[SSC24-P5-22]

Optimal Control Design of Rotation Floating Space Robot during Grasping Operation: An RL Approach

Roshan Sah, **Kaushik Das**
Space Systems, Tata Consultancy Services(TCS)-Research
Bangalore, India
sahroshan11@gmail.com

ABSTRACT

This paper introduces an innovative model-free learning-based controller designed to comprehensively control an orbiting space robot engaged in proximity and grasping operations. Proximity control techniques are crucial for space robots, specifically robotic manipulators on a floating satellite base, facilitating tasks like in-orbit servicing and debris grasping. Traditional controllers need help managing these robots' coupled motion due to the satellite base's floating nature. While conventional controllers have been utilized for coupled control in nonlinear systems, their complexity increases with the growing degrees of freedom in the robot. In contrast, Model-free Reinforcement Learning (RL) has successfully mastered intricate policies within robotic manipulation. However, existing research has predominantly focused on controlling the space robotic arm, neglecting the satellite base. This paper addresses the vent by proposing a coupled controller for the space robotic arm and the satellite base orientation. This simultaneous control is essential for ensuring the proper functioning of onboard sensors and equipment with specific pointing requirements. The Proximal Policy Optimization (PPO) algorithm is employed in this study to control the position (3 DOF) and orientation (3 DOF) of the end-effector while also managing the orientation of the satellite base(3 DOF). To the best of the authors' knowledge, this paper marks the first application of a model-free RL method for the simultaneous 9 DOF control of a floating space robot. Furthermore, the paper proposes enhancements to standard reward functions in RL algorithms to enhance the learning algorithm's performance. The policy training is executed using a Pybullet framework environment, and the paper presents a trained policy performance of a rotation floating space robot at standard reward functions.

1 INTRODUCTION

The proliferation of space debris, encompassing spent rocket stages, non-operational satellites, and other fragments, has escalated exponentially within Earth's orbits over the past decade. The mitigation of this debris is imperative for the protection of currently operational satellites, the facilitation of practical future space exploration endeavors, and the sustainable utilization of orbital paths in the forthcoming years. Proximity in-orbit operations, such as satellite servicing,[1] refueling[2],[3] assembly of operational satellites, and active debris removal[4],[5] are crucial for long-term sustainability. Autonomous space robots are instrumental in achieving these objectives. These robots, which include robotic arms mounted on satellite bodies maneuvering in space, are pivotal for proximity operations with target objects. This area has seen significant research advancements over the last ten years.

Based on the control mechanisms, space robots are broadly classified into two categories: the conventional Free-Floating robots and the more advanced Rotation-Floating robots.[6] In a Free-Floating space robot (FFSR), only the robotic manipulator is actively controlled, while the satellite base remains unrestrained, allowing it to drift freely during proximity operations. Conversely, Rotation-Floating space robots (RFSR)[7]implement simultaneous control of both the satellite base's orientation and the position and orientation of the robotic manipulator. Due to the intricate dynamic coupling of the space robots, constructing accurate kinematic and dynamic models for the space robot presents significant challenges. Numerous methodologies have been proposed to address this issue, including the Virtual Manipulator framework, the Generalized Jacobian Matrix approach, and the Dynamically Equivalent Manipulator concept. Research indicates that both the kinematic and dynamic models of the RFSR are affected by variations or uncertainties in dynamic parameters.

The complexity of the space robot model in an unstructured environment makes target capture

challenging, complicating the controller design. The control of such floating space robots is difficult due to the highly coupled and non-linear dynamics. Previously, we have addressed potential research gaps and developed optimal and model-dependent learning-based controllers for floating space robots[8] along with MPC controllers[97] to control space robots. However, the complexity increases significantly when utilizing a dexterous and advanced robotic arm, such as the UR5, due to the extensive degrees of freedom in the coupled system. Consequently, we developed a simulator, Hardware In Loop (HILs),[10] and the corresponding controllers in PyBullet to simulate a higher degree of freedom system, specifically a 6 DOF floating satellite base and a 6 DOF UR5 robotic arm.

Deep reinforcement learning is a recent advancement in robot control that integrates deep learning with reinforcement learning. This model-free algorithm does not require prior knowledge of the problem model, allowing agents to interact with the environment to learn an optimal policy. However, numerous robotic challenges involve continuous, high-dimensional action spaces. T. P. Lillicrap et al. introduce the deep deterministic policy gradient algorithm (DDPG) to address these challenges. This approach integrates the actor-critic methodology with the advantages of deep Q-networks (DQN) to tackle continuous control problems effectively.[11]

Given the achievements of Deep Reinforcement Learning (RL) in diverse, intricate domains, it is now being applied to acquire policies for controlling floating space robots. Yan et al. utilized the Soft Q-learning (SQL) algorithm to develop stochastic energy and entropy-based policies for motion planning of free-floating space manipulators. Li et al. integrated constraints and obstacle avoidance into the motion planning of Free-floating robots using DDPG.[12] Du et al. implemented a DDPG-based position controller for a 3 DOF free-floating system, demonstrating enhanced efficiency through a pre-training phase.[13] Hu et al. applied a Multi-constrained Reward Deep Deterministic Policy Gradient (MRDDPG) for real-time trajectory planning of a dual-arm floating space robot aiming to reach a target in space with additional constraints.[14] Wu et al. addressed a similar problem with limited constraints using the Deep Deterministic Policy Gradient (DDPG) algorithm.[15] D'Ambrosio et al. work on a Proximal Policy Optimization (PPO) algorithm within Deep Reinforcement Learning (DRL) for space manipulator path planning during the motion-synchronization phase with a mission target. The PPO-optimized guidance law enables a 7-DoF robotic manipulator to maintain its end effector stationary relative to the target, with joint rates integrated and controlled by a model-based feedback linearization controller validated through extensive simulation testing.[16]

