2023-2024-MECH5051/5052 SENIOR DESIGN

CATCHING FLOATING OBJECTS IN SPACE
USING REINFORCEMENT LEARNING ALGORITHM TO CONTROL ROBOT ARM

GROUP 21
Directed by Dr. Janet Dong

TEAM MEMBER:
Linfeng Liu
Jinjia Guo
Xinyu Wang

Date of Submission: 04/24/2024

---

*The three authors of this report have made equal contributions to this project.

Content

Catching Floating Objects in Space Using Reinforcement Learning Algorithm to Control Robot Arm

Linfeng Liu[1], Jinjia Guo[1] and Xinyu Wang[1]
Advisor: Dr. Janet Dong

1. College of Engineering and Applied Science, Mechanical and Materials Engineering, University of Cincinnati, Cincinnati, OH 45221-0070

Abstract. The project focuses on developing a robotic arm controlled by a reinforcement learning algorithm to solve the critical problem of space debris. Through extensive research and investigation, comparing and selecting from traditional algorithms, visual feedback algorithms, LSTM, and other algorithms, we finally determined a vision-based system guidance for the robotic arm, using Yolo and SAC (Soft Actor-Critic) models to autonomously Algorithms for identifying and grasping objects in space. The main innovations of this project include integrating two different deep learning models and customized reward functions and introducing the L2 regularization function to optimize the neural network and achieve the generalization of the optimized model. The main results prove that the robotic arm can successfully grasp free-floating objects in a zero-gravity environment simulated in the Pybullet physics engine, and the performance is improved through model iterative training of static and dynamic objects. The project lays the foundation for advancing autonomous space systems, with plans for further enhancements and more realistic test scenarios. This initiative represents a major advance in space debris management, combining advanced artificial intelligence with engineering solutions to solve a pressing global problem.

Keywords: Aerospace, space debris, moving objects, grasping planning, SAC algorithm, target detection

1. Introduction

Satellites in orbit underpin our modern lives. They are used in many areas and disciplines, including space science, Earth observation, meteorology, climate research, telecommunication, navigation, and human space exploration. They offer a unique perspective, a resource for collecting scientific data, commercial opportunities and various essential applications and services, which lead to unrivalled possibilities for research and exploitation. However, in the past decades, with increasing space activities, a new and unexpected hazard has started to emerge space debris.

One might ask, “What is orbital debris?” Although we don’t see space junk in the sky, beyond the clouds and as far away as the naked eye can see, it enters Low Earth Orbit (LEO). LEO is an orbiting space junkyard. There are millions of pieces of space junk flying in low Earth orbit. Most orbital debris consist of human-generated objects such as spacecraft debris, tiny specks of paint on a spacecraft, rocket parts, satellites that no longer function, or explosions of orbiting objects traveling at high speeds through space.

Most "space junk" moves very fast, reaching speeds of up to 18,000 miles per hour, which is nearly seven times faster than a bullet. Due to the velocity and volume of debris in low Earth orbit, current and future space-based services, exploration, and operations pose safety risks to people and property in space and on Earth. There are many reasons why low-Earth orbit has developed into an orbital graveyard. For example, the deliberate destruction of China's Fengyun-1C spacecraft in 2007 and the accidental collision of U.S. and Russian spacecraft in 2009 increased the number of large orbital debris in low-Earth orbit by about 70%, posing a huge threat to operating spacecraft. Brings greater risk of collision. In low Earth orbit. There is no international space law to clean up debris in low-Earth orbit. Low-Earth orbit is now considered the world's largest garbage dump, and removing space debris from low-Earth orbit is costly because the space debris problem is so severe — there are nearly 6,000 tons of material in low-Earth orbit.

