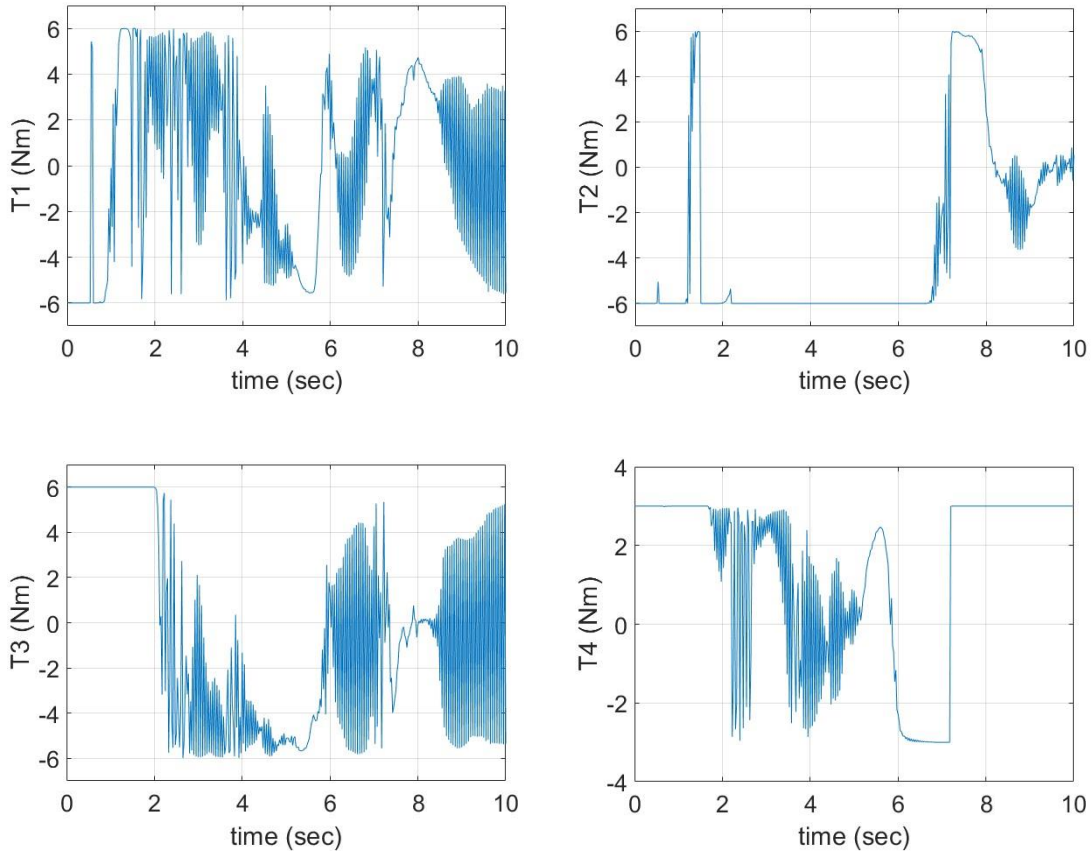


Figure 6.30 Case 2 test pose tracking errors (d, ϑ) over time



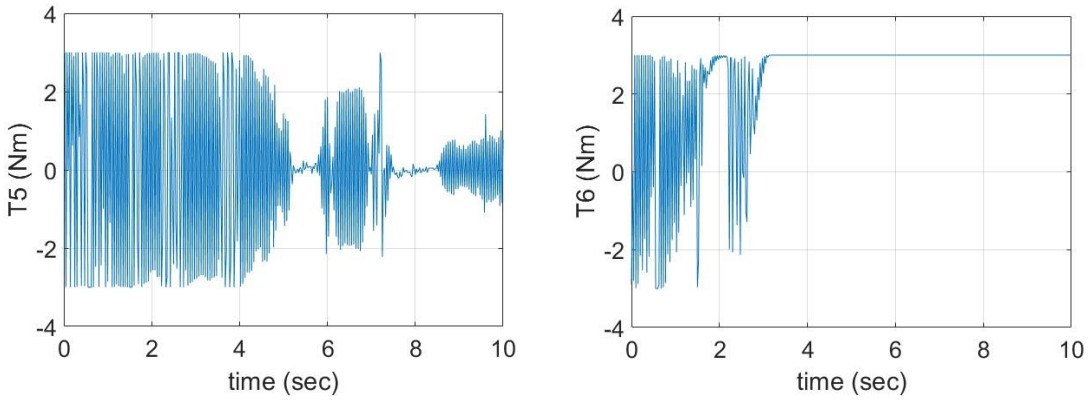


Figure 6.31 Case 2 test joint torques time histories

6.3.3.3 Random Joint, Obstacle & Target Positions Case 3: with Noisy Observations

This case is similar to case three in Section 5.4.3. The same reward function with the velocity constraint term is used. However, it is trained with using multi-scenario RL and with noisy observations, such as target positions, orientations, and obstacle positions, The training result is shown in Figure 6.32.

The fully trained agent has the ability to achieve successful capture of the randomly placed target pose from different initial joint positions with an 75% success rate. It can also achieve capture of the target pose while avoiding the randomly placed obstacles with an 60% success rate.

In addition to improving the motion using velocity constraint and under noisy observations. The minimum distance error achieved by the agent was of d = 0.0085m and orientation error of ϑ = 2.3172 degrees.

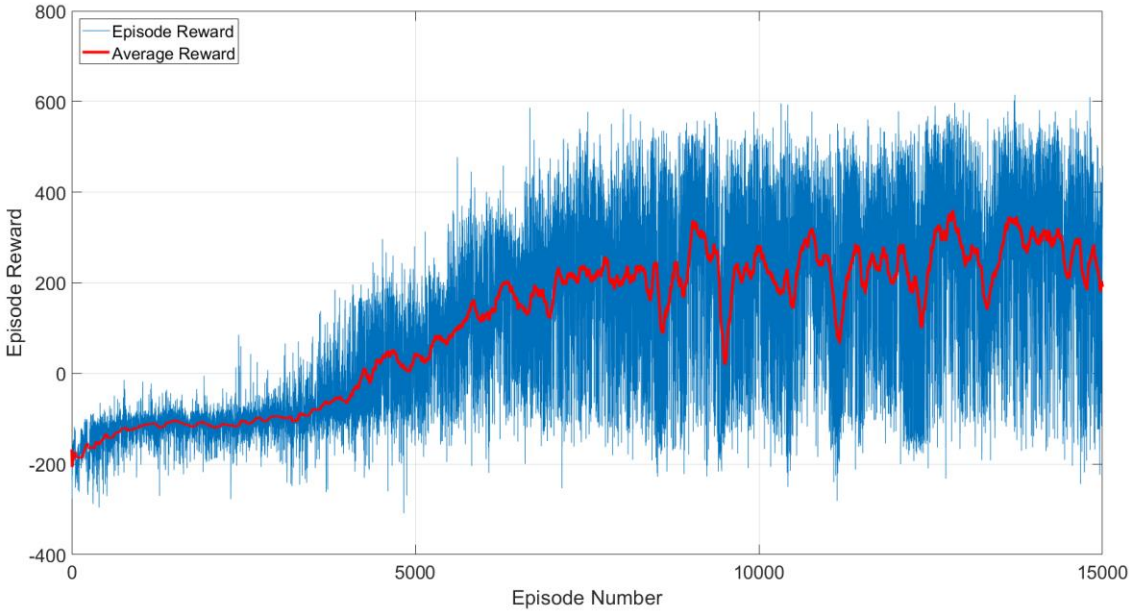


Figure 6.32 Multi-scenario RL Case 3 Training result :15,000 episodes

For testing case 3 the initial joint positions are chosen randomly, see Table 6.6. Figure 6.33 shows the pose error between the EE and the target over time. The trained agent guides the EE towards the desired target pose range within approximately 8 seconds.

The agent achieves a minimum distance error of d = 0.0179m and orientation error of ϑ = 18.3973 degrees. Figure 6.34 shows the time histories of actions (control torques) at all six joints as given by the agent.

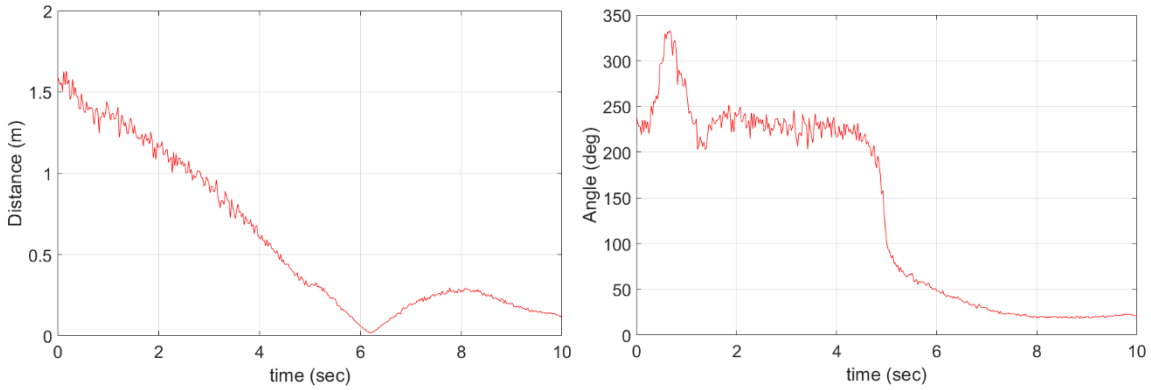
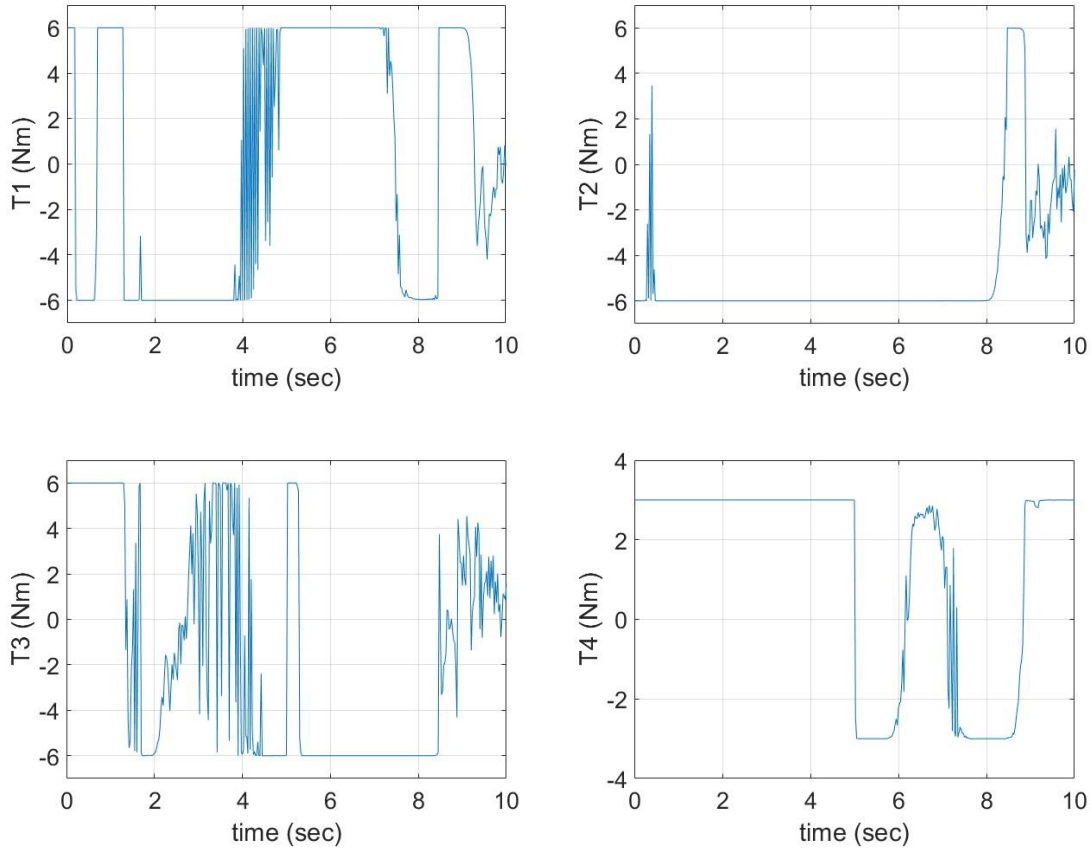


Figure 6.33 Case 3 test pose tracking errors (d, ϑ) over time



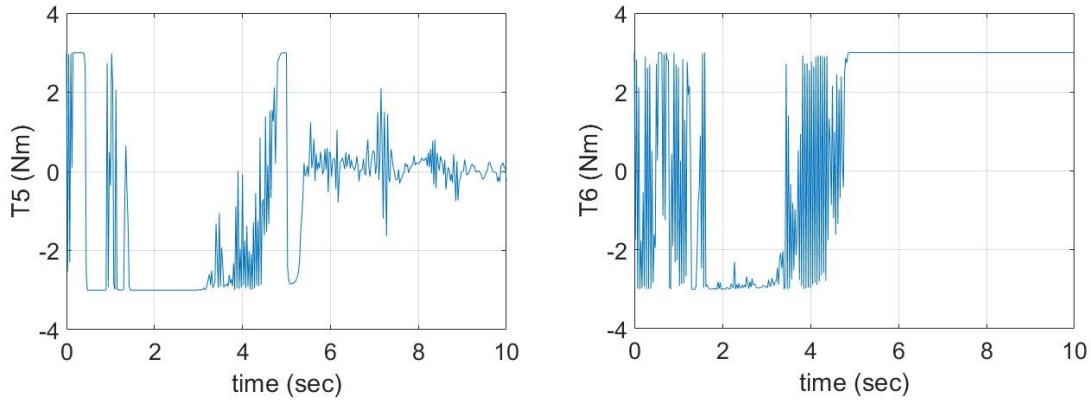


Figure 6.34 Case 3 test joint torques time histories.