Wang et al. introduced a sophisticated multi-target trajectory planning strategy for a 6-DoF free-floating space robot, utilizing the Proximal Policy Optimization (PPO) algorithm. They incorporated the Action Ensembles Based on the Poisson Distribution (AEP) method, enhancing the policy's ability to approximate the optimal solution efficiently.[17] In our previous work,[18] We developed a Deep RL algorithm for RFSR for proximity control, which doesn't provide significant policy reward. We have extended the same work for grasping with more iterations and the time step in this paper.

The paper is organized as follows: Section (2) delineates the precise objective of the research and introduces the system alongside the simulator employed in the current study. Section (3) offers a comprehensive background on Reinforcement Learning and Proximal Policy Optimization (PPO) pertinent to the research. Section (4) follows with an in-depth discussion of the PPO policy-based controller. Section (5) elaborates on the simulation specifics, including the initial setup and the objectives for training the policy. Subsequently, Section (6) presents the results of the PPO-based controller and compares them with outcomes derived from standard reward functions. Finally, Section (7) delivers the concluding remarks and future work.

2 PROBLEM STATEMENT

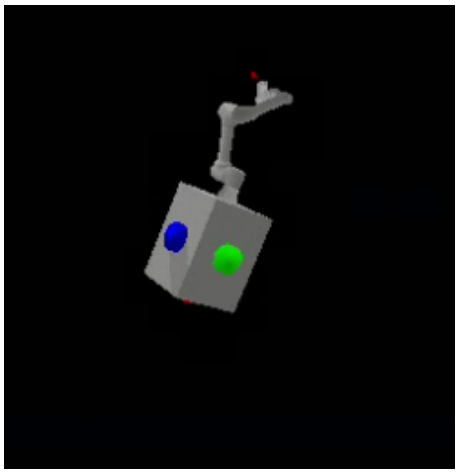

Figure 1: PyBullet Simulation Environment of RFSR.

Figure 1 presents a snapshot of the PyBullet sim-

ulation environment developed for this study. Gravity has been eliminated in the simulation to simulate the floating behavior of the base satellite. The grey cuboid structure represents a 6-DOF floating satellite, to which three orthogonal reaction wheels (red, green, and blue) have been integrated for attitude control. A 6-DOF robotic arm is mounted on the floating satellite to reach the red-colored target. In this study, a commercially available Universal Robot (UR) 5 arm is rigidly attached to the satellite base. The position and orientation of the target are assumed to be known for this work. The goal is to develop a model-free reinforcement learning-based controller to achieve three simultaneous objectives: guide the robotic arm's end-effector to the desired position and orientation (derived in real-time from the target), stabilize the attitude of the base satellite, and maintain its desired orientation. Proximal Policy Optimization (PPO) trains a policy to accomplish these objectives.

3 RL Based CONTROL

This section outlines foundational concepts for understanding the Proximal Policy Optimization (PPO) algorithm. The subsequent section details the controller for the problem and provides the necessary simulation parameters.

3.1 Markov Decision Process

A Markov Decision Process (MDP) is generally characterized by a tuple (S, A, R, P, γ). Here, S represents the collection of states the agent can visit, and A represents the set of actions available to the agent. P signifies the state transition probability, while R denotes the reward the agent receives when transitioning from state $(s_t \in S)$ to $(s'_t \in S)$. γ is the discount factor. The policy $\pi(s)$ determines the action taken in a state. The objective of the reinforcement learning algorithm is to identify an optimal policy π^* that maximizes the expected cumulative discounted rewards:

$$J(\pi) = E_\pi \left[\sum_t \gamma^t r(s_t, a_t) \right] \tag{1}$$

3.2 Proximal Policy Optimization

Proximal Policy Optimization[19] is a Policy Gradient technique where the policy and the associated value function are optimized and learned directly by maximizing the accumulated reward $J(s, \pi(s|a))$. Typically, the policy is represented as a neural network with a parameterized function $\pi_\theta(a|s)$. PPO uses the actor-critic approach, where the Actor approximates the policy function $\pi_\theta(a|s)$ to determine an action for a given state, and the Critic evaluates this action by estimating the action value function $Q_\phi(s, a)$. The policy parameters are adjusted to maximize the value function estimated by the Critic.

$$\theta \leftarrow \theta + \alpha_\theta \nabla_\theta J(s, \pi_\theta(s|a)) \tag{2}$$

The critic is updated using the TD error for action value.