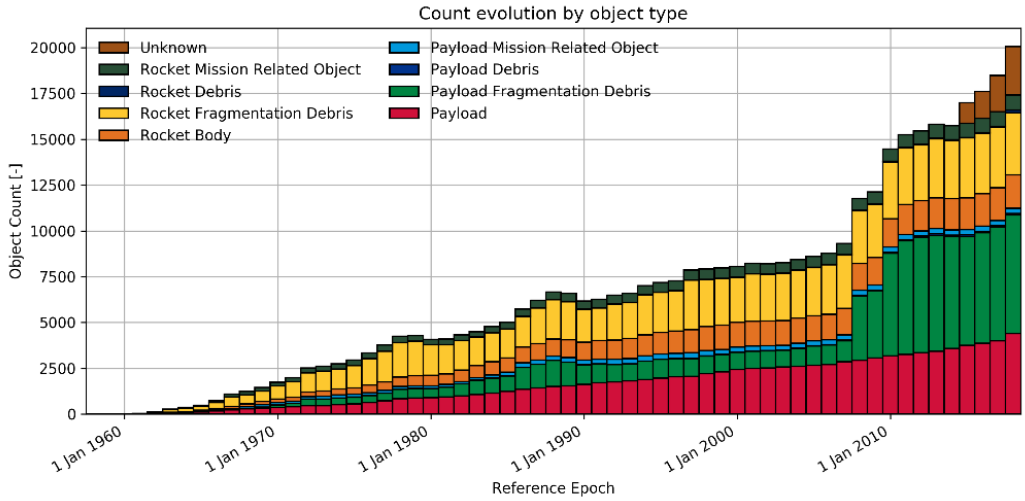


Figure 1 The amount of space debris is growing rapidly over time.

Project Objectives: The primary objective of this project is to develop a robust, reliable control system for a robotic arm, using advanced machine learning algorithms to enable it to autonomously grasp free-floating objects in a simulated space environment. The project aims to address the growing issue of space debris, which poses significant risks to satellites and other space operations.

The project aims to develop and evaluate a vision-guided robotic arm system for space debris management in a simulated weightless environment. The system leverages two key technologies: a YOLO (You Only Look Once) model to track objects through camera inputs, and a SAC (Soft Actor-Critic) model to control the arm's movements. The success of the project is gauged by three primary criteria:

- Accuracy: The robotic arm's ability to precisely identify and capture targeted space debris.
- Safety: The system's capability to operate without generating additional debris or causing harm to the surrounding space environment.
- Reliability: Consistent and dependable performance of the robotic arm across a variety of simulated conditions, ensuring robust operation

The project makes a significant contribution to space debris management by developing a vision-guided robotic arm controlled through advanced reinforcement learning algorithms.

The primary contributions of this work include:

1. Integration of Advanced Technologies:

   The project innovatively combines Yolo for object detection and SAC (Soft Actor-Critic) for arm control, effectively enabling the robotic arm to autonomously identify and grasp objects in a simulated space environment.

2. Algorithmic Enhancements:

   Customized reward functions are implemented to optimize the neural network. These enhancements not only improve the performance of the SAC model but also enhance the generalization capabilities of the system.

3. Demonstrated Success in Simulated Environments:

   Using the Pybullet physics engine, the project demonstrates that the robotic arm can successfully grasp free-floating objects in a zero-gravity environment. The iterative training scenarios involving static and dynamic objects have proven to enhance model performance over time.

2. Related Existing work and Research status.

2.1 Research status of moving object grabbing

The problem of grasping moving objects is a very hot topic in the field of robotic arm control. Many scholars are working on solving this problem. In this study, the methods to solve this problem are roughly divided into the following two types: one is a prediction-based grabbing method. The specific idea is to first collect the movement trajectory of the object, predict the future movement trajectory of the object through the prediction network, and then Let the robotic arm plan a path that allows the end of the robotic arm to meet the moving object. The second is a tracking-based grabbing method. The idea is to detect the moving state of the object in real time, and then use the motion planner to plan the path in real time to ensure that the distance between the end of the robotic arm and the moving object remains unchanged. That is, the moving object is stably tracked before grabbing. The following is a detailed explanation of the current research status of mobile object grasping planning based on these two methods.

2.1.1 Prediction-based catching methods

The prediction-based grasping method first collects the movement trajectory of the object, predicts the future movement trajectory of the object through the prediction network, and then plans a path for the robotic arm that allows the end of the robotic arm to meet the moving object. In early research, computers were in their early stages of development and did not have high computing power and real-time image processing capabilities. At the same time, the performance of the sensor also limited the real-time detection capabilities of moving objects. Therefore, most of the research on the grasping of moving objects at that time focused on the grasping of simple moving objects. In this way, the movement trajectory of the object can be collected, and then the movement path of the robotic arm can be planned offline. For example, taking linear motion object grabbing as an example, Park [1] et al. considered the speed of the conveyor belt and the positions of the grasping parts and robots, and used dynamic programming to solve the optimal motion path of the robotic arm to achieve robotic arm alignment. Tracking of parts on conveyor belts. For objects with curved motion trajectories, Kimura [2] and others used the function fitting method to predict the future trajectory of the ball by using the mobile phone's trajectory of the ball moving in a parabolic trajectory, and then planned the trajectory of the robotic arm to grasp the object. ball task. Chen [3] and others used the global convergence homotropy algorithm to derive the optimal motion trajectory of a space robot docking with a moving target. This method does not require prior planning of the motion trajectory of the end of the robotic arm and can directly obtain the optimal trajectory from the target information. Wang Shuguo [4] and others expressed the movement trajectory of the object in a three-dimensional rectangular coordinate system. Then use the polynomial interpolation method to predict the movement trajectory of the object and then grab it. This type of method predicts the future movement trajectory or arrival position of the moving object, and then controls the robotic arm to arrive in advance.