All the multi-scenario RL agents trained in this section are summarized in Table 6.7 below.

Table 6.7 Multi-scenario RL Training Summary

	random joints	random obstacle	random target	success rate-capture	capture + avoid obstacle	min d	min ϑ
Normal	✓	-	-	94%	40%	0.0031	4.4
VC	✓	-	-	98%	63%	0.011	4.3
Noise	✓	-	-	87%	50%	0.0038	2.23
Normal	✓	✓	-	97%	86%	0.0061	3.035
VC	✓	✓	-	97%	88%	0.0067	2.093
Noise	✓	✓	-	100%	89%	0.0072	2.6
Normal	✓	✓	✓	86%	82%	0.008	3.13
VC	✓	✓	✓	97%	86%	0.0094	6.98
Noise	✓	✓	✓	75%	60%	0.0085	2.317

Chapter 7 EXPERIMENTAL TESTING AND VALIDATION

Summary: This chapter presents the development of 6DOF hardware-in-the-loop robotic experimental system. This chapter includes parts of the experimental results that have been published in the reference papers B, E, and H.

7.1 Introduction

The 6DOF hardware-in-the-loop robotic experimental system previously detailed in Section 3.2 is built and tested. Our initial experimental results in computer vision, sensing capabilities, and motion control are described in this chapter. The testbed is set up to examine the concept of debris capture by space robotic manipulators. The main scope is to validate the feasibility of the tumbling target maneuvers, the proposed space robotic control strategy in order to achieve soft capture, and the different sensory capabilities .

7.2 Computer Vision

The tracking of the target is done using AI computer vision – Yolo V5 [90]. The computer vision algorithm was trained to track the relative motion of the target's position and orientation with a hand-eye configuration using the Intel depth camera (D435). The algorithm learns to identify the target position

first, then the attitude by looking at three features of the target: nozzle, coupling ring, and solar panel from various angles. After implementing the trained algorithm, the camera successfully tracks the target features as shown in Figure 7.1.

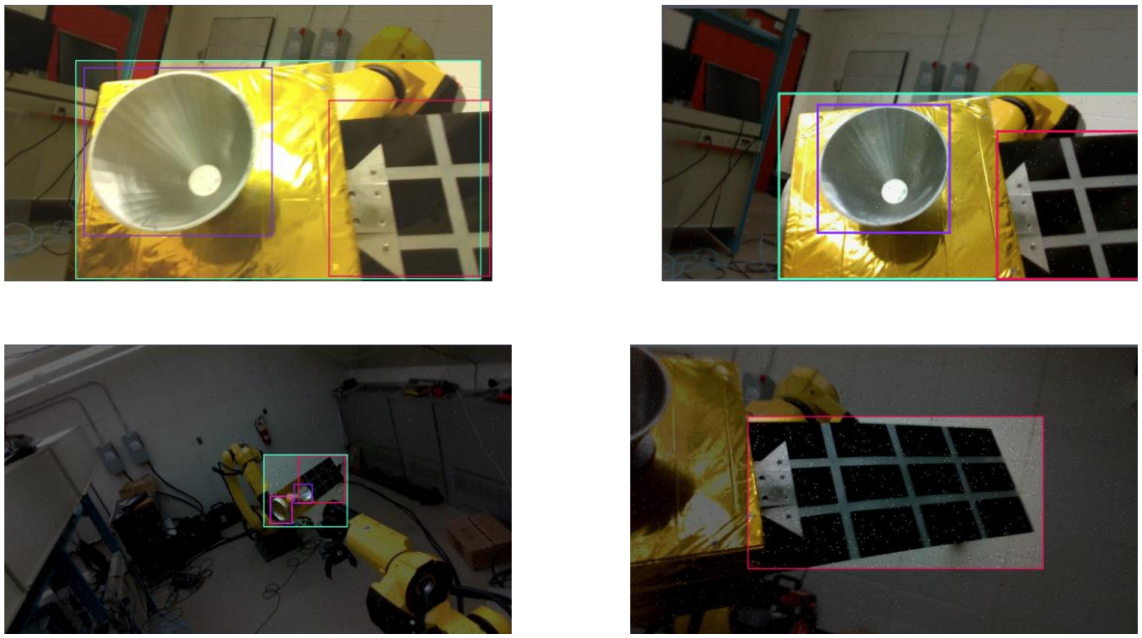


Figure 7.1 Computer vision bounding box and pose estimation

To verify the computer vision system's tracking capabilities, the target satellite was moved away from the camera in its optical axis by 0.5m, downward by 0.5m, and sideway by 0.5m. While the camera tracks the target and estimates its relative position. In this case, there was no rotation of the target. The position control accuracy of the Fanuc robot is 0.02mm, and the trajectory of the robot's end effector is treated as the ground truth.

Figure 7.3 shows the position tracking error of the Yolo algorithm. It

shows that the position error is minimal when the target is moved away along the camera's optical axis. As the target moves away from the camera's optical axis, the error increases to 3cm, which is about 6%. This is because the nozzle appears skewed in the camera's view, and the center of the frame that crops the nozzle, which is the position of the nozzle estimated by Yolo, is no longer the center of the nozzle, which is the actual position of the nozzle. Currently, the Yolo algorithm cannot solve this problem. A future study is needed to address this problem.

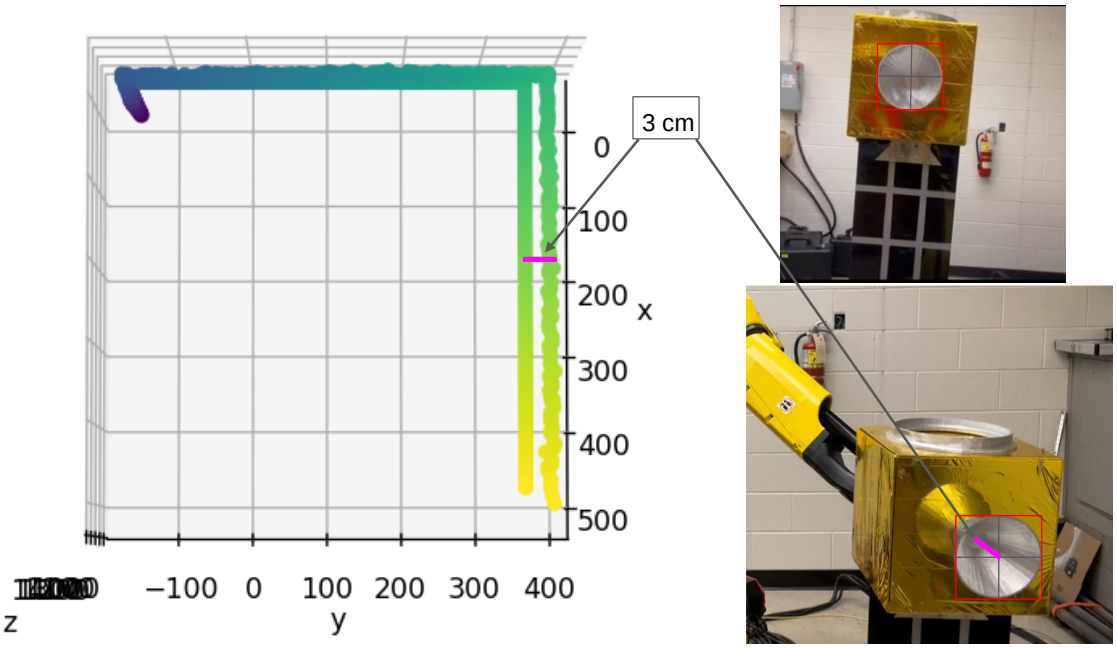


Figure 7.2 Computer vision tracking result

7.3 ATI Force/Torque Sensing and Gravity Compensation

The ATI force/load sensor attached between the physical robot's EE and

Mock-up satellite can sense the values before and after any force is applied to the satellite, as shown in Figure 7.3 . After the gripper is optimally placed during the pre-grasping phase, the gripper is controlled to close and capture the target satellite. When this occurs the ATI force sensor will record the values and move the target accordingly. This will mimic a free-floating target in space.

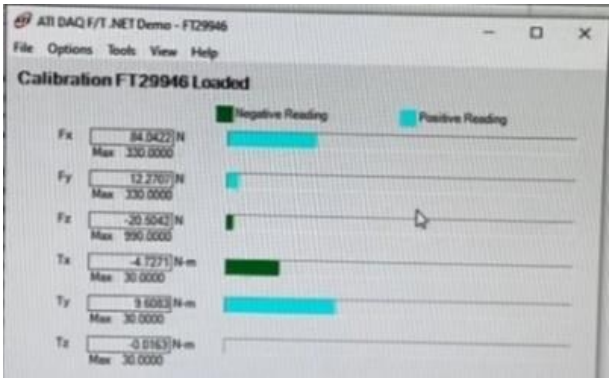


Figure 7.3 ATI force/load sensor app using Matlab

This is done by initially measuring the forces & moments caused by gravity (ATI measurements before disturbance):

$$F_0 = R.W \tag{7.1}$$

$$M_0 = r_0 \times F = r_0 \times (R.W) \tag{7.2}$$

where R is the rotation matrix from the inertial frame to the sensors local frame as seen in Figure 7.4, W is the target's weight and r_0 is the position vector from the sensors local frame to the target's center of gravity. These values are calculated from the robot and target specifications.

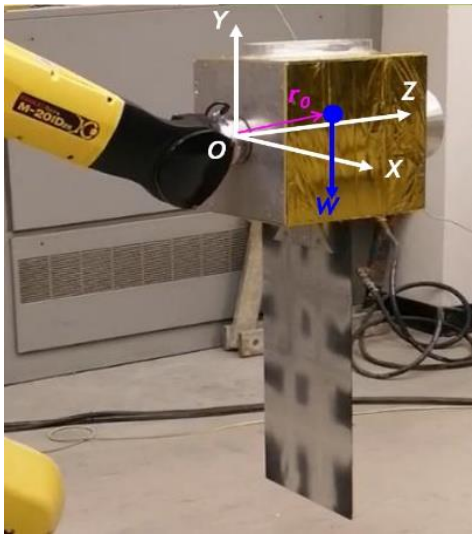


Figure 7.4 ATI sensor frame

After a disturbance occurs (contact with the target), the force and moment are measured by the ATI sensor again:

$$F = R.W + F_c \tag{7.3}$$

$$M = r_0 \times (R.W) + r \times F_c \tag{7.4}$$

Then the difference is calculated.

$$F_c = F - R.W \tag{7.5}$$

$$r \times F_c = M - r_0 \times (R.W) \tag{7.6}$$

Finally, we use F_c and $(r - r_0)$ in the dynamic model of the target to calculate the motion caused by the disturbance. This motion is mapped into robotic joint angles that are fed into the physical robot carrying the Mock-up satellite that will mimic a floating target.

To test this concept, we record the sensor values for 10 seconds, while we apply a disturbance to the Mock-up satellite.

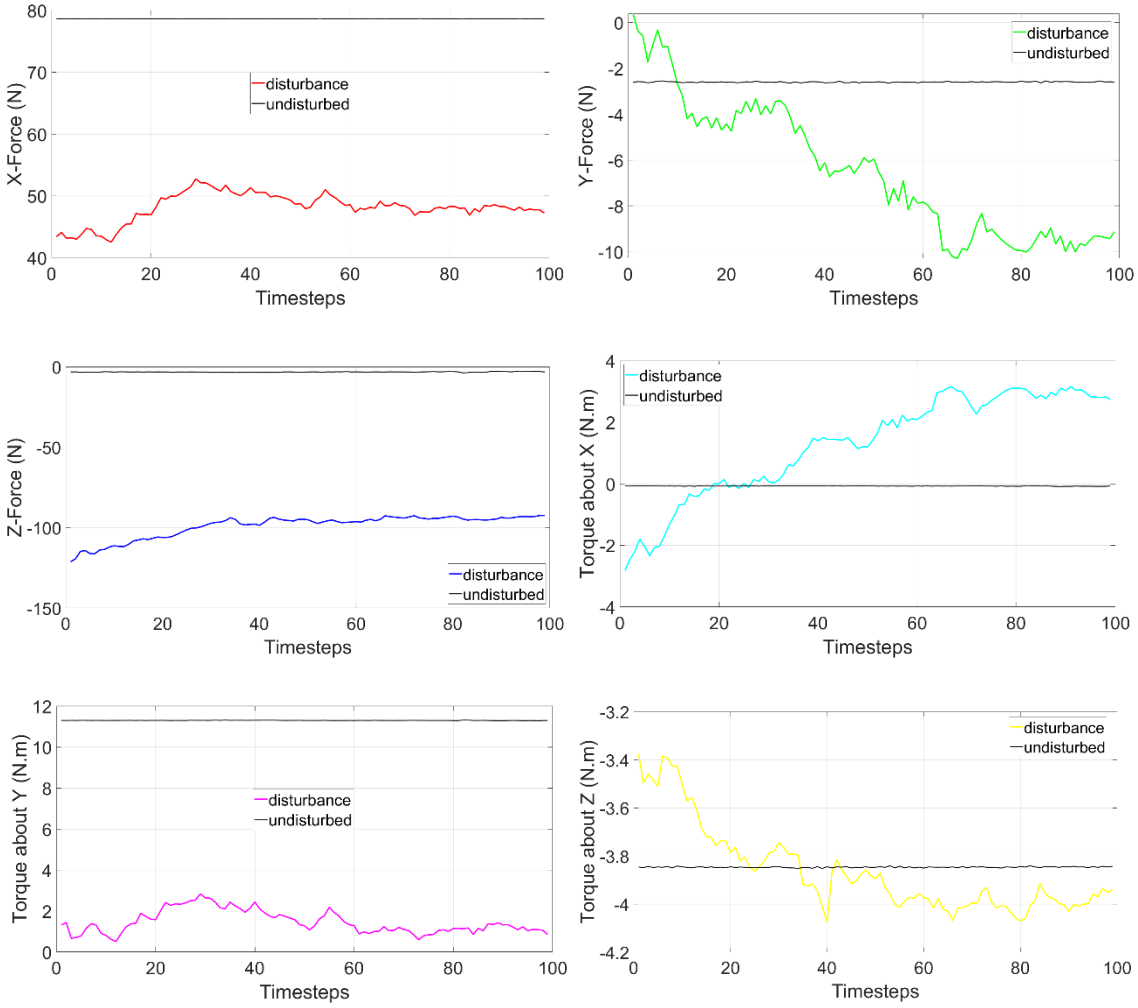


Figure 7.5 Forces and torques measured with and without an external disturbence

Figure 7.5 illustrates the forces along the x, y, and z axes, along with the corresponding torques about these axes. The black line represents the measurements in the absence of external disturbances (only sensing the wight

of mock-up), while the colored lines depict the sensory data collected during disturbances. Next, the differences are computed, and the resulting accelerations of the mockup satellite are determined. Using this motion, the position of the mockup target is calculated after a gravity compensation delay (t_{gc} seconds). For the measured forces shown in Figure 1, the target moves from an initial pose of (x, y, z, roll, pitch, yaw) coordinates (1055, 0, 1055, 146, -90, 30) to a final pose of (160.2, -0.01, 1303.4, -0.004, -57.7, -0.001) after 10 seconds.