$$\delta_t = r_t + \gamma Q_\phi(s', a') - Q_\phi(s, a) \tag{3}$$

$$\phi \leftarrow \phi + \alpha_\phi \delta_t \nabla_\phi Q_\phi(s, a) \tag{4}$$

In place of maximizing the accumulated reward, PPO instead maximizes the following clipped objective function:

$$J^{CLIP}(\theta) = E[min(r(\theta)\hat{A}_{\theta_{old}}(s, a), clip(r(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_{\theta_{old}})] \tag{5}$$

where $\hat{A}$ is the estimated advantage function and $r(\theta)$ is the probability ratio between the new and old policies:

$$r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} \tag{6}$$

A restraint is put on large changes in policy by forcing this ratio $r(\theta)$ to stay within a small interval around 1, in the range $[1 - \epsilon, 1 + \epsilon]$, where ϵ is a hyperparameter. This further improves the training stability. The clipped objective function is then augmented with an error term on the value estimation and an entropy term for better performance. The final objective function for PPO thus looks as follows:

$$J^{PPO}(\theta) = E[J^{CLIP}(\theta) - c_1(V_\phi(s) - V_{target})^2 + c_2 H(s, \pi_\theta)] \tag{7}$$

In this context, c_1 and c_2 are hyper-parameters. For a more comprehensive explanation of the algorithm, readers can consult the original PPO paper by Schulman et al. (2017). With the foundational knowledge of PPO established, the following section discusses the implementation of PPO for the specific task of controlling a 9 DOF Rotational Floating space robot to reach a target.

4 CONTROLLER

For the current problem, the goal is threefold: manage the position and orientation of the end-effector to align with the target and maintain the base satellite's orientation at the desired level. The

entire issue is simulated in the PyBullet physics simulator. The overall controller features a two-layered architecture, where the learning algorithm constitutes the outer layer, and a velocity-tracking-based PD controller forms the inner loop (not MPC as it is a heavy controller to handle such simulation in Pybullet[20]). The PPO policy translates the controllable and observable states into the desired velocities for the joints and reaction wheels.[18] The built-in PyBullet controller then uses these velocities as a reference to calculate the joint and reaction wheel torques, which ultimately serve as the control input to the coupled non-linear system. PyBullet solves the dynamics and provides the integrated and updated states of the entire system. This process repeats continuously. This section outlines the MDP employed and the parameters/hyperparameters utilized for the outer loop of the controller shaped by the PPO algorithm. Figure 2 visually represents the entire section.

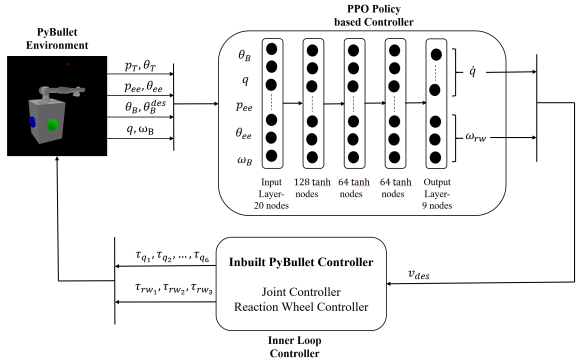

Figure 2: Controller Structure

4.1 States and Action

The states chosen for the MDP formulation consider both objectives and physical observability requirements. We aimed to encompass relevant and observable states comprehensively. Thus, the state space defined for the PPO policy is:

$$s_t = \theta_B, q, p_{ee}, \theta_{ee}, \omega_B \tag{8}$$

These states, including θ_B, representing base satellite orientation in quaternions, and others like joint angles q, end effector position p_{ee}, orientation θ_{ee} (expressed as quaternions), and base satellite angular velocity ω_B, are all observable using basic sensors such as IMU, joint encoders, and cameras provided by the PyBullet simulator.

The state space totals a dimensionality of 20.

The action space consists of actuators necessary to control the coupled space robot system to achieve specified goals. In our setup, these actuators are joint motors and reaction wheels, governing their respective velocities. Therefore, the action space is defined as:

$$a_t = \omega_{rw}, \dot{q} \tag{9}$$

Here, ω_{rw} denotes the angular velocities of three orthogonal reaction wheels, and $\dot{q}$ represents the six-dimensional vector of joint rates for the UR5 manipulator.

4.2 Reward Function

For the current problem, θ_B^{des} is the desired orientation of the base satellite. p_T and θ_T are the inertial position and orientation of the target, which also serve as the desired pose for the UR5 end-effector. The corresponding error metrics thus are:

$$\begin{aligned} e_{\theta_B} &= \theta_B^{des} \otimes \theta_B^{-1} \\ e_p &= p_T - p_{ee} \\ e_{\theta_{ee}} &= \theta_T \otimes \theta_{ee}^{-1} \end{aligned}$$

The orientation errors for the base and end-effector are calculated as the quaternion difference between the current orientation with that of the desired orientation. The agent is assumed to have reached the goal state when $|e_p| < 0.15, |e_{\theta_{ee}}| < 0.15$ and $|e_{\theta_B}| < 0.15$, which is when it gets a large positive reward.

The reward function aims to minimize the errors mentioned. Typically, the reward is a known function (polynomial, logarithmic, or exponential) of the distance between the current and goal state, scaled appropriately with constants. However, using such reward functions to control the 9 DOF of the space robot proved inadequate. With three independent control objectives (in 9 DOF) to meet simultaneously, a single reward function is needed to guide the agent effectively. It was noted that reducing position errors increased orientation errors and vice versa. The reward function should penalize wrong actions and motivate the agent as it nears the goal (when error norms are below 0.3). Thus, a single reward function could only address some requirements due to the need to achieve three different goals simultaneously.