With the rapid development of computers. The improvement of computer computing power enables sensors to detect the motion status of objects in real time and plan the movement path of the robotic arm in parallel. As a result, the shortcomings of offline planning have been improved. Scholars have proposed to detect the motion status of objects in real time and then predict the trajectory online and plan the meeting point. For example, in Aghili [5]'s research on grabbing moving objects, he used the extended Kalman filter as a prediction model to obtain the trajectory of the moving object, using the movement time of the moving object, the angle between the end of the manipulator and the moving object, and the moving distance. , and the acceleration of the moving object are used as constraints to construct a loss function to solve the optimal grasping trajectory of the robotic arm. In the research of Riley [6] et al., a 60Hz binocular vision system was used to obtain the position information of the moving ball in real time, and then used this information to fit its trajectory based on the parabolic model, and then predicted the ball and the end of the robotic arm in real time. Where to meet. Then plan the robot's point-to-point movement path to realize the function of catching the ball. Senoo [7] and others used high-speed cameras to quickly capture the position of moving objects to obtain their motion characteristics and online predict the coordinates of the hitting point of the robotic arm, and then controlled the end of the robotic arm to successfully achieve hitting the ball with the robotic arm. Due to the development of sensors, Binocular vision cameras have been fully used in moving object detection. Houshang [8] uses a vision-based method to predict the trajectory of moving objects using an autoregressive discrete-time model and uses a self-tuning adaptive controller to control the end of the robotic arm to track the trajectory, thereby realizing the grasping of moving objects by the robotic arm.

2.1.2 Tracking-based catching methods

The idea of the tracking-based grasping method is to detect the moving state of the object in real time, and then use the motion planner to plan the path in real time to ensure that the distance between the end of the robotic arm and the moving object remains unchanged, that is, to steadily track the moving object, and then grab it.

Whether it is a predictive crawling method, or a tracking based crawling method. The detection of object status is essential, and the accuracy of obtaining object motion characteristics directly affects subsequent grabbing performance. Among today's detection methods, the use of cameras (i.e. vision) to obtain moving motion features has incomparable advantages over other methods, such as fast response speed and good real-time performance. Therefore, many studies on moving object grasping are based on the visual system to obtain the status information of the moving object. Many of the tracking-based methods use 3D images to calculate the grasped 3D pose. For example, Allen et al. [9] use stereoscopic optical flow to calculate the 3D pose of a moving object and use it to control the robotic arm to match the pose in real time until successful grasping. Bing et al. [10] proposed using CAD models to match the feature points of moving objects detected in real time by cameras, and then based on the tracking results, they proposed a real-time motion planning method to achieve robotic arm grabbing. Asada [11] et al. used a fixed marker combined with a binocular camera installed at the end of the robotic arm to estimate the motion state of the moving object, and then controlled the robotic arm to track the moving object based on the image Jacobian matrix.

As the speed of moving objects increases, the visual serving method used by the above scholars exposes the problem of slow convergence speed. That is, the object moves outside the working space of the robot arm, and the robot arm still fails to track the object and fails to grasp it. To address this problem, navigation-based grasping path planning has been proposed by scholars for grasping moving objects. For example, for the situation where the visual servo convergence is slow during the tracking phase when the robot arm grabs a moving object. Su [12] et al. proposed a navigation method that can dynamically adjust the speed to ensure stable tracking of objects at the end of the robotic arm, and then switch back to the conventional motion planning algorithm for grasping operations. Mehrandezh [13] and others proposed that a combination of navigation and conventional trajectory tracking methods can be used to match the speed of the end of the robotic arm with the moving object, and then quickly grab the moving object. Keshmiri [14] and others combined the image features of moving objects with navigation methods to solve the grasping problem. Agah [15] and others directly used navigation methods to solve the problem and based on the motion characteristics of the robot arm, they proposed the mechanical arm motion dynamics model and the principle of speed matching improve the speed command in the conventional navigation method, thus solving the problem of the speed mismatch between the end of the robot arm and the moving object in the pure navigation method.