These coordinates are then converted into physical robot joint angles, where the initial joint angles are (0, 0, 0, 0, 0, 0) and the final joint angles are (-0.02, -46.4, 34.1, 0, -1.7, -0.003). The physical robot then executes StreamMotion to achieve the desired position.

Using the force/torque values that were recorded, we compensate for the weight of the satellite by calculating the difference as previously described.

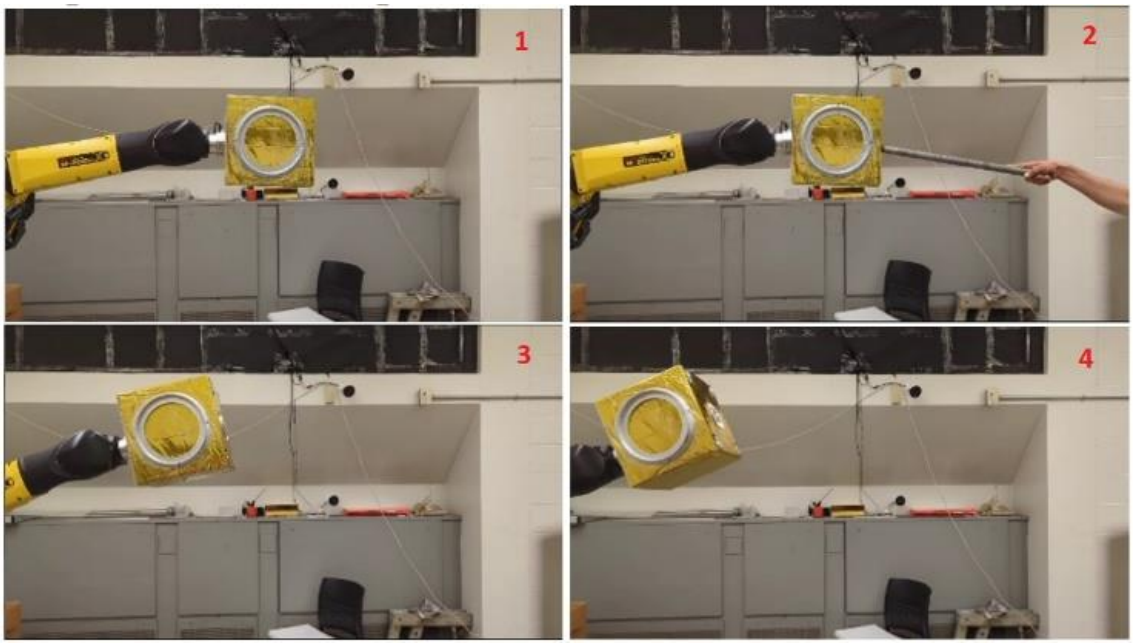


Figure 7.6 ATI sensor frame

Building on the success of the gravity compensation concept, future work will integrate sensory data into the gripper control, enabling the gripper to move in coordination with the target satellite and achieve a soft capture.

7.4 Target Satellite Motion

This section includes the experimental test that utilizes the industrial robotic arm mounted with the Mock-up satellite to mimic the motion of an actual 6DOF free-floating debris tumbling in space. The target satellite motion is based on the tumbling rotational rate of a GOES 8 geostationary satellite, as observed in [99] and the rotational rate of Envisat as observed in [100]. Specifically, a drift rate of -0.01 m/s in the x-axis and a tumbling rate of 0.28,

2.8, 0.28 deg/s about the roll, pitch, and yaw axis.

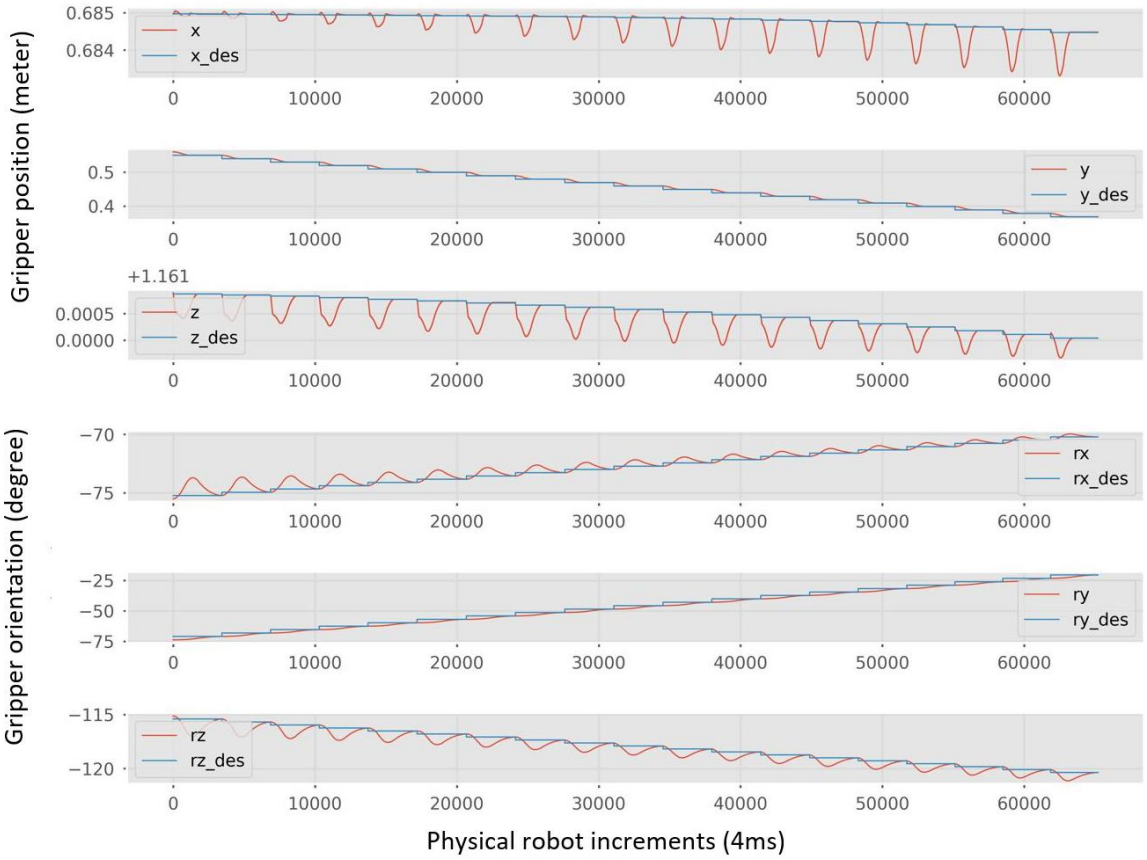


Figure 7.7 Robotic arm mounted Mock-up sattelite cartesian motions

Figure 7.7 shows the desired cartesian motions of the tumbling target as previously described and the actual cartesian motions achieved by the physical robotic arm vs timesteps used to create the motion commands. Notably, there exists some different caused by the physical robotic arm IK, however the scale is less than 0.001 and these differences are negligible.

Figure 7.8 shows the desired joint commands fed into the robotic arm

and the actual joints positions received from the robot at each timestep. The Mock-up satellite successfully mimics a tumbling debris in space. Snapshots of the motion are shown in Figure 7.9.

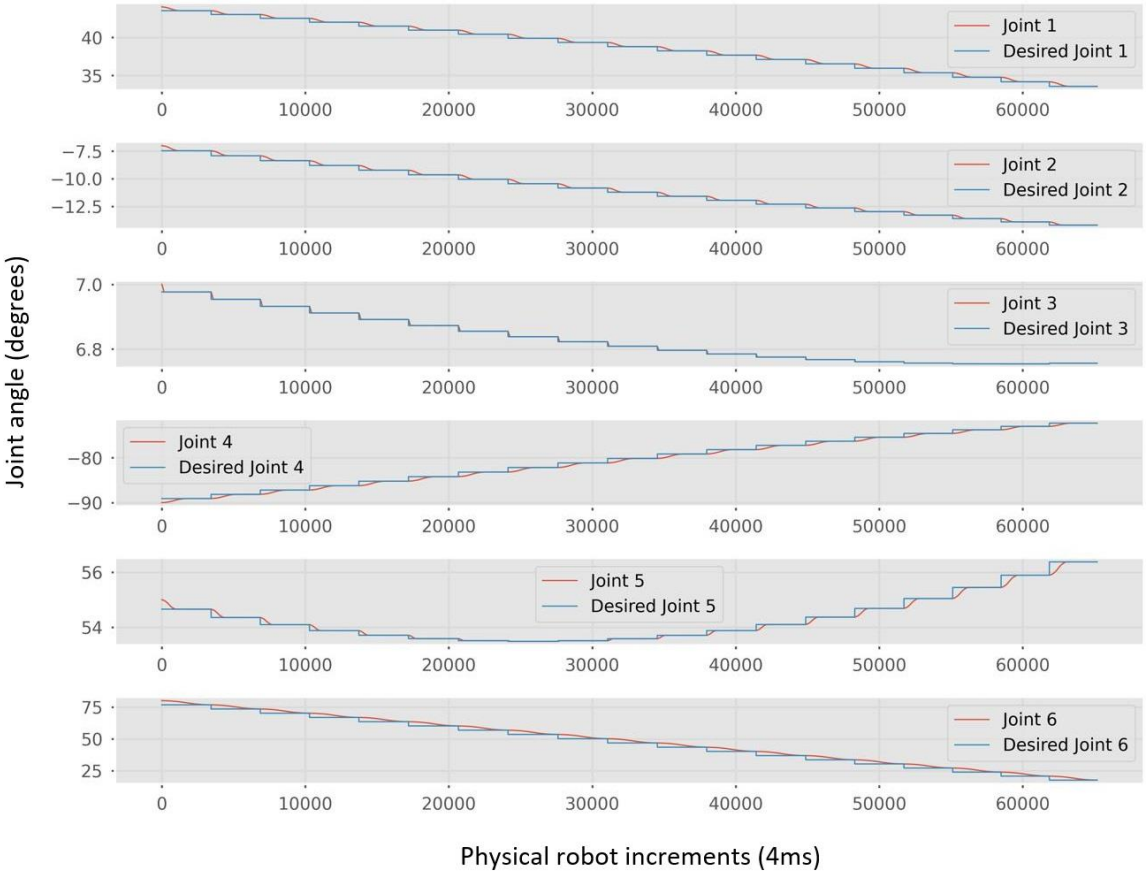


Figure 7.8 Robotic arm mounted Mock-up sattelite joint motions

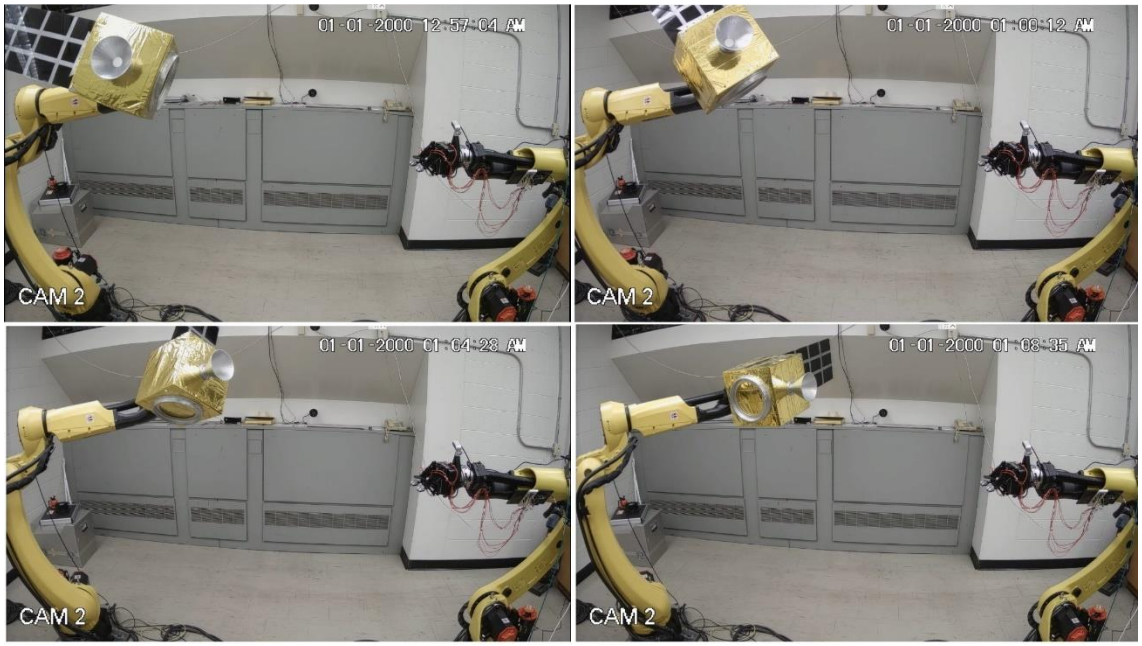


Figure 7.9 Mock-up sattelite tumbing in space

The physical robotic arm was able to successfully perform the 6-DOF motion maneuvers that a tumbling satellites/ target achieves in the microgravity conditions.