Consequently, a new method for distributing rewards was implemented. Rewards are scheduled based on errors in the three control dimensions, with weightage distributed accordingly. If one error's norm is below 0.3, it receives a smaller negative reward compared to errors with norms above 0.3. This scheme was thus adopted:

$$r_t = \begin{cases} (-|e_p| - |e_{\theta_{ee}}| - |e_{\theta_B}|)/10, & \text{if } |e_p| < 0.3, |e_{\theta_{ee}}| < 0.3, |e_{\theta_B}| < 0.3 \\ (-|e_p| - |e_{\theta_{ee}}| - 3|e_{\theta_B}|)/5, & \text{if } |e_p| < 0.3, |e_{\theta_{ee}}| < 0.3, |e_{\theta_B}| > 0.3 \\ (-|e_p| - 3|e_{\theta_{ee}}| - |e_{\theta_B}|)/5, & \text{if } |e_p| < 0.3, |e_{\theta_{ee}}| > 0.3, |e_{\theta_B}| < 0.3 \\ (-|e_p| - 3|e_{\theta_{ee}}| - 5|e_{\theta_B}|)/10, & \text{if } |e_p| < 0.3, |e_{\theta_{ee}}| > 0.3, |e_{\theta_B}| > 0.3 \\ (-3|e_p| - |e_{\theta_{ee}}| - |e_{\theta_B}|)/5, & \text{if } |e_p| > 0.3, |e_{\theta_{ee}}| < 0.3, |e_{\theta_B}| < 0.3 \\ (-3|e_p| - |e_{\theta_{ee}}| - 5|e_{\theta_B}|)/10, & \text{if } |e_p| > 0.3, |e_{\theta_{ee}}| < 0.3, |e_{\theta_B}| > 0.3 \\ (-5|e_p| - 3|e_{\theta_{ee}}| - 2|e_{\theta_B}|)/10, & \text{if } |e_p| > 0.3, |e_{\theta_{ee}}| > 0.3, |e_{\theta_B}| < 0.3 \\ (-3|e_p| - 2|e_{\theta_{ee}}| - 5|e_{\theta_B}|)/10, & \text{if } |e_p| > 0.3, |e_{\theta_{ee}}| > 0.3, |e_{\theta_B}| > 0.3 \end{cases}$$

After getting this reward, the agent is also penalized for a higher angular velocity of the base satellite as well as higher fluctuations in the commanded velocity. The final reward function thus looks like

$$r_t = r_t - 0.25(0.1|\omega_B| + 0.1|a_t - a_{t-1}|) \tag{10}$$

This improvised reward function was compared with standard norm-based and logarithmic-based reward functions. However, the standard reward functions failed to converge to a solution while the current scheduling-based reward function successfully did.

$$r_{norm} = -a|e_p| - b|e_{\theta_{ee}}| - c|e_{\theta_B}| \tag{11}$$

$$r_{log} = -log(a|e_p|) - log(b|e_{\theta_{ee}}|) - log(c|e_{\theta_B}|) \tag{12}$$

The standard reward functions fail to vary a, b, c parameters and instead keep them constant. However, when three independent control objectives are to be simultaneously met, the current work proposes that these scaling factors should also depend on their corresponding errors for better performance.

4.3 Network Structure

To train the PPO policy and value function, actor and critic networks were set up as multilayer perceptrons with three hidden layers containing 128, 64, and 64 nodes, respectively. The *tanh* activation function was used, proving more stable than *ReLU* for this problem. The buffer size was 512, and the policy update was limited by a maximum KL Divergence of 0.075. The learning rate was 0.0003, and the PPO clipping factor was 0.2.

5 SIMULATION DETAILS

The zero gravity environment is set for the simulations as shown in Fig. 1. The mass of the satellite base is kept to be almost the same as the mass of the arm in order to increase the complexity and test the efficacy of the PPO policy in adverse conditions. The satellite base is at $p_B = \{0, 0, 0.35\}$ with its axes parallel to the inertial axes. The target is kept stationary for the scope of this work and is stationed at $p_T = \{-0.25, 0.6, 1.25\}$. This is, thus, the goal for the end-effector. The desired orientation is such that the end-effector should grasp the target from below, while the base satellite axes should remain aligned with the inertial axes. The criteria for the success of an episode while training is that the norm of all three errors should simultaneously be less than 0.15. Thus, the agent is assumed to have reached the goal state when $|e_p| < 0.15, |e_{\theta_{ee}}| < 0.15$ and $|e_{\theta_B}| < 0.15$.

5.1 Training

The average duration of an episode depended on the total number of time steps during which the PPO policy was trained. An episode resets when all three objectives are met or at 5000 steps, whichever comes first. The simulation shows that the episode initially starts with 5000 steps (indicating no convergence) but decreases as training progresses. Once the policy is fully trained, the average episode duration stabilizes at around $3500-4000$ steps, which is the time the agent needs to reach the goal state according to the learned policy.

The saturation of the mean episode reward at a higher value proves that the policy has been successfully learned. The value of the loss that PPO aimed to optimize has decreased significantly from the initial training episodes. Despite some increases in loss due to exploration, the agent has learned to navigate back towards the goal states. The loss value for the critic network, which learns the value function for the policy, significantly decreases from a high initial value to nearly zero. This outcome provides evidence that the PPO policy has successfully learned to achieve the three control objectives simultaneously.