2.1.3 Reinforce learning and development.

In fact, whether they are prediction-based grasping methods or tracking-based grasping methods, they all rely on accurate models of the robotic arm and prior knowledge of the environment. However, in actual grasping task applications, it is impossible to accurately obtain the mechanical motion model and related parameters of the environment. Therefore, traditional moving object grabbing methods have problems such as insufficient stability and weak generalization. Therefore, some scholars have conducted in-depth research on grasping methods that do not rely on robotic arm models, some of which are methods based on reinforcement learning. Deep reinforcement learning is an effective method for robotic arms to acquire control strategies autonomously through trial and error. This method can perform some more complex grabbing tasks by processing raw sensor input information, such as image information. For example, in [16], it is proposed to combine image convolutional networks with reinforcement learning control, using large-scale image data from monocular cameras to train large-scale convolutional neural networks so that the robotic arm can autonomously identify objects and estimate the probability of successful grasping. Probability. Experimental results show that learning-based crawling methods have great application prospects. In the literature [17[18]], for disorderly placed objects, they used reinforcement learning to implement pre-grasping operations of the robotic arm, such as pushing or moving obstacles in front of the objects to be grasped. Moreover, the trained control strategy can be deployed to new robotic arms. Literature [19] proposes some evaluation indicators for various grasping methods based on deep reinforcement learning. In the literature [20], a QT-opt framework based on deep reinforcement learning was proposed, which is vision-based closed-loop control. They used 7 real robots to conduct approximately 800 hours of testing on this framework within 4 months. train. Finally, a complete set of crawling operation strategies was obtained, which can respond to different crawling situations and dynamically respond to interference during the crawling process. Experimental results show that when this framework is applied to actual robots, it achieves a grasping success rate of approximately 96% for objects outside the field of view.

2.2 Application of reinforcement learning in robotic arm control

With the development of deep learning in recent years, reinforcement learning has been increasingly used in the field of robot control. The high integration of deep reinforcement learning, and robot control allows robots to achieve many complex tasks that were difficult to complete before. For example, [21] uses DRL to implement an end-to-end control method that can directly map sensor readings to operations without requiring a specific task model or strategy. Experimental results show that this method can be directly trained in the real world using the DRL model algorithm of offline policy (of-policy) and realizes the dexterous operation of the manipulator to rotate the valve. Reference [22] applied the SAC reinforcement learning algorithm to the case of quadruped robot walking, achieving state-of-the-art performance. Literature [23] explored distributed and asynchronous policy reinforcement learning. They used multiple robots to conduct policy learning and data collection in parallel, shortening the training time, improving sample utilization efficiency, and allowing the robot to successfully open the door. Literature [24] proposes an ADR learning framework based on reinforcement learning, which realizes that the strategies trained in the simulation environment can be used to solve complex manipulation problems on actual robots and enables the humanoid manipulator to successfully rotate the Rubik's Cube. To train reinforcement learning algorithms faster, Ref. [24] performs deep reinforcement learning in simulations to learn potential states, and then adapts them to real robots using unlabeled real robot data. This method successfully transfers learned skills to actual robots. And realize the precise operation of building blocks.

2.3 Innovation and application of this project

This article aims at the problem of robotic arms grabbing free-floating objects in the universe. The project proposes a free-floating object grabbing system based on the reinforcement learning algorithm Soft-Actor Critic (SAC). The system is divided into two parts: the target detection module and the robotic arm control module based on deep reinforcement learning. Due to the dynamic characteristics of moving objects, the target detection system first needs to enable the Kinect camera to output depth information, and then combine the YOLO [25] algorithm to detect the three-dimensional coordinates of the moving object in real time. The robotic arm control system adopts a tracking + grabbing integrated strategy to successfully achieve the grabbing goal by maintaining the relative difference in position and attitude between the end of the robotic arm and the moving object. Realize the integrated operation of controlling the end of the robotic arm based on Soft Actor-Critic (SAC) to stably track the moving object first and then grab it. At the same time, the control module is separated from the target detection module, making it more convenient to directly apply it from simulation training to real robotic arms. In addition, some optimizations have been made to the reinforcement learning algorithm SAC used, and the reward function has been redesigned. Here are our main innovations and contributions:

1. Integration of Advanced Technologies:

The project innovatively combines Yolo for object detection with the SAC (Soft Actor-Critic) algorithm for arm control. This integration enables the robotic arm to autonomously identify and grasp objects in a simulated space environment, showcasing a sophisticated application of machine learning and computer vision technologies.

2. Algorithmic Enhancements:

The paper introduces customized reward functions to optimize the neural network. These algorithmic enhancements not only boost the performance of the SAC model but also significantly improve the system's ability to generalize across different scenarios, enhancing its reliability and effectiveness.

3. Demonstrated Success in Simulated Environments:

Utilizing the Pybullet physics engine, the project demonstrates that the robotic arm is capable of successfully grasping free-floating objects in a zero-gravity environment. The iterative training scenarios with both static and dynamic objects have been shown to progressively improve the model's performance, validating the effectiveness of the training approach and the robustness of the technology in simulated conditions.