The forces and torques measured by the ATI force/load sensor, depicted in Figure 3.10, are collected during the tumbling motion of the mock-up satellite using the MATLAB app. Figure 7.10 illustrates the force values in the x, y, and z directions recorded by the sensor, while Figure 7.11 presents the torque values about the x, y, and z axes captured by the sensor. We are able to successfully obtain the values, and in the future work these values will be integrated in the gripper controller.

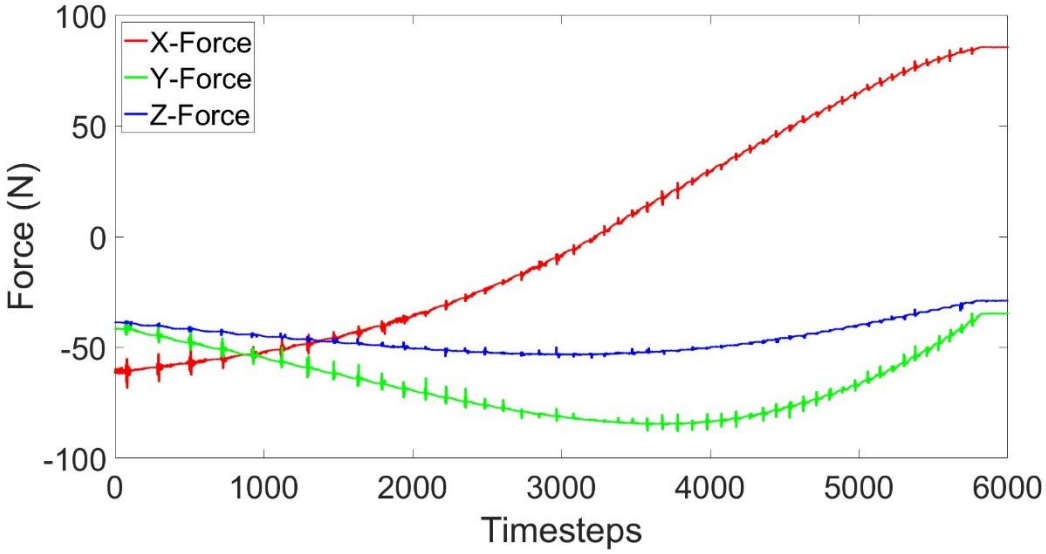


Figure 7.10 ATI force/load sensor: force values

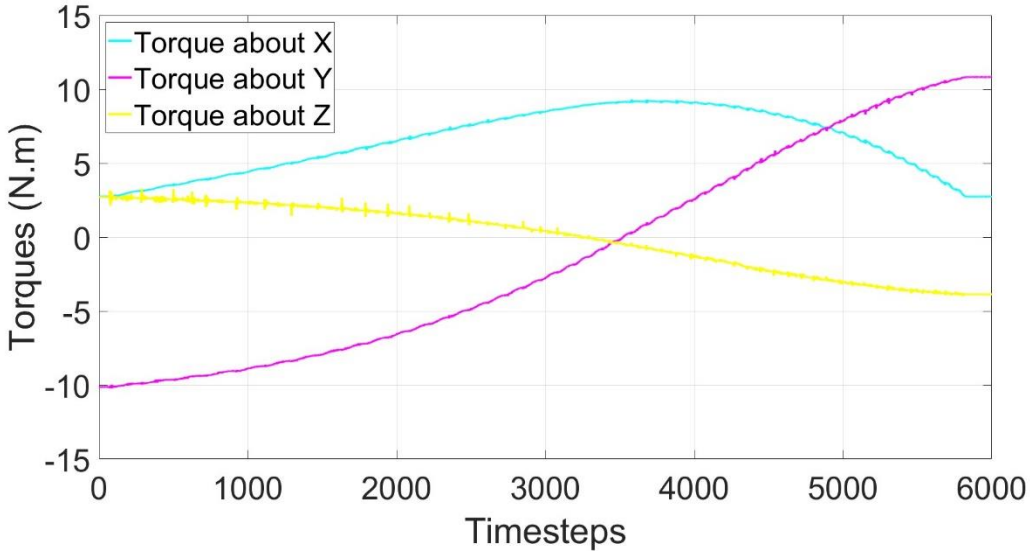


Figure 7.11 ATI force/load sensor: torque values

7.5 Chaser Robot Motion

This section includes the experimental test that utilizes the industrial robotic arm mounted with the camera & gripper interface to mimic the motion of an actual 6DOF free-floating robotic capture mission in space. After obtaining the pose of the target using the camera, the path of the chaser robotic arm is calculated. Figure 7.12 shows the desired cartesian motions of the gripper mounted robotic arm calculated and the actual cartesian motions achieved by the physical robotic arm vs timesteps used for motion commands.

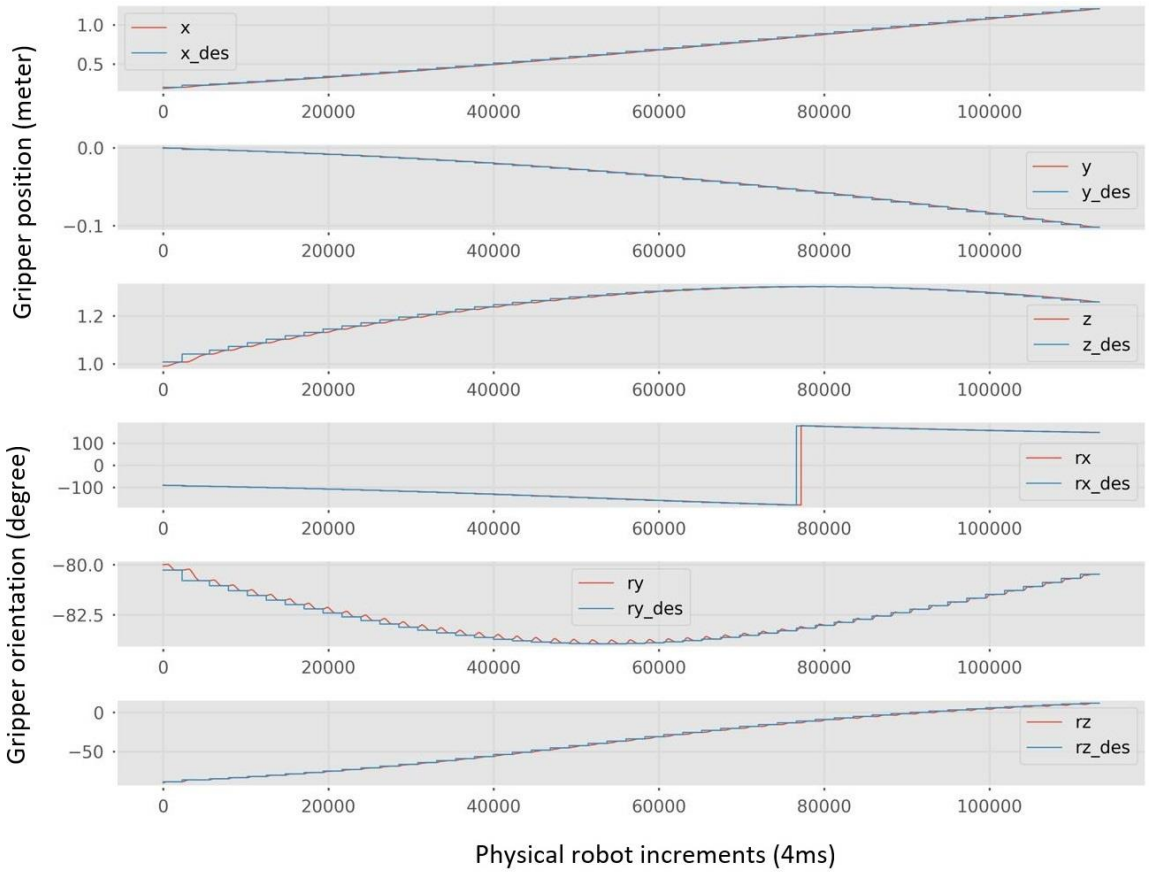


Figure 7.12 Robotic arm mounted gripper cartesian motions

Figure 7.13 shows desired the joint commands after kinematic equivalence that are fed into the industrial robot to achieve capture and the actual joint positions feedback from the physical robot at each increment. Snapshots of the motion are shown in Figure 7.14.

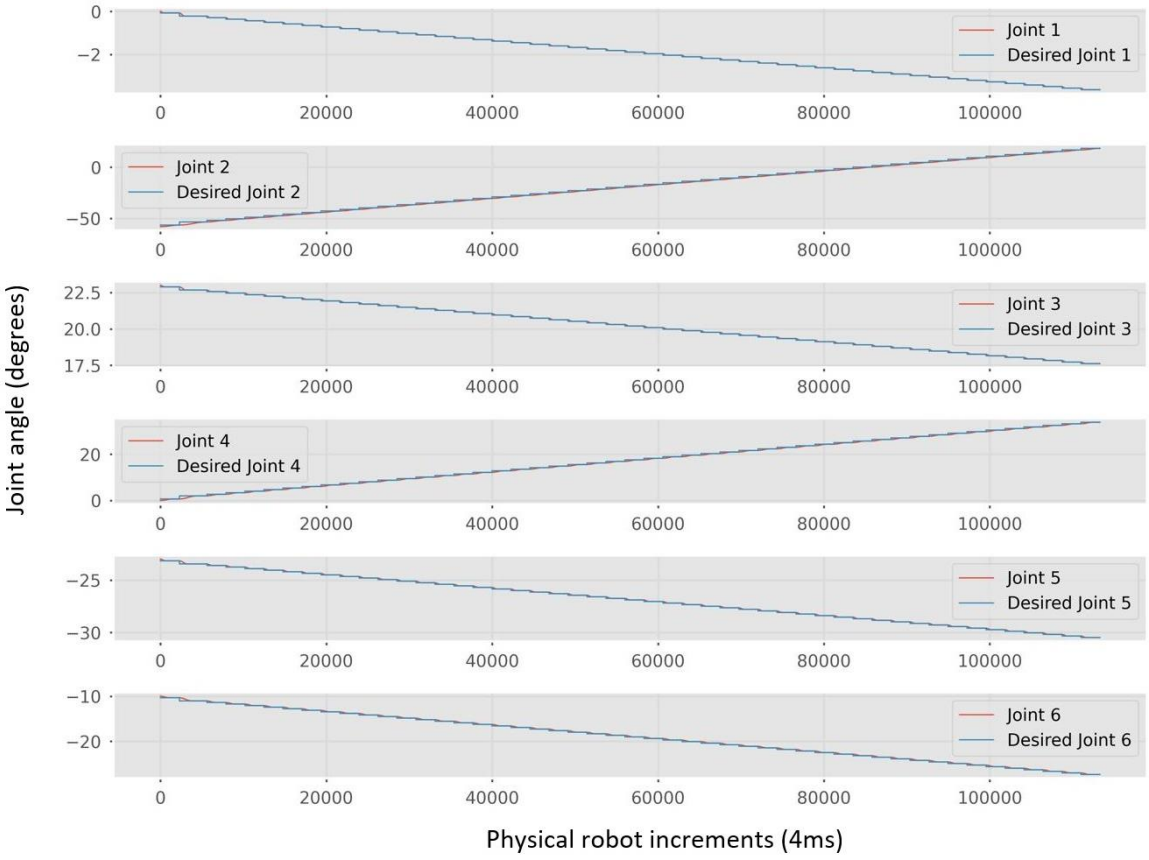


Figure 7.13 Robotic arm mounted gripper joint motions

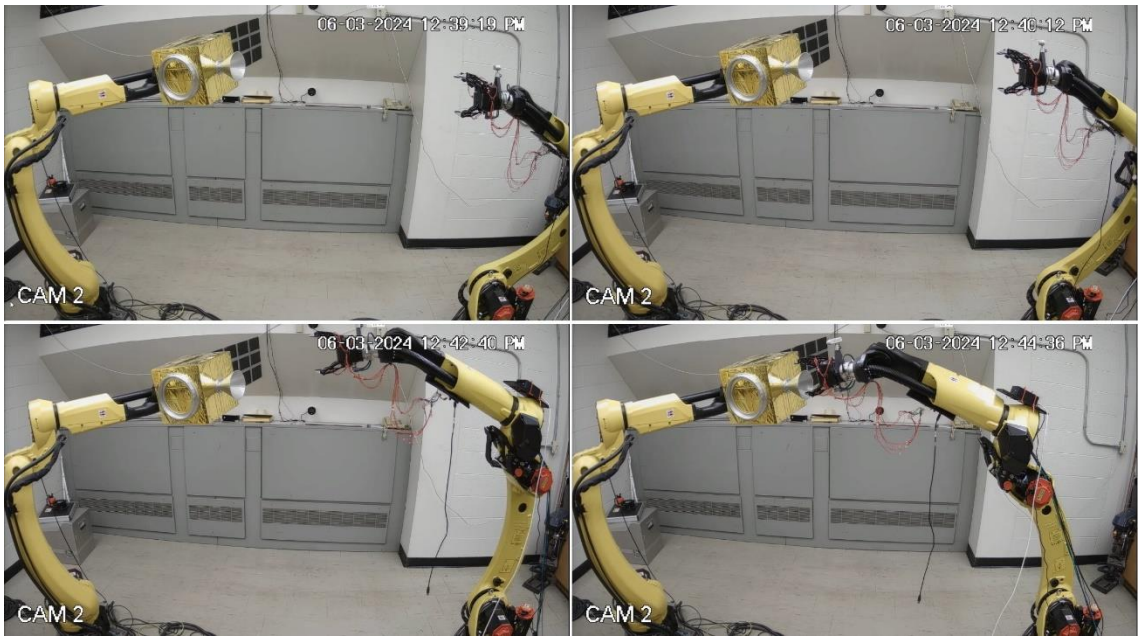


Figure 7.14 Gripper capture of target

The gripper is guided by the Intel camera capable of obtaining the target's grapple fixture position during the approach phase. As shown in Figure 7.15, the camera's point of view during the pre-capture target approach highlights that the target eventually moves out of view when in close proximity.