5.2 Comparison with Standard Reward Functions

This subsection presents the mean episode length and mean episode reward for two standard reward functions. With the network and remaining hyperparameters kept constant, the agent was trained using a logarithmic reward function. The mean episode length remains 5000 for most of the training period. Since this is the episode limit, it is clear that the agent failed to learn a successful policy. A similar trend is observed in the mean episode reward. Unlike earlier results, the rewards do not saturate when using the logarithmic reward function. This again demonstrates that a fixed logarithmic reward function is unable to train a policy for achieving three independent control objectives in 9 DOF.

The average episode length stays at 5000 throughout the entire training period when a standard norm-based reward function is applied. Given that this is the maximum limit for an episode, it is clear that the agent failed to learn an effective policy. Additionally, the rewards converge, indicating that the policy met the necessary objectives using the norm-based reward function within 4.9 million steps.

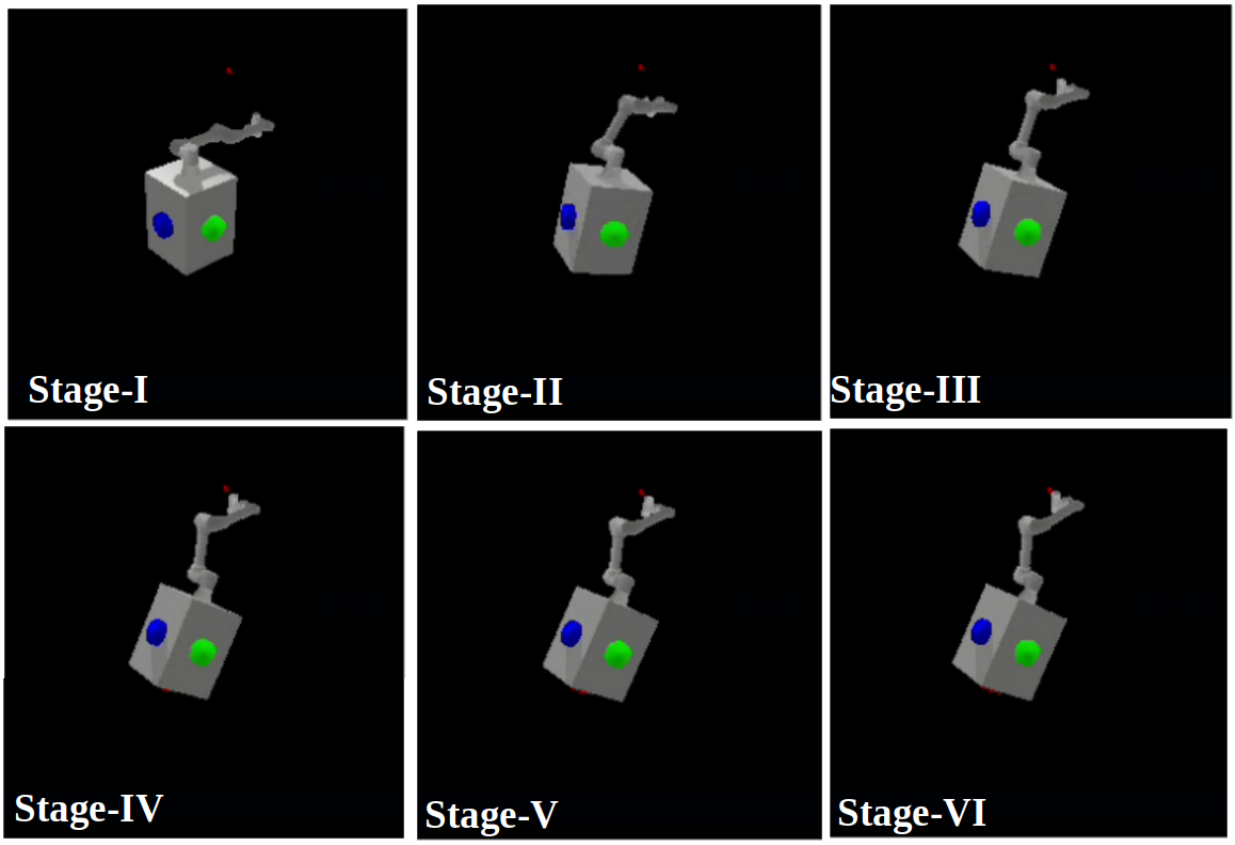

Figure 3: Staging of Grasping operation of Space Robot by RL.

6 RESULTS & DISCUSSION WORK

This section presents the results for the control objectives mentioned before. The training results are presented first, followed by the results obtained from the trained policy.

6.1 Trained Space Robot

This subsection shows a schematic diagram of a staging of grasping operation by Space Robot by RL method at 4.9 million steps, which is shown in figure 3. Stage I is the initial stage of the space robot with the target (Marked as a red box), which is initially at rest. Stage II shows that the UR5 has started moving closer to the target as all the criteria are met by the PPO of RL. Stage III shows the arm of the robot moving close to the target with all the joints moving close to the target; the movement caused by the arm is balanced by the reaction wheel of the satellite to have control over the base of the satellite. Stages IV and V show the movement of the arm, forming a trajectory to catch the target in which the base of the satellite is inclined toward the left and is controlled by the reaction wheel of the satellite to overcome the momentum caused by the arm. The stage-VI shows that the end effector (gripper not attached) is moving at the exact position of the targets, which a successful capturing of the target body can be done. The schematic diagram of the movement of the space robot plots is shown in the next subsections.