3. Engineering Standards and Codes & constrains.

3.1 Research on Applicable Standards and Codes

The engineering standards and codes applicable to the project were rigorously researched to ensure compliance and safety. Key standards included:

ISO 13482[27]:

This standard pertains to robots and robotic devices, specifically safety requirements for personal care robots, which are analogous to the safety measures needed for space robotic systems.

IEEE 1872-2015[28]:

This standard defines ontologies for robotics and automation, providing a framework for the integration of AI and machine learning models within robotic systems.

3.2 Identification and Application of Standards and Codes

The standards were applied throughout the design, simulation, and testing phases of the project to ensure that all aspects of the robotic arm's operation were safe, reliable, and effective:

Safety Compliance:

Ensuring that the robotic arm's design and operational protocols adhere to the ISO 13482 safety requirements, focusing on minimizing risks associated with robotic interaction with objects in space.

Documentation and Reporting:

Following IEEE 1872-2015, comprehensive documentation was maintained, detailing the development process, machine learning models, and testing results. This documentation ensures transparency and facilitates future audits and reviews.

Environmental Considerations:

The project also adhered to environmental standards to minimize any potential harm that could arise from the deployment of robotic systems in space. This included compliance with guidelines on electromagnetic emissions [29] and material safety [30] to prevent further space debris creation.

3.2 Constraints

The constraints of the project involving a robotic arm controlled by a reinforcement learning algorithm for space debris management can be assessed from various aspects as outlined:

Economic Constraints

Economic considerations are primarily centered around the costs of computational resources and software development, rather than physical manufacturing or space operations. The investment focuses on the necessary tools and technologies to support complex algorithm testing.

Environmental Constraints

The project has minimal direct environmental impacts since there are no physical launches or hardware deployments. The main environmental consideration involves the indirect effects such as the electricity consumption for powering the computers running the simulations.

Sustainability Constraints

In a simulated setup, sustainability mainly pertains to efficient use of computational resources and minimizing the carbon footprint associated with high-performance computing tasks. It emphasizes the reduction of electronic waste through optimized software practices and energy-efficient hardware.

Manufacturability Constraints

Manufacturability constraints are less immediate in a simulation-based project. The focus shifts towards the design and development of algorithms and systems that can be theoretically scaled and adapted for eventual physical production, rather than current manufacturability.

Ethical Constraints

Ethical considerations revolve around the development and testing of autonomous systems in a controlled environment without real-world consequences. This stage allows for the exploration of ethical dilemmas and the establishment of guidelines for future deployment in space, where the actions of autonomous systems have significant implications.

Health and Safety Constraints

The health and safety issues are confined to ensuring the cybersecurity and integrity of the simulation systems. There are no risks of physical injury or environmental harm, which reduces the complexity of safety management compared to real-world testing.

Specifications Constraints

The project's specifications focus on the accuracy and reliability of the simulations to mimic real-world conditions as closely as possible. This includes ensuring that the models perform well within the parameters set by the simulation environment and are capable of being adapted based on simulated feedback without the real-world trial and error.

4. Time Schedule

The project was meticulously planned with a detailed timeline that outlined each phase of development, from initial research to final testing. This timeline was essential for keeping the project on track and ensuring all objectives were met within the set deadlines.

4.1 Major Milestones and Phases

In the Initial Research and Design Phase, spanning the first four weeks, we conducted an extensive literature review and feasibility study. This initial phase included designing and simulating components of a robotic arm, setting a solid foundation for our project's technical framework.

During Weeks 5 to 8, in the Algorithm Development and Initial Testing Phase, we focused on testing various control algorithms and developing Yolo and Soft Actor-Critic (SAC) models. These models were crucial for enhancing the robotic arm's ability to track and grasp objects. We conducted initial simulations in a controlled environment to assess the basic functionality of our designs.

In the Refinement and Advanced Simulations Phase, which lasted from Week 9 to Week 14, we refined our machine learning models based on feedback from the initial tests. We then engaged in advanced simulations, rigorously testing the robotic arm under conditions that mimic space, handling fixed, 2D-moving, and 3D-moving objects to ensure versatility and robustness in diverse operational scenarios.

Finally, in the Evaluation and Documentation Phase during Weeks 15 and 16, we conducted a final evaluation of the project outcomes in relation to our initial objectives. We assessed different grippers' performance to determine the most effective one for our needs. We completed all necessary documentation, including the project report and technical papers, ensuring a comprehensive wrap-up of our project activities and outcomes.