In such cases, the controller relies on the last known position estimate and continues moving toward that position. Future work will involve integrating external cameras at various locations, as shown in Figure 3.11, in addition to the eye-in-hand camera, to maintain continuous tracking of the target.

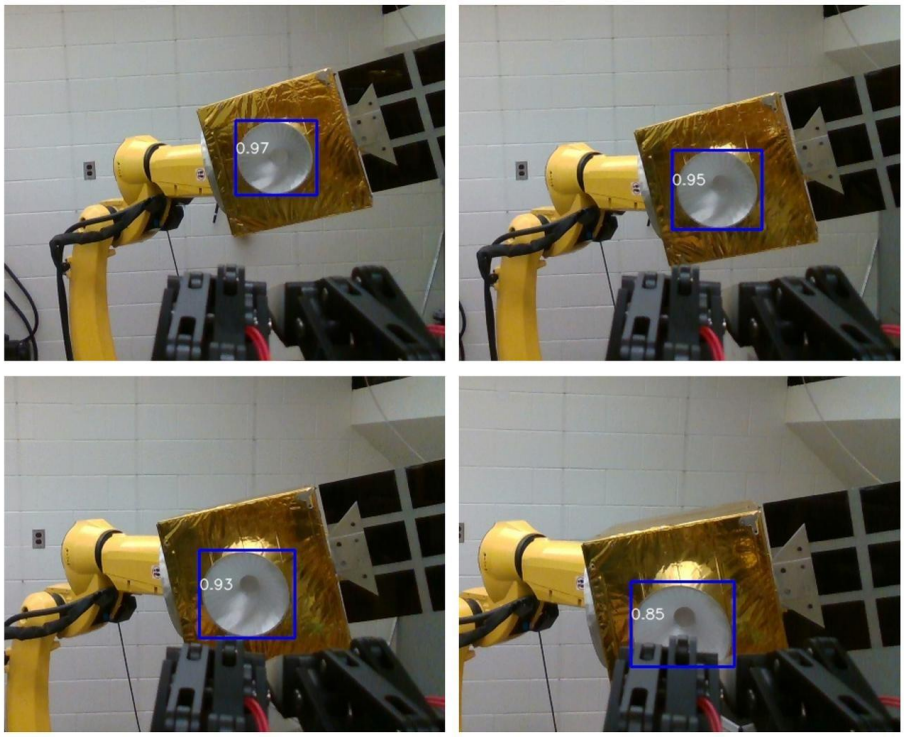


Figure 7.15 Camera point-of-view and bounding box during pre-capture phase

The fully simultaneous capture mission is shown in Figure 7.16.

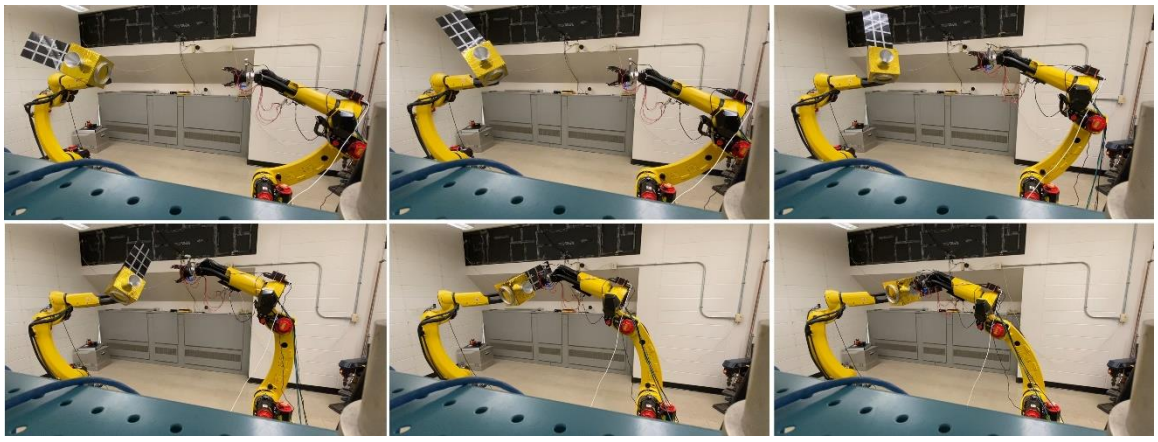


Figure 7.16 Full debris capture mission

7.6 Capture Mission - Controlled by Trained Agent

In this section, we conduct a capture mission experiment using the proposed motion planning for a free-floating space manipulator, leveraging reinforcement learning. The capture of the target will be achieved using EE motions generated by the RL trained agent in the previous chapters. The gripper mounted physical robot will move according to the path created in simulation by a trained RL agent as shown in Figure 7.22. The experiment is achieved with the following steps. The camera starts by taking an image of the target and extract the target pose position from the image features, where the orientation is kept facing forward with zeros row, pitch, and yaw angles.

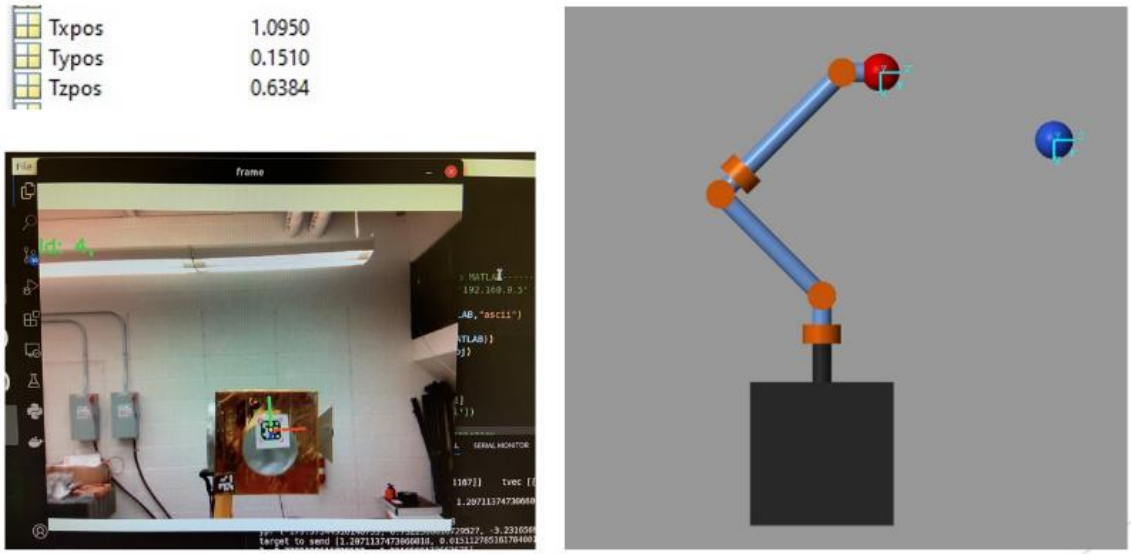


Figure 7.17 Target position extracted from camera (left), Target placed relative to EE in simulation (right)

Figure 7.17 (left) shows the camera's point of view, and the target

position estimates. The values estimated are ($x_T, y_T, z_T, \alpha_T, \beta_T, \gamma_T$ $= 1.0950, 0.1510, 0.6384, 0, 0, 0$), these values are sent to MATLAB /Simulink environment shown in Figure 7.17 (right). The environment will receive in the target pose as input observations and run the simulation of the free-floating space manipulator with the RL trained agent to achieve target alignment pose alignment. The trained agent provides the solution instantaneously, Figure 7.18 shows the motion solution to achieve capture.

The pose error between the EE and the target over time is seen in Figure 7.19. The trained agent guides the EE towards the desired target pose range within approximately 4 seconds.

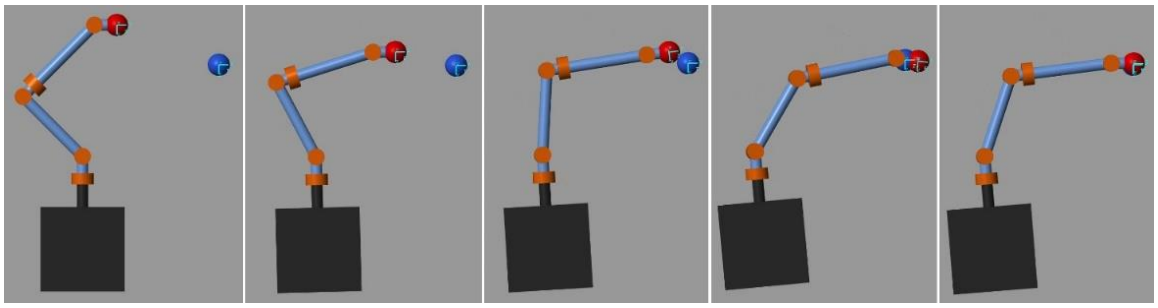


Figure 7.18 Motion solution obtained from trained agent

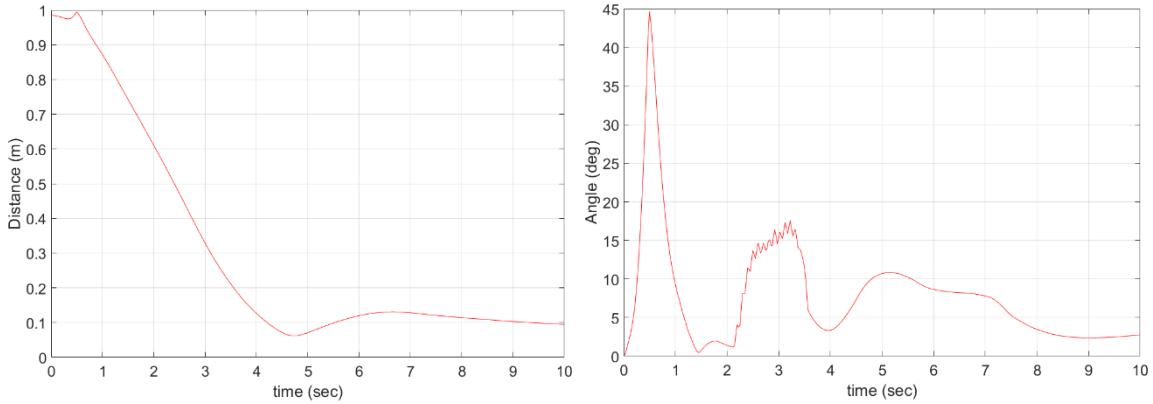


Figure 7.19 Pose tracking errors (d, ϑ) over time

Figure 7.20 shows the free-floating space manipulator's EE absolute motions in micro-gravity.