6.2 Trained Policy Performance

The successfully trained policy, utilizing scheduled rewards, was tested under the initial conditions of the PyBullet simulation to verify its performance. This subsection presents the robot's results to demonstrate the effectiveness of the learned policy. Figure 4 shows the end-effector's inertial position v/s the target position. The end-effector has reached sufficiently close to the target, satisfying the success criterion ($|e_p| < 0.15$). Additionally, as shown in Figure 5, the end-effector and target orientations align towards the end of a successful episode ($|e_{\theta_{ee}}| < 0.15$).

Figure 6 displays the 6-DOF motion of the base satellite in response to the movement of the UR5 manipulator arm as it approaches the target in the desired orientation.

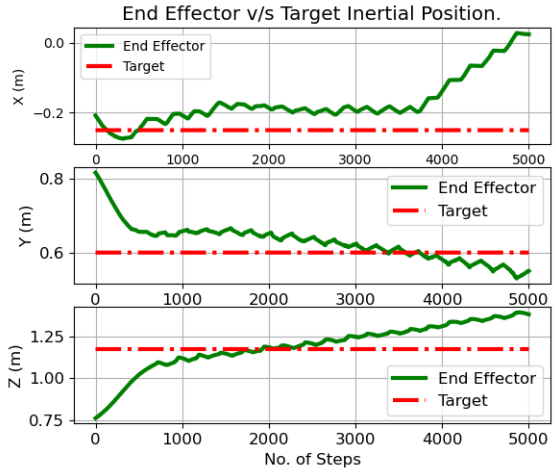

Figure 4: End-Effector vs Target states: Inertial Positions.

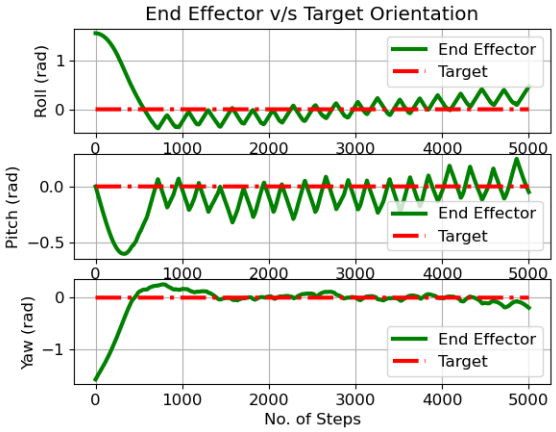

Figure 5: End-Effector vs Target states: Euler angles (deg).

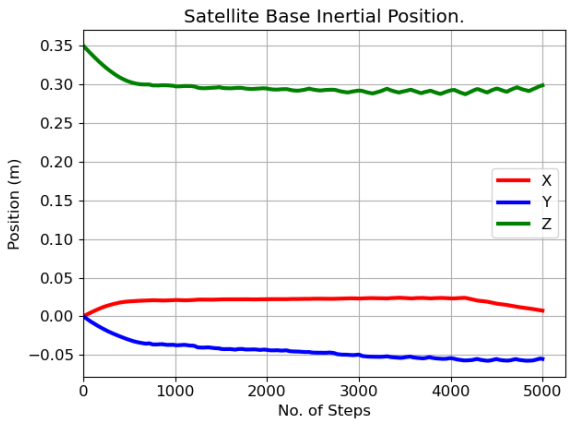

Figure 6: Satellite Base motion: Inertial Positions.

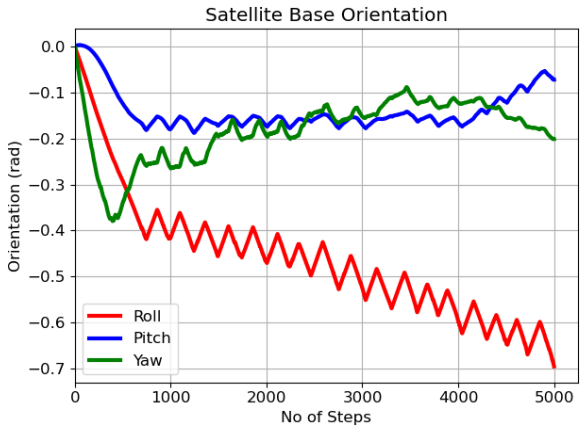

Figure 7: Satellite Base motion: Orientation.

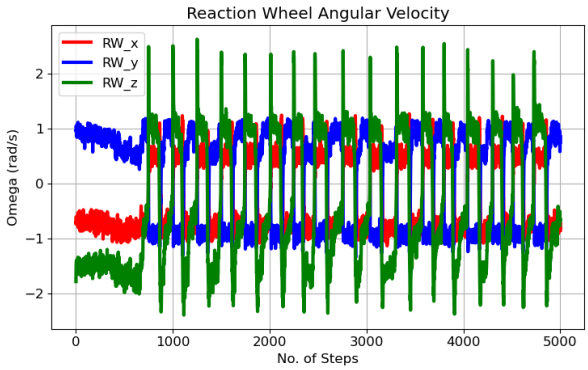

Figure 8: Actuator motions: Reaction wheels.

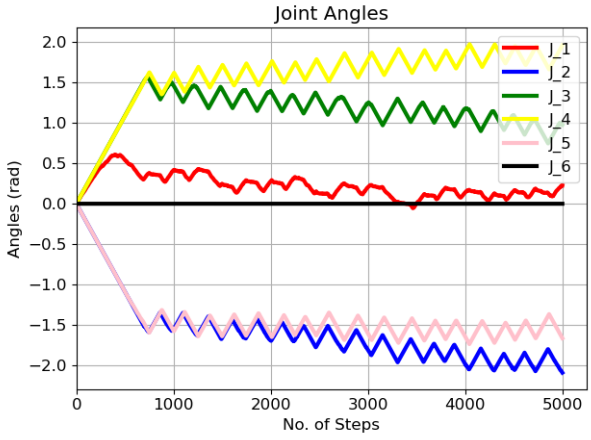

Figure 9: Actuator motions: Arm Joints.