4.2 Monitoring Progress

During our project, we diligently monitored progress through various methods to ensure we stayed on track. We held weekly team meetings to address immediate concerns and align on progress, ensuring all members were on the same page. Additionally, we regularly updated our project advisor and engaged in review sessions where we assessed our milestones against the project timeline. At the midpoint of the project, we conducted a formal review which provided a comprehensive evaluation of our status. This review was pivotal in allowing us to make significant adjustments to our design and approach, effectively addressing any delays or challenges we encountered throughout the project lifecycle. These strategies helped us maintain momentum and adapt our strategies as needed to meet our objectives.

4.3 Gantt chart of this project

	Week 1-2	Week 3-4	Week 5-6
Discussion	\	How to separate the project to tasks	Coding for further simulation
Researching	Find possible methods	Find proper environment and algorithm	\
Coding	\	Basic coding for environment	Complete total simulation
Meeting	Discuss about the feasibility	Talk about the camera position	How to deploy the algorithm to kuka

Figure 2 The Gantt chart of first 6 weeks.

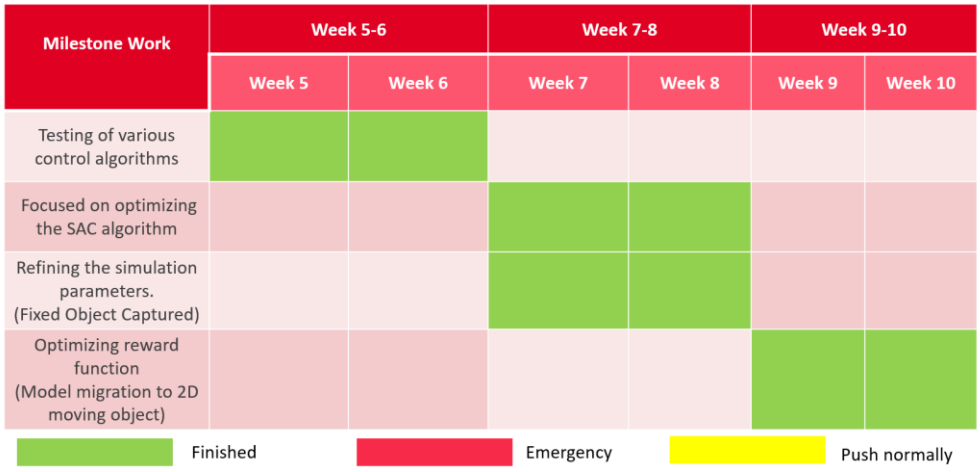


Figure 3 The Gantt chart of the middle 6 weeks.

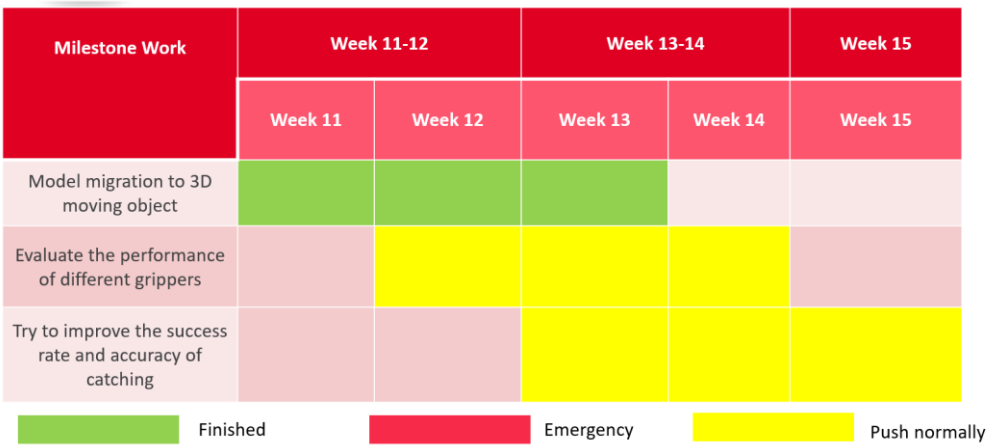


Figure 4 The Gantt chart of the last 5 weeks.

5. Alternative methods and selection bias

In this part, we will discuss the alternatives we proposed and how we did the selection.

5.1 Problem statement

The core objective of our project is to design a sophisticated vision-guided control system that empowers a robotic arm to accurately capture a free-floating object within a three-dimensional space. The challenge lies in not only establishing a simulation environment that closely mirrors the complexities of such a task but also in fine-tuning the algorithms that will enable precise tracking and manipulation. Our end goal is to achieve seamless coordination between visual input and mechanical action, allowing the robotic arm to locate a cube set through camera supervision and execute a flawless grasp, thereby overcoming the obstacles presented by an environment with no gravitational anchoring. To achieve this goal, we imported the kuka iiwa 14 robotic arm into the pybullet virtual physical environment and simulated the outer space environment by setting the gravity to 0. And apply different algorithms to the robot arm to observe the grabbing results of the robot arm.