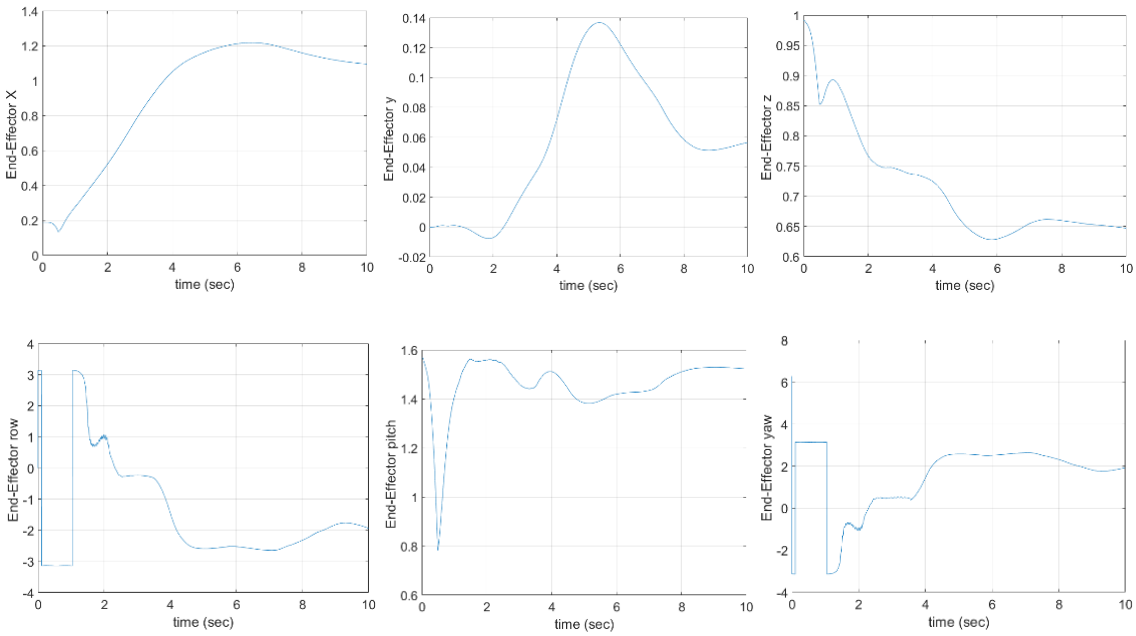
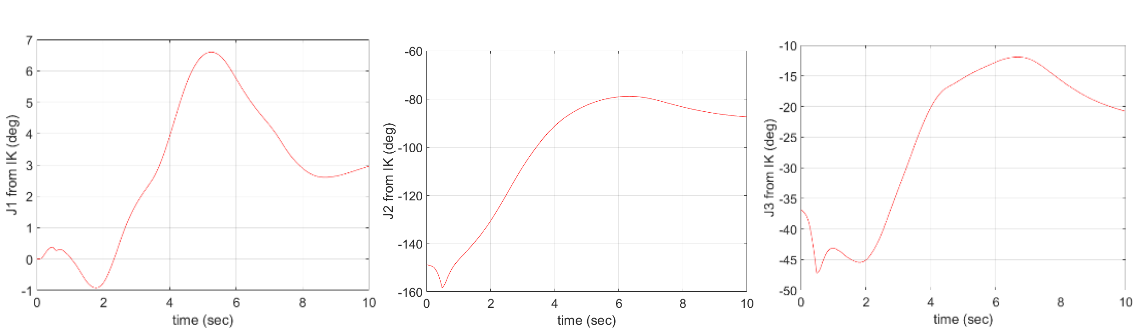


Figure 7.20 Time histories of simulated space manipulator EE pose

The path is recorded and using kinematic equivalence described in Section 3.2. The path is converted to joint commands that are fed into the gripper mounted industrial robot to achieve EE and target alignment. the joint commands are shown in Figure 7.21.



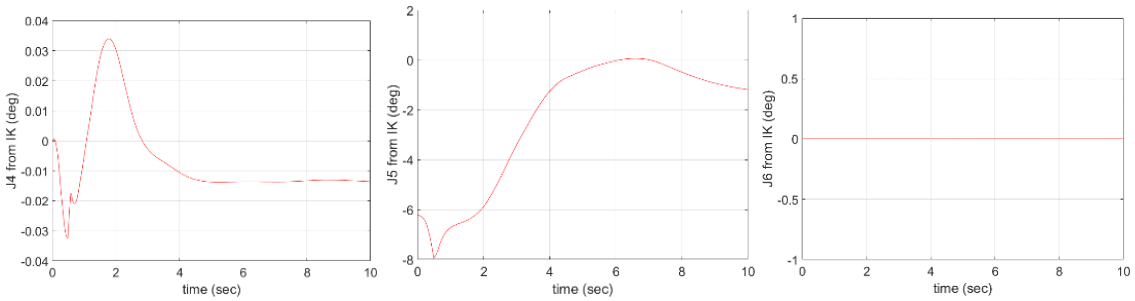


Figure 7.21 Time histories of Joint commands that are executed by the physical robot to achieve capture

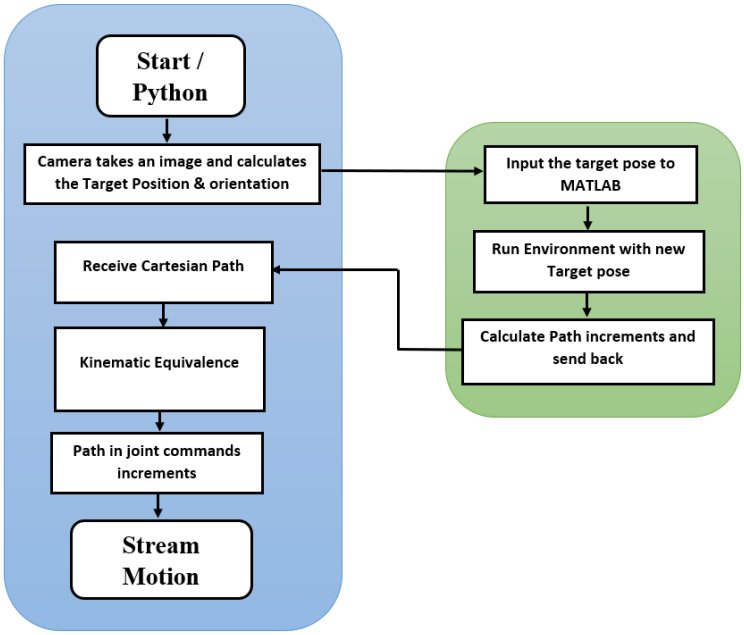


Figure 7.22 Control scheme of camera/gripper mounted robot

The full experimental capture mission is shown in Figure 7.23.

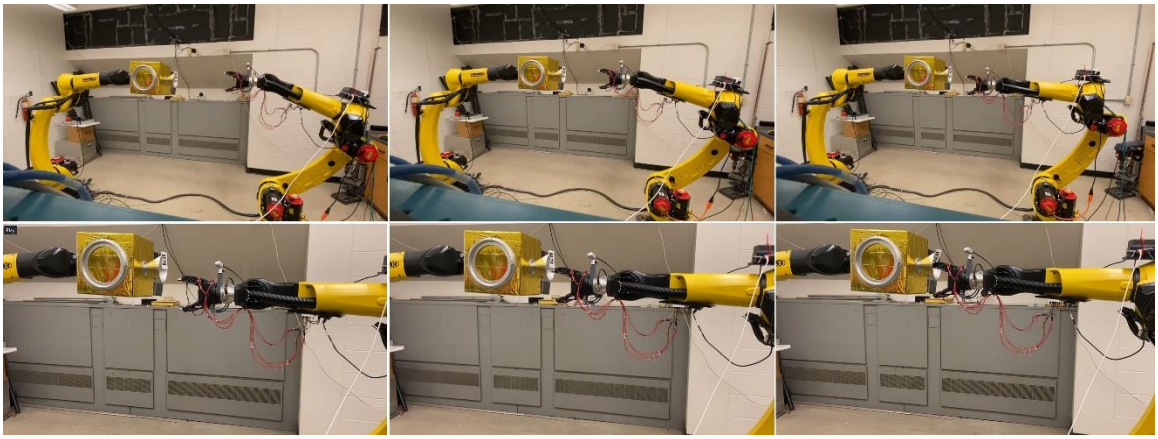


Figure 7.23 Capture mission experiment

In summary, during the pre-grasping phase, the focus is on non-contact HIL simulation. Once the target pose is acquired, the mission is simulated. The simulation runs for 10 seconds, with 400 timesteps, each representing a 0.25 second interval. However, the physical arm requires discrete commands at 0.004 second intervals. To ensure that the physical robot's end effector (EE) accurately follows the simulated space manipulator's EE, each simulated timestep is subdivided into the corresponding number of physical robot timesteps. This process inevitably introduces a delay. The two industrial robots are synchronized at each simulation timestep, meaning that before they can move to the next simulated positions, they must first reach the positions from the previous timestep, regardless of any delay. This ensures synchronization between the robots at every new timestep.

In the post-grasping phase, the focus shifts to contact-based HIL simulation. Before advancing to each new timestep, forces and torques are captured using ATI sensors, and gravity compensation is applied. These measurements are recorded, and the target's motion is simulated. After this simulation, the target's new position is determined, and the physical robot moves accordingly, with the space robot following.

The process introduces a force delay [101], due to the need to record forces and torques and apply gravity compensation before every simulation timestep.

Chapter 8 CONCLUSION AND FUTURE WORK

Summary: This chapter summarizes the contributions of the study and provides suggestions for future research directions for the continuation of the current study.

8.1 Contribution

This dissertation focuses on using Reinforcement Learning for path planning and control of a free-floating space robotic manipulator for debris removal. This research systematically investigated the dynamics and control of the space manipulator system, including the environment design, control strategy, and experimental validation. The main contributions of this research are summarized as follows.

8.1.1 Space Manipulator Control using Reinforcement Learning

The current work develops the framework for using reinforcement learning for path planning & control of a free-floating space manipulator to achieve precise capture. An innovative reward function is introduced in the RL framework to ensure accurate alignment of the manipulator's end effector with its target, despite disturbances from the spacecraft and the need for obstacle and collision avoidance. In this study the use of quaternions for orientation representation was implemented to avoid the singularities associated with

conventional Euler angles and enhance the training process' efficiency. Furthermore, the reward function incorporates joint velocity constraints to refine the path planning for the manipulator joints, enabling efficient obstacle and collision avoidance. A key feature of this study is the inclusion of observation noise in the training process to enhance the robustness of the agent. Results demonstrate that the proposed reward function enables effective exploration of the action space, leading to high precision in achieving the desired objectives.

Additionally, the reinforcement learning agent is trained with varying initial conditions that are randomly set at the start of each episode, including the manipulator's initial joint angles, target positions, and obstacle locations. The trained agent is expected to find suitable paths for any target location, avoiding obstacles regardless of the manipulator's initial position.

The study provides a solid theoretical foundation for the application of reinforcement learning in complex free-floating space robotic operations and offers insights for future space missions.

8.1.2 HIL Ground Testbed and Experimental Verification

This research details my efforts to design and build a hardware-in-the-loop ground testbed with active gravity compensation in our lab at York University. The testbed is capable of performing the 6DOF motion maneuvers

that a free-floating robotic manipulator or tumbling satellite/target can achieve in space-like conditions. It utilizes two industrial-sized 6DOF robotic arms, and various sensing equipment, including cameras and force/torque sensors. This work also describes how our testbed can be used for assessing the control algorithms for the robotic capture of a non-collaborative target in space, and our initial experimental results in motion control, computer vision, and sensing capabilities.

My contribution involved developing the main interface between the laboratory PCs and the industrial robotic controller, specifically utilizing the StreamMotion protocol as the communication baseline. This enabled the subsequent implementation of motions on the physical robot, including the mock-up satellite (target) and chaser (gripper) end-effector movements.

Additionally, I developed the forward and inverse kinematics for the physical robots, focusing on establishing kinematic equivalence between the free-floating space manipulator and the physical robot. Furthermore, I tested the camera's pose estimation capabilities using reliable, custom-coded motions and integrated the estimated pose values as inputs to our controller or as observations for our trained agent.

Lastly, I incorporated the ATI force/torque sensor readings into MATLAB, allowing for the recording of sensory data, calculation of force/torque discrepancies, and motion adjustments to compensate for these differences.

This facility will serve as an invaluable tool for the development and validation of space robotic manipulators, ultimately improving their effectiveness and reliability in space missions.

8.2 Conclusion

Path planning & control of a free-floating space robotic manipulator is challenging due to the dynamic coupling between the base spacecraft and manipulator which critically impacts control and precision. This research has made significant contributions to the field of debris removal using space robotics by addressing the complex problem of motion planning for a 6DOF free-floating space manipulator using Reinforcement Learning.

The research has ensured accurate pose alignment between the end-effector and target debris, implemented collision avoidance strategies with the target, manipulator links, and external obstacles, and integrated optimization terms in the reward function to smooth joint velocities. Additionally, it has adapted to the high mass ratio between the manipulator and its base spacecraft.

To enhance the robustness of the RL agent against real-world sensor imperfections, observation noise was incorporated into the training process. The agent was also trained with varying initial conditions, such as random initial joint angles, target positions, and obstacle locations, ensuring the ability

to find suitable paths for any target location while avoiding obstacles. This provides a solid theoretical foundation for applying RL in complex free-floating space robotic operations and offers valuable insights for future space missions.

A standout contribution is the creation of a hardware-in-the-loop ground testbed with my colleagues at York University, which replicates the 6DOF motion maneuvers of a free-floating robotic manipulator or tumbling satellite/target in space-like conditions. This testbed utilizes two industrial-sized 6DOF robotic arms and different sensing equipment, including cameras and force/torque sensors, for enhanced sensing capabilities.

The testbed facilitates the experimental validation of control algorithms for robotic capture of non-cooperative targets in space, with initial experimental results demonstrating progress in motion control, computer vision, and sensing capabilities. Ultimately, this research has established an invaluable tool for developing and validating space robotic manipulators, thereby improving their effectiveness and reliability in space missions. These contributions represent significant progress in the preparation for autonomous space debris removal and other space missions involving robotic manipulators.

The limitations of this work include several unmodeled aspects of the system's physics in simulation. Notably, rigid bodies were used instead of flexible robotic joints. More importantly, the simulation does not account for the presence of friction within the robotic manipulator system.

Another limitation arises during the obstacle avoidance training. While the reward function penalizes collisions between the robotic links and obstacles or other links, no penalty is applied for interactions between the links and the free-floating base. This omission may lead to unintended base movement due to reaction forces.

Additionally, when introducing noise into the observations, the noise was modeled as following a zero-mean multivariate normal distribution. In practice, however, real-world sensors typically exhibit noise patterns that deviate from this assumption.