As previously mentioned, one of the control objectives of this study was to maintain the base satellite's orientation during arm motion to ensure the proper functioning of onboard sensors. It is evident from Fig. 7 that the base orientation has remained

close to its desired orientation, such that the criteria for success ($|e_{\theta_B}| < 0.15$) is met.

Figure 8 displays the commanded velocities calculated by the PPO policy, which are then provided to the inner loop PyBullet controller for execution. As depicted in Fig. 8, the reaction wheel velocities are shown. Additionally, Fig. 9 illustrates the necessary arm joints for the end-effector to successfully reach its target.

7 CONCLUSION & FUTURE WORK

The paper introduces a pioneering application of model-free Reinforcement Learning (RL) for the simultaneous control of a robotic arm and its satellite base in orbital proximity operations. Addressing a gap in the existing literature primarily focusing on robotic arm control alone, this study employs the Proximal Policy Optimization (PPO) algorithm to manage the intricate dynamics of a floating space robot's 9 degrees of freedom (DOF). The research demonstrates significant advancements in autonomous control capabilities by integrating advancements in reward function design and utilizing the Pybullet-ROS framework for simulation. Results indicate successful policy learning within 4.9 million timesteps, outperforming standard reward function approaches under comparable conditions. Moreover, enhancements to standard reward functions were explored to optimize the learning process, demonstrating improved performance compared to conventional RL approaches. This work marks a substantial step towards enhancing the autonomy and efficiency of space robotic missions, particularly in tasks requiring precise positioning and orientation control amidst complex orbital environments. In our previous work, we have developed a HILS setup,[21] orbital environment for LEO,[22] and end-of-life calculation of debris after disposal,[23] thruster design[24252627] with optimal thrust control[2829] and space robot ADCS control system[5],[30] which can be used as a parameter for the development of RL method for space robot in real-time.

Future research could expand upon several avenues suggested by this study: 1. Investigating more sophisticated reward-shaping techniques could enhance the RL algorithm's learning efficiency and robustness. 2. Extending the applicability of the proposed approach to more complex environments and tasks, such as dynamic orbital scenarios or multi-robot cooperative missions, LEO atmospheric drags[31323334] could broaden its practical utility. 3. Exploring real-world implementation challenges and validating the learned policies in physical space environments remains a crucial next step toward operational deployment.

References

[1] A. Flores-Abad, O. Ma, and K. Pham, "A review of robotics technologies for on-orbit services," 2013.

[2] R. Sah, R. Srivastava, R. Lima, V. Vatsal, and K. Das, "Numerical modeling of liquid propellant mass transfer with sloshing during on-orbit refueling," in *2024 IEEE Aerospace Conference*, pp. 1–14, 2024.

[3] R. Sah, R. Srivastava, S. Saha, and K. Das, "Liquid propellant slosh analysis with mass transfer during on-orbit refueling operation," *AIAA SCITECH 2024 Forum*, pp. 1–10. doi: 10.2514/6.2024-2229.

[4] R. Sah, R. Srivastava, and K. Das, "Design development of debris chaser small satellite with robotic manipulators for debris removal," *AIAA SCITECH 2022 Forum*, 2022. doi: 10.2514/6.2022-2109.

[5] R. Srivastava, R. Sah, and K. Das, "Attitude determination and control system for a leo debris chaser small satellite," in *AIAA SCITECH 2022 Forum*, 2022. doi: 10.2514/6.2022-0519.

[6] R. Srivastava, R. Sah, and K. Das, "Free floating and rotation floating approaches for control of space robots: A comparative study," in *2022 IEEE Aerospace Conference (AERO)*, pp. 1–12, 2022. doi: 10.1109/AERO53065.2022.9843709.

[7] R. Srivastava, R. Sah, and K. Das, "Nonlinear model predictive control of rotation floating space robots for autonomous active debris removal," *IFAC-PapersOnLine, 22nd IFAC Symposium on Automatic Control in Aerospace ACA 2022*, vol. 55, no. 22, pp. 147–152, 2022. doi: 10.1016/j.ifacol.2023.03.025.

[8] R. Srivastava, R. Lima, R. Sah, and K. Das, "In-orbit control of floating space robots using a model dependant learning based methodology," in *2023 IEEE Aerospace Conference*, pp. 1–10, 2023. doi: 10.1109/AERO55745.2023.10115732.

[9] R. Srivastava, R. Sah, and K. Das, "Model predictive control of floating space robots for close proximity on-orbit operations," *AIAA SCITECH 2023 Forum*, 2023. doi: 10.2514/6.2023-0157.

[10] R. Srivastava, R. Sah, S. Saha, R. Lima, S. Shukla, and K. Das, "Hardware-in-the-loop verification of model predictive control of rotation floating space robot for fine manipulation," *AIAA SCITECH 2024 Forum*.

[11] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019.

[12] C. Yan, Q. Zhang, Z. Liu, X. Wang, and 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)*, pp. 654–660, IEEE, 2018.