5.2 Robot arm kinematics model

Robotic arm kinematics studies the motion characteristics of the robotic arm. In kinematics research, the force generated when the robot arm moves or the force exerted by its movement is generally regarded as an ideal state, which is 0. In the scope of robotic arm kinematics research, the joint data of the operating arm is the focus of the research. For example: acceleration, position, velocity, and all higher-order derivatives of position variables (including derivatives with time or other variables). Common robotic arms in industry are composed of connecting rods. In the kinematics of the robotic arm, each link on the robotic arm is set up with a coordinate system. With the existence of the coordinate system, the relationship between the links is very easy to express. In addition, the relative relationship between each link after being connected through joints is the focus of robotic arm kinematics research. This chapter explains how to use the manipulator joint variables as independent variables to describe the functional relationship between the position and attitude of the manipulator end effector and the manipulator base.

The robot arm can number the links in sequence, for example, the fixed base is link 0, the first movable link is link 1, and so on, the link at the end of the robot arm is link n.

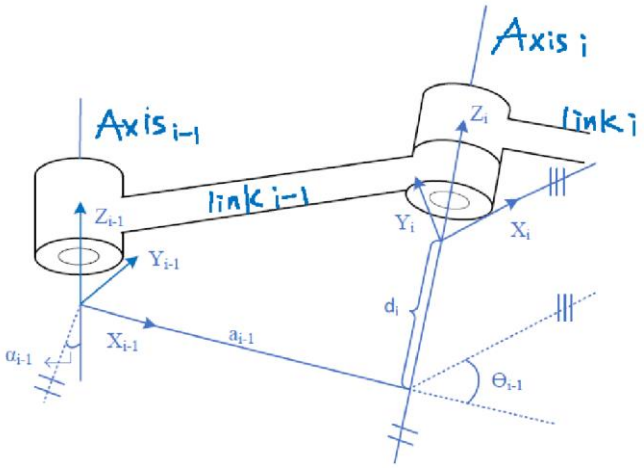


Figure 5 Connecting Rod Relationship Description Diagram

The distance between any two axes in three-dimensional space is a certain value. As shown in Figure 5, the kinematics of the manipulator defines the length of the connecting link as the length of the common vertical line between joint $axis_{i-1}$ and joint $axis_i$, which is a_{i-1}. This is the first parameter that defines the relative position of the two joint axes.

The second parameter is the connecting rod torsion angle. Assume that a plane is made, and the plane is perpendicular to the common perpendicular between the two joint axes. Then the joint $axis_{i-1}$ and the joint $axis_i$ are projected onto the plane, and the plane rotates from the $axis_{i-1}$ according to the right-hand rule. α_{i-1} steering $axis_i$ measures the angle between the two axes and uses the rotation angle α_{i-1} to define the torsion angle of connecting $link_{i-1}$.

Generally, the two links of a robotic arm are the same as a human arm. There is a common joint axis between them. Usually, it is expressed by a parameter, which is called the connecting rod offset. The offset distance of the connecting rod on the joint $axis_i$ is recorded as di. Another parameter is used to describe the angle between the rotation of two adjacent connecting links around the common axis. This parameter is called the joint angle and is recorded as θ_i. In general, the parameters describing the relative position between the manipulator axes are the above four, as shown in Figure 5.

The physical structure of the robotic arm consists of multiple joints and connecting rods between joints. Each joint has a certain degree of freedom and can translate or rotate. When modeling a robotic arm, to facilitate the description of the position of each joint, a coordinate system is set at each joint to obtain a joint chain. For the positional relationship of joint chains, the most widely used method is a matrix representation method between joints proposed by Denavit and Hartenberg in 1955, namely the D-H method [31]. The D-H method describes four main parameters of the robot arm joint: connecting link length a, torsion angle α, offset angle d; and joint angle θ.