Regarding the limitations of our hardware-in-the-loop (HIL) testing facility, there is currently a delay during the post-capture phase. Sensors measure force and torque values, calculate the resulting motion from these reactions, apply gravity compensation, and subsequently move the target. However, this gravity compensation operates in an open-loop configuration due to inherent delays. Future efforts will focus on minimizing these delays to enable real-time gravity compensation.

Another limitation pertains to scaling from a single simulated timestep to multiple physical robotic timesteps. Although the physical robots accurately replicate the trajectory of the space manipulator and target system, their slower velocities result in a longer duration to complete the capture mission. Future work will aim to resolve this time discrepancy to better synchronize the

simulated and physical systems.

8.3 Future Work

The following research is summarized as follows to continue and expand the current work.

(i) Consider different optimization terms in the reward function, including joint torques constraints, base spacecraft reaction minimization, and energy consumption.

(ii) Improve the reward function to minimize the EE and Target pose errors even further in dynamic Agents by increasing exploitation over exploration.

(iii) Extend the research to include more complex environmental conditions, such as varying lighting, and different types of debris, to enhance the robustness and versatility of the system.

(iv) Further investigate using reinforcement learning to capture a moving target and the best reward function to achieve it.

(v) Continue to refine and enhance the hardware components of the testbed, incorporating new sensors, actuators, and materials to improve performance and reliability.

(vi) The camera orientation estimation must be trained and verified, to obtain the full target pose needed for our motion planning algorithms.

(vii) Integrate the force/torque sensory data obtained as feedback to the gripper motion controller to achieve post capture stabilization.

(viii) Achieve the capture mission experiment for a closed-loop configuration, leveraging feedback from the camera and controlling the motion using the RL trained agent.

Bibliography

[1] NASA Space Science Data Coordinated Archive, n.d. https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1957-001B.

[2] D.J. Kessler, N.L. Johnson, J.C. Liou, M. Matney, The kessler syndrome: implications to future space operations, Advances in the Astronautical Sciences 137 (2010) 2010.

[3] Space Environment Statistics, n.d. https://sdup.esoc.esa.int/discosweb/statistics/.

[4] J.-C. Liou, N.L. Johnson, N.M. Hill, Controlling the growth of future LEO debris populations with active debris removal, Acta Astronautica 66 (2010) 648–653.

[5] A. Flores-Abad, O. Ma, K. Pham, S. Ulrich, A review of space robotics technologies for on-orbit servicing, Progress in Aerospace Sciences 68 (2014) 1–26. https://doi.org/10.1016/j.paerosci.2014.03.002.

[6] C. Bonnal, J.-M. Ruault, M.-C. Desjean, Active debris removal: Recent progress and current trends, Acta Astronautica 85 (2013) 51–60. https://doi.org/10.1016/j.actaastro.2012.11.009.

[7] M. Shan, J. Guo, E. Gill, Review and comparison of active space debris capturing and removal methods, Progress in Aerospace Sciences 80 (2016)

18–32. https://doi.org/10.1016/j.paerosci.2015.11.001.

[8] E. Papadopoulos, F. Aghili, O. Ma, R. Lampariello, Robotic Manipulation and Capture in Space: A Survey, Front. Robot. AI 8 (2021). https://doi.org/10.3389/frobt.2021.686723.

[9] Canadarm, Canadarm2, and Canadarm3 – A comparative table, n.d. https://www.asc-csa.gc.ca/eng/iss/canadarm2/canadarm-canadarm2-canadarm3-comparative-table.asp.

[10] Dextre's data sheet, n.d. https://www.asc-csa.gc.ca/eng/iss/dextre/data-sheet.asp.

[11] Japanese Experiment Module Remote Manipulator System, n.d. https://iss.jaxa.jp/en/kibo/about/kibo/rms/.

[12] European Robotic Arm, n.d. https://www.esa.int/Science_Exploration/Human_and_Robotic_Exploration/International_Space_Station/European_Robotic_Arm.

[13] ETS-VII (Engineering Test Satellite VII), n.d. https://www.eoportal.org/satellite-missions/ets-vii#overview.

[14] SPACE EXPLORATION AND INFRASTRUCTURE MISSION PARTNE, n.d. https://mda.space/space-exploration-and-infrastructure.

[15] W.-J. Li, D.-Y. Cheng, X.-G. Liu, Y.-B. Wang, W.-H. Shi, Z.-X. Tang, F. Gao, F.-M. Zeng, H.-Y. Chai, W.-B. Luo, Q. Cong, Z.-L. Gao, On-orbit service (OOS) of spacecraft: A review of engineering developments, Progress in Aerospace Sciences 108 (2019) 32–120. https://doi.org/10.1016/j.paerosci.2019.01.004.

[16] C.G. Henshaw, The darpa phoenix spacecraft servicing program: Overview and plans for risk reduction, in: International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), European Space Agency, 2014. http://robotics.estec.esa.int/i-SAIRAS/isairas2014/Data/Session%205b/ISAIRAS_FinalPaper_0043.pdf

(accessed June 25, 2024).

[17] E. Galceran, M. Carreras, A survey on coverage path planning for robotics, Robotics and Autonomous Systems 61 (2013) 1258–1276. https://doi.org/10.1016/j.robot.2013.09.004.

[18] Path planning algorithm development for autonomous vacuum cleaner robots | IEEE Conference Publication | IEEE Xplore, (n.d.). https://ieeexplore.ieee.org/abstract/document/6850799 (accessed June 25, 2024).

[19] B. Pang, Y. Song, C. Zhang, H. Wang, R. Yang, A Swarm Robotic Exploration Strategy Based on an Improved Random Walk Method, Journal of Robotics 2019 (2019) 6914212. https://doi.org/10.1155/2019/6914212.

[20] Probabilistic roadmaps for path planning in high-dimensional configuration spaces | IEEE Journals & Magazine | IEEE Xplore, (n.d.). https://ieeexplore.ieee.org/abstract/document/508439 (accessed June 25, 2024).

[21] M. Ulrich, G. Lux, L. Jürgensen, G. Reinhart, Automated and Cycle Time Optimized Path Planning for Robot-Based Inspection Systems, Procedia CIRP 44 (2016) 377–382. https://doi.org/10.1016/j.procir.2016.02.021.

[22] B. Fu, L. Chen, Y. Zhou, D. Zheng, Z. Wei, J. Dai, H. Pan, An improved A* algorithm for the industrial robot path planning with high success rate and short length, Robotics and Autonomous Systems 106 (2018) 26–37. https://doi.org/10.1016/j.robot.2018.04.007.

[23] Sampling-Based Robot Motion Planning: A Review | IEEE Journals & Magazine | IEEE Xplore, (n.d.). https://ieeexplore.ieee.org/abstract/document/6722915 (accessed June 25, 2024).

[24] J.J. Kuffner, S.M. LaValle, RRT-connect: An efficient approach to single-

query path planning, in: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), 2000: pp. 995–1001 vol.2. https://doi.org/10.1109/ROBOT.2000.844730.

[25] Sampling-based algorithms for optimal motion planning - Sertac Karaman, Emilio Frazzoli, 2011, (n.d.). https://journals.sagepub.com/doi/abs/10.1177/0278364911406761 (accessed June 25, 2024).

[26] B. Han, S. Liu, RRT Based Obstacle Avoidance Path Planning for 6-DOF Manipulator, in: 2020 IEEE 9th Data Driven Control and Learning Systems Conference (DDCLS), 2020: pp. 822–827. https://doi.org/10.1109/DDCLS49620.2020.9275125.

[27] Z.H. Zhu, R.V. Mayorga, A.K.C. Wong, Dynamic robot manipulator trajectory planning for obstacle avoidance, Mechanics Research Communications 26 (1999) 139–144.

[28] J. Sun, J. Tang, S. Lao, Collision Avoidance for Cooperative UAVs With Optimized Artificial Potential Field Algorithm, IEEE Access 5 (2017) 18382–18390. https://doi.org/10.1109/ACCESS.2017.2746752.

[29] N. Zhang, Y. Zhang, C. Ma, B. Wang, Path planning of six-DOF serial robots based on improved artificial potential field method, in: 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2017: pp. 617–621. https://doi.org/10.1109/ROBIO.2017.8324485.

[30] L. Luo, H. Wen, Q. Lu, H. Huang, W. Chen, X. Zou, C. Wang, Collision-Free Path-Planning for Six-DOF Serial Harvesting Robot Based on Energy Optimal and Artificial Potential Field, Complexity 2018 (2018) 3563846. https://doi.org/10.1155/2018/3563846.

[31] S.N. Gai, R. Sun, S.J. Chen, S. Ji, 6-DOF Robotic Obstacle Avoidance Path Planning Based on Artificial Potential Field Method, in: 2019 16th

International Conference on Ubiquitous Robots (UR), 2019: pp. 165–168. https://doi.org/10.1109/URAI.2019.8768792.

[32] S. Liu, Q. Zhang, D. Zhou, Obstacle Avoidance Path Planning of Space Manipulator Based on Improved Artificial Potential Field Method, J. Inst. Eng. India Ser. C 95 (2014) 31–39. https://doi.org/10.1007/s40032-014-0099-z.

[33] X. Hu, X. Huang, T. Hu, Z. Shi, H. Li, Coupling minimization with obstacles avoidance of free-floating space robots based on hybrid map in configuration space, International Journal of Advanced Robotic Systems 15 (2018) 1729881418816557. https://doi.org/10.1177/1729881418816557.

[34] M.-K. Ng, Y.-W. Chong, K. Ko, Y.-H. Park, Y.-B. Leau, Adaptive path finding algorithm in dynamic environment for warehouse robot, Neural Comput & Applic 32 (2020) 13155–13171. https://doi.org/10.1007/s00521-020-04764-3.

[35] C. Lamini, S. Benhlima, A. Elbekri, Genetic Algorithm Based Approach for Autonomous Mobile Robot Path Planning, Procedia Computer Science 127 (2018) 180–189. https://doi.org/10.1016/j.procs.2018.01.113.

[36] A. Borboni, R. Bussola, R. Faglia, P.L. Magnani, A. Menegolo, Movement Optimization of a Redundant Serial Robot for High-Quality Pipe Cutting, Journal of Mechanical Design 130 (2008). https://doi.org/10.1115/1.2918907.

[37] C. Blum, X. Li, Swarm Intelligence in Optimization, in: C. Blum, D. Merkle (Eds.), Swarm Intelligence: Introduction and Applications, Springer, Berlin, Heidelberg, 2008: pp. 43–85. https://doi.org/10.1007/978-3-540-74089-6_2.

[38] M. Wang, J. Luo, U. Walter, Trajectory planning of free-floating space robot using Particle Swarm Optimization (PSO), Acta Astronautica 112 (2015) 77–88. https://doi.org/10.1016/j.actaastro.2015.03.008.

[39] L. Claussmann, M. Revilloud, S. Glaser, D. Gruyer, A study on al-based approaches for high-level decision making in highway autonomous driving, in: 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2017: pp. 3671–3676. https://doi.org/10.1109/SMC.2017.8123203.

[40] A Survey on Visual Navigation for Artificial Agents With Deep Reinforcement Learning | IEEE Journals & Magazine | IEEE Xplore, (n.d.). https://ieeexplore.ieee.org/abstract/document/9146614 (accessed June 25, 2024).

[41] A.S. Polydoros, L. Nalpantidis, Survey of Model-Based Reinforcement Learning: Applications on Robotics, J Intell Robot Syst 86 (2017) 153–173. https://doi.org/10.1007/s10846-017-0468-y.

[42] S. Gu, E. Holly, T. Lillicrap, S. Levine, Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017: pp. 3389–3396. https://doi.org/10.1109/ICRA.2017.7989385.

[43] S. Dubowsky, E. Papadopoulos, The kinematics, dynamics, and control of free-flying and free-floating space robotic systems, IEEE Transactions on Robotics and Automation 9 (1993) 531–543. https://doi.org/10.1109/70.258046.

[44] F. Pfeiffer, Manipulator Trajectory Planning and Control, IFAC Proceedings Volumes 19 (1986) 325–330. https://doi.org/10.1016/S1474-6670(17)59499-9.

[45] K. Nanos, E.G. Papadopoulos, On the Dynamics and Control of Free-floating Space Manipulator Systems in the Presence of Angular Momentum, Front. Robot. AI 4 (2017). https://doi.org/10.3389/frobt.2017.00026.

[46] T. Rybus, K. Seweryn, J.Z. Sasiadek, Control System for Free-Floating

Space Manipulator Based on Nonlinear Model Predictive Control (NMPC), J Intell Robot Syst 85 (2017) 491–509. https://doi.org/10.1007/s10846-016-0396-2.

[47] E. Papadopoulos, S. Dubowsky, Dynamic Singularities in Free-Floating Space Manipulators, Journal of Dynamic Systems, Measurement, and Control 115 (1993) 44–52. https://doi.org/10.1115/1.2897406.

[48] W. Xu, B. Liang, Y. Xu, Practical approaches to handle the singularities of a wrist-partitioned space manipulator, Acta Astronautica 68 (2011) 269–300. https://doi.org/10.1016/j.actaastro.2010.07.004.

[49] Z. Vafa, S. Dubowsky, The Kinematics and Dynamics of Space Manipulators: The Virtual Manipulator Approach, The International Journal of Robotics Research 9 (1990) 3–21. https://doi.org/10.1177/027836499000900401.

[50] E.G. Papadopoulos, Path Planning For Space Manipulators Exhibiting Nonholonomic Behavior., in: IROS, 1992: pp. 669–675. http://nereus.mech.ntua.gr/Documents/pdf_ps/IROS92.pdf (accessed June 25, 2024).