[13] D. Du, Q. Zhou, N. Qi, X. Wang, and Y. Liu, "Learning to control a free-floating space robot using deep reinforcement learning," in *2019 IEEE International Conference on Unmanned Systems (ICUS)*, pp. 519–523, IEEE, 2019.

[14] X. Hu, X. Huang, T. Hu, Z. Shi, and 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)*, pp. 1079–1082, IEEE, 2018.

[15] Y.-H. Wu, Z.-C. Yu, C.-Y. Li, M.-J. He, B. Hua, and Z.-M. Chen, "Reinforcement learning in dual-arm trajectory planning for a free-floating space robot," *Aerospace Science and Technology*, vol. 98, p. 105657, 2020.

[16] M. D'Ambrosio, L. Capra, A. Brandonisio, S. Silvestrini, and M. Lavagna, "Redundant space manipulator autonomous guidance for in-orbit servicing via deep reinforcement learning," *Aerospace*, vol. 11, no. 5, 2024.

[17] S. Wang, X. Zheng, Y. Cao, and T. Zhang, "A multi-target trajectory planning of a 6-dof free-floating space robot via reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3724–3730, IEEE, 2021.

[18] R. Srivastava, R. Lima, R. Sah, and K. Das, "Deep reinforcement learning based control of rotation floating space robots for proximity operation in pybullet," *IEEE SMC 2023*, 2023.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[20] R. Sah, R. Srivastava, V. Vatsal, R. Lima, and K. Das, "Target grasp-conditioned whole-body control of satellite manipulators using neural fields with mpc," in *2024 IEEE Aerospace Conference*, pp. 1–12, 2024.

[21] R. Sah, R. Srivastava, and K. Das, "Development of on-ground hardware in loop simulation facility for space robotics," *Small Satellite Conference*, 2023. https://doi.org/10.48550/arXiv.2306.03785.

[22] R. Sah, R. Srivastava, and K. Das, "Constellation design of remote sensing small satellite for infrastructure monitoring in india," *Small Satellite Conference*, 2021. doi: 10.48550/arXiv.2107.09253.

[23] R. Sah, R. Srivastava, and K. Das, "The post mission disposal analysis of pslv debris at de-orbited altitude," *AIAA SCITECH 2023 Forum*, 2023. doi: 10.2514/6.2023-2551.

[24] R. Sah, R. Srivastava, and K. Das, ""design and analysis of cold gas thruster to de-orbit the pslv debris"," *Small Satellite Conference*, 2022. doi: 10.48550/arXiv.2208.03773.

[25] R. Sah, R. Srivastava, and K. Das, "Iterative design of rcs thruster for debris chaser small satellite based on δv budget," *Small Satellite Conference*, 2022. https://digitalcommons.usu.edu/smallsat/2022/all2022/270/.

[26] P. Sherpaili, R. Sah, S. Hegde, and B. B. Chaudhary, "Design of chemical propellant thruster to deorbit nanosatellite: Studsat-ii," in *Proceedings of the XVII Vibration Engineering & Technology of Machinery Conference VETOMAC, Dec 15-12, DMAE, IOE, Pulchowk, Nepal*, pp. 1–12, 2022. doi: 10.48550/arXiv.2212.11992.

[27] R. Sah, P. Sherpaili, A. Anand, and S. Hegde, "Design of the Propulsion System of Nanosatellite: StudSat2," *arXiv e-prints*, p. arXiv:2107.10992, July 2021. doi: 10.48550/arXiv.2107.10992.

[28] R. Sah, R. Srivastava, S. Saha, and K. Das, "Optimal finite thrust rendezvous maneuvers of chaser spacecraft towards capturing target object," *AIAA SCITECH 2024 Forum*.

[29] R. Sah, R. Srivastava, and K. Das, "Design of low thrust controlled maneuvers to chase and de-orbit the space debris," *Conference on Artificial Intelligence (AI) Enabled Aerobots and Hydrobots*, 2022. doi: 10.48550/arXiv.2204.00674.

[30] R. Srivastava, R. Sah, and K. Das, "Attitude and in-orbit residual magnetic moment estimation of small satellites using only magnetometer," *Small Satellite Conference*, 2021. doi: 10.48550/arXiv.2107.09257.

[31] R. Sah and S. Ghosh, "Numerical investigation of supersonic turbulent flow over open cavities," *International Journal of Fluid Mechanics and Thermal Sciences*, vol. 7, no. 2, p. 22, 2021. doi: 10.11648/j.ijfmts.20210702.11.

[32] Y. Sowmyashree, D. I. P. Aishwarya, S. Spurthy, R. Sah, B. V. Pratik, H. V. Srikanth, and R. Suthan, "Study on effect of semi-circular dimple on aerodynamic characteristics of NACA 2412 airfoil," *AIP Conference Proceedings*, vol. 2204, 01 2020. doi: 10.1063/1.5141572.

[33] R. Sah, P. Sherpaili, A. Anand, and G. Das, "Engineering design and analysis of recuperative gas combustion chamber," *ERCAM*, 2017. doi: 10.13140/RG.2.2.34603.92964.

[34] A. Kumar, R. Sah, A. Saha, S. Karmakar, A. Roy, and V. M. Reddy, "Investigation of highly swirl stabilized spray combustion using low-cost computational techniques," *Proceeding of the 2nd National Aero Propulsion Conference at IIT Kharagpur, India*, 2018.