[51] S.K. Agrawal, R. Garimella, G. Desmier, Free-floating closed-chain planar robots: Kinematics and path planning, Nonlinear Dyn 9 (1996) 1–19. https://doi.org/10.1007/BF01833290.

[52] J. Luo, M. Yu, M. Wang, J. Yuan, A fast trajectory planning framework with task-priority for space robot, Acta Astronautica 152 (2018) 823–835. https://doi.org/10.1016/j.actaastro.2018.09.023.

[53] X. Shao, G. Sun, C. Xue, X. Li, Nonsingular terminal sliding mode control for free-floating space manipulator with disturbance, Acta Astronautica 181 (2021) 396–404. https://doi.org/10.1016/j.actaastro.2021.01.038.

[54] X. Ye, Z.-H. Dong, J.-C. Hong, Research on Adaptive Reaction Null Space Planning and Control Strategy Based on VFF–RLS and SSADE–ELM

Algorithm for Free-Floating Space Robot, Electronics 8 (2019) 1111. https://doi.org/10.3390/electronics8101111.

[55] A New Reinforcement Learning Based Adaptive Sliding Mode Control Scheme for Free-Floating Space Robotic Manipulator | IEEE Journals & Magazine | IEEE Xplore, (n.d.). https://ieeexplore.ieee.org/abstract/document/9138387 (accessed June 25, 2024).

[56] H. Nguyen, H. La, Review of Deep Reinforcement Learning for Robot Manipulation, in: 2019 Third IEEE International Conference on Robotic Computing (IRC), 2019: pp. 590–595. https://doi.org/10.1109/IRC.2019.00120.

[57] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, MIT press, 2018. https://books.google.com/books?hl=en&lr=&id=uWV0DwAAQBAJ&oi=fnd&pg=PR7&dq=R.+S.+Sutton,+A.+G.+Barto,+Reinforcement+Learning:+An+Introduction,+2nd+Edition+&ots=mjoNq3Y4j7&sig=bYvIZtyIlEqf7EO6-jRmIM7Uq_Y (accessed June 25, 2024).

[58] C. Yan, Q. Zhang, Z. Liu, X. Wang, B. Liang, Control of Free-Floating Space Robots to Capture Targets Using Soft Q-Learning, in: 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2018: pp. 654–660. https://doi.org/10.1109/ROBIO.2018.8665049.

[59] DDPG-Based Adaptive Robust Tracking Control for Aerial Manipulators With Decoupling Approach | IEEE Journals & Magazine | IEEE Xplore, (n.d.). https://ieeexplore.ieee.org/abstract/document/9345436 (accessed June 25, 2024).

[60] X. Hu, X. Huang, T. Hu, Z. Shi, J. Hui, MRDDPG Algorithms for Path Planning of Free-Floating Space Robot, in: 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), 2018:

pp. 1079–1082. https://doi.org/10.1109/ICSESS.2018.8663748.

[61] Learning to Control a Free-floating Space Robot using Deep Reinforcement Learning | IEEE Conference Publication | IEEE Xplore, (n.d.). https://ieeexplore.ieee.org/abstract/document/8995991 (accessed June 25, 2024).

[62] Y. Li, X. Hao, Y. She, S. Li, M. Yu, Constrained motion planning of free-float dual-arm space manipulator via deep reinforcement learning, Aerospace Science and Technology 109 (2021) 106446. https://doi.org/10.1016/j.ast.2020.106446.

[63] B. Liang, Z. Chen, M. Guo, Y. Wang, Y. Wang, Space Robot Target Intelligent Capture System Based on Deep Reinforcement Learning Model, J. Phys.: Conf. Ser. 1848 (2021) 012078. https://doi.org/10.1088/1742-6596/1848/1/012078.

[64] H. Li, D. Gong, J. Yu, An obstacles avoidance method for serial manipulator based on reinforcement learning and Artificial Potential Field, Int J Intell Robot Appl 5 (2021) 186–202. https://doi.org/10.1007/s41315-021-00172-5.

[65] Y. Li, D. Li, W. Zhu, J. Sun, X. Zhang, S. Li, Constrained Motion Planning of 7-DOF Space Manipulator via Deep Reinforcement Learning Combined with Artificial Potential Field, Aerospace 9 (2022) 163. https://doi.org/10.3390/aerospace9030163.

[66] J. Blaise, M.C.F. Bazzocchi, Space Manipulator Collision Avoidance Using a Deep Reinforcement Learning Control, Aerospace 10 (2023) 778. https://doi.org/10.3390/aerospace10090778.

[67] M. Jepma, S. Nieuwenhuis, Pupil Diameter Predicts Changes in the Exploration–Exploitation Trade-off: Evidence for the Adaptive Gain Theory, Journal of Cognitive Neuroscience 23 (2011) 1587–1596. https://doi.org/10.1162/jocn.2010.21548.

[68] F. Yang, Z.H. Dong, X. Ye, Research on Equivalent Tests of Dynamics of On-orbit Soft Contact Technology Based on On-Orbit Experiment Data, J. Phys.: Conf. Ser. 1016 (2018) 012013. https://doi.org/10.1088/1742-6596/1016/1/012013.

[69] T. Rybus, Obstacle avoidance in space robotics: Review of major challenges and proposed solutions, Progress in Aerospace Sciences 101 (2018) 31–48. https://doi.org/10.1016/j.paerosci.2018.07.001.

[70] L. Santaguida, Z.H. Zhu, Development of air-bearing microgravity testbed for autonomous spacecraft rendezvous and robotic capture control of a free-floating target, Acta Astronautica 203 (2023) 319–328. https://doi.org/10.1016/j.actaastro.2022.11.056.

[71] J. Kang, Z.H. Zhu, L.F. Santaguida, Analytical and Experimental Investigation of Stabilizing Rotating Uncooperative Target by Tethered Space Tug, IEEE Transactions on Aerospace and Electronic Systems 57 (2021) 2426–2437. https://doi.org/10.1109/TAES.2021.3061798.

[72] J. Jia, Y. Jia, S. Sun, Preliminary design and development of an active suspension gravity compensation system for ground verification, Mechanism and Machine Theory 128 (2018) 492–507. https://doi.org/10.1016/j.mechmachtheory.2018.06.018.

[73] C.R. Carignan, D.L. Akin, The reaction stabilization of on-orbit robots, IEEE Control Systems Magazine 20 (2000) 19–33. https://doi.org/10.1109/37.887446.

[74] Z.H. Zhu, J. Kang, U. Bindra, Validation of CubeSat tether deployment system by ground and parabolic flight testing, Acta Astronautica 185 (2021) 299–307. https://doi.org/10.1016/j.actaastro.2021.04.028.

[75] W. Xu, B. Liang, Y. Xu, Survey of modeling, planning, and ground verification of space robotic systems, Acta Astronautica 68 (2011) 1629–1649. https://doi.org/10.1016/j.actaastro.2010.12.004.

[76] W. Xu, B. Liang, Y. Xu, C. Li, W. Qiang, A Ground Experiment System of Free-floating Robot For Capturing Space Target, J Intell Robot Syst 48 (2007) 187–208. https://doi.org/10.1007/s10846-006-9087-8.

[77] T. Boge, O. Ma, Using advanced industrial robotics for spacecraft Rendezvous and Docking simulation, in: 2011 IEEE International Conference on Robotics and Automation, 2011: pp. 1–4. https://doi.org/10.1109/ICRA.2011.5980583.

[78] I. Rekleitis, E. Martin, G. Rouleau, R. L'Archevêque, K. Parsa, E. Dupuis, Autonomous capture of a tumbling satellite, Journal of Field Robotics 24 (2007) 275–296. https://doi.org/10.1002/rob.20194.

[79] Q. Liu, X. Xiao, F. Mou, S. Wu, W. Ma, C. Hu, Study on a Numerical Simulation of a Manipulator Task Verification Facility System, in: 2018 IEEE International Conference on Mechatronics and Automation (ICMA), 2018: pp. 2132–2137. https://doi.org/10.1109/ICMA.2018.8484524.

[80] F. Mou, X. Xiao, T. Zhang, Q. Liu, D. Li, C. Hu, W. Ma, A HIL Simulation Facility for Task Verification of the Chinese Space Station Manipulator, in: 2018 IEEE International Conference on Mechatronics and Automation (ICMA), 2018: pp. 2138–2144. https://doi.org/10.1109/ICMA.2018.8484301.

[81] H. Liu, B. Liang, X. Wang, B. Zhang, Autonomous path planning and experiment study of free-floating space robot for spinning satellite capturing, in: 2014 13th International Conference on Control Automation Robotics & Vision (ICARCV), 2014: pp. 1573–1580. https://doi.org/10.1109/ICARCV.2014.7064550.

[82] M. Wilde, S. Kwok Choon, A. Grompone, M. Romano, Equations of Motion of Free-Floating Spacecraft-Manipulator Systems: An Engineer's Tutorial, Front. Robot. AI 5 (2018). https://doi.org/10.3389/frobt.2018.00041.

[83] K. Yoshida, Y. Umetani, Control of Space Manipulators with Generalized

Jacobian Matrix, in: Y. Xu, T. Kanade (Eds.), Space Robotics: Dynamics and Control, Springer US, Boston, MA, 1993: pp. 165–204. https://doi.org/10.1007/978-1-4615-3588-1_7.

[84] V. Dubanchet, D. Saussié, D. Alazard, C. Bérard, C.L. Peuvédic, Modeling and control of a space robot for active debris removal, CEAS Space J 7 (2015) 203–218. https://doi.org/10.1007/s12567-015-0082-4.

[85] W. Xu, B. Liang, C. Li, Y. Liu, Y. Xu, Autonomous target capturing of free-floating space robot: Theory and experiments, Robotica 27 (2009) 425–445. https://doi.org/10.1017/S0263574708004839.

[86] Optimal Concurrent Control for Space Manipulators Rendezvous and Capturing Targets Under Actuator Saturation | IEEE Journals & Magazine | IEEE Xplore, (n.d.). https://ieeexplore.ieee.org/abstract/document/9121679 (accessed June 26, 2024).

[87] H. Wang, On adaptive inverse dynamics for free-floating space manipulators, Robotics and Autonomous Systems 59 (2011) 782–788. https://doi.org/10.1016/j.robot.2011.05.013.

[88] A.M. Giordano, G. Garofalo, M. De Stefano, C. Ott, A. Albu-Schäffer, Dynamics and control of a free-floating space robot in presence of nonzero linear and angular momenta, in: 2016 IEEE 55th Conference on Decision and Control (CDC), 2016: pp. 7527–7534. https://doi.org/10.1109/CDC.2016.7799432.

[89] FANUC M-20iD/25 Robot - Light to Medium Payload Robot | FANUC, Fanucamerica (n.d.). https://www.fanucamerica.com/products/robots/series/m-20/m-20id-25 (accessed June 26, 2024).

[90] YOLOv5, PyTorch (n.d.). https://pytorch.org/hub/ultralytics_yolov5/ (accessed June 26, 2024).

[91] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic Policy Gradient Algorithms, in: Proceedings of the 31st International Conference on Machine Learning, PMLR, 2014: pp. 387–395. https://proceedings.mlr.press/v32/silver14.html (accessed June 26, 2024).

[92] T. Degris, M. White, R.S. Sutton, Off-Policy Actor-Critic, (2013). https://doi.org/10.48550/arXiv.1205.4839.

[93] C.J.C.H. Watkins, P. Dayan, Q-learning, Mach Learn 8 (1992) 279–292. https://doi.org/10.1007/BF00992698.

[94] R. Bellman, Dynamic Programming, Science 153 (1966) 34–37. https://doi.org/10.1126/science.153.3731.34.

[95] B. Hejase, U. Ozguner, Lyapunov Stability Regulation of Deep Reinforcement Learning Control with Application to Automated Driving, in: 2023 American Control Conference (ACC), 2023: pp. 4437–4442. https://doi.org/10.23919/ACC55779.2023.10155918.

[96] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, (2019). https://doi.org/10.48550/arXiv.1509.02971.

[97] R. Campa, K. Camarillo, M. Ceccarelli, Unit quaternions: A mathematical tool for modeling, path planning and control of robot manipulators, Robot Manipulators, M. Ceccarelli (Ed.), In-Teh (2008) 21–48.

[98] C. Chen, H. Tang, J. Hao, W. Liu, Z. Meng, Addressing action oscillations through learning policy inertia, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2021: pp. 7020–7027. https://ojs.aaai.org/index.php/AAAI/article/view/16864 (accessed September 16, 2024).

[99] R.L. Cognion, Rotation rates of inactive satellites near geosynchronous earth orbit, in: Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference, Maui Economic Development

Board Kihei, HI, 2014. https://amostech.com/TechnicalPapers/2014/Poster/COGNION.pdf (accessed June 26, 2024).

[100] D. Kucharski, G. Kirchner, F. Koidl, C. Fan, R. Carman, C. Moore, A. Dmytrotsa, M. Ploner, G. Bianco, M. Medvedskij, A. Makeyev, G. Appleby, M. Suzuki, J.-M. Torre, Z. Zhongping, L. Grunwaldt, Q. Feng, Attitude and Spin Period of Space Debris Envisat Measured by Satellite Laser Ranging, IEEE Transactions on Geoscience and Remote Sensing 52 (2014) 7651–7657. https://doi.org/10.1109/TGRS.2014.2316138.

[101] C. Qi, D. Li, W. Ma, Q. Wei, W. Zhang, W. Wang, Y. Hu, F. Gao, Distributed delay compensation for a hybrid simulation system of space manipulator capture, IEEE/ASME Transactions on Mechatronics 27 (2021) 2367–2378.