Therefore, each link of the robot can be described by four kinematic parameters, two of which are used to describe the link itself, and the other two parameters are used to describe the connection relationship between the links. Usually, for rotating joints, θ is the joint variable, and the other three link parameters are fixed; for moving joints, d is the joint variable, and the other three link parameters are fixed. This rule that uses connecting rod parameters to describe the motion relationship of a mechanism is called the Denavit-Hartenberg method, or the D-H parameter method for short. In the D-H parameter method, to obtain the transfer matrix between a pair of adjacent connecting rods, that is, to obtain the transformation of the same three-dimensional coordinate in the three-dimensional space between the coordinate system $\{i\}$ and the coordinate system $\{i-1\}$. Usually, the transfer matrix is obtained through two translations and two rotations. After homogenizing the transfer matrix, the transformation relationship matrix ${}^{i-1}_{i}T$ is obtained.

$$
\begin{aligned}
{}^{i-1}_{i}T &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha_{i-1} & -sin\alpha_{i-1} & 0 \\ 0 & sin\alpha_{i-1} & cos\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} cos\theta_i & -sin\theta_i & 0 & 0 \\ sin\theta_i & cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} cos\theta_i & -sin\theta_i & 0 & a_{i-1} \\ sin\theta_i cos\alpha_{i-1} & cos\theta_i cos\alpha_{i-1} & -sin\alpha_{i-1} & -sin\alpha_{i-1}d_i \\ sin\theta_i sin\alpha_{i-1} & cos\theta_i sin\alpha_{i-1} & cos\alpha_{i-1} & cos\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

In the formula, ${}^{i-1}_{i}T$ represents the transformation relationship between two adjacent coordinate systems, that is, the coordinate system $\{i\}$ relative to the coordinate system $\{i-1\}$. Through ${}^{i-1}_{i}T$, the point coordinates in the coordinate system can be converted to the coordinate system $\{i-1\}$ coordinates, that is:

$$
{}^{i}P = {}^{i-1}_{i}T \ {}^{i}P
$$

Among them, ${}^{i}P$ is the vector coordinate of the point P in the coordinate system $\{i\}$; ${}^{i-1}P$ is the position vector of the point P in the coordinate system $\{i-1\}$

5.3 Different ways of robot arm control

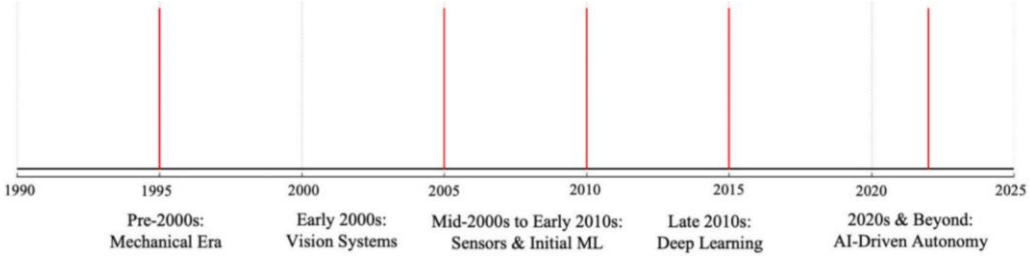


Figure 6 Development trajectory of catching objects.

In Jahanshahi's review [32], the combination of artificial intelligence, especially through deep learning, reinforcement learning, transfer learning, and convolutional neural networks, has revolutionized robotic mastery. These advances facilitate autonomous, efficient, and sophisticated manipulation in challenging outer space environments, moving from traditional mechanical grips to complex systems powered by advanced algorithms. This transition emphasizes the critical integration of sensory perception, mastery planning, and execution mechanisms, enhancing the ability of robots to sense, interact with, and manipulate objects using unprecedented precision and adaptability.

Here, we select four typical algorithms for comparison.

5.3.1 Traditional Algorithm

The traditional algorithm in paper [33] provides an optimal trajectory planning scheme for a space robot to capture a tumbling object, which is inherently more complex due to the dynamic coupling between the robot manipulator and its base. The innovative approach begins by calculating the Path Independent Workspace (PIW), free of dynamic singularities, and the Path Dependent Workspace (PDW) using the proposed algorithm. Motion equations derived from Euler dynamics equations and quaternion representation predict the grasping point's long-term motion on the tumbling object.

Leveraging the PIW and the predicted motion trajectories, the scheme meticulously plans the end-effector's trajectory to intercept the object with zero relative velocity, thereby avoiding any impact at the moment of capture. To ensure a successful capture, three proposed criteria determine the optimal capture occasion, focusing on safety, reliability, and speed. The trajectory is then optimized by minimizing a cost function constrained by the acceleration magnitude.

The advantages of this method include a highly precise and robust planning scheme that can navigate the complex dynamics of space robotics. The consideration of dynamic coupling and singularities, along with the use of optimization for trajectory planning, showcases a deep understanding of the unique challenges in space environments.

However, the disadvantages are notable. The algorithm may be computationally intensive due to the complexity of the calculations involved, particularly when predicting long-term motion and planning trajectories. This complexity can also make implementation difficult, as it requires high accuracy in actuators and fast response from the control system. Additionally, while the method is optimized for the space environment, its generalization to unpredictable or unknown environments remains questionable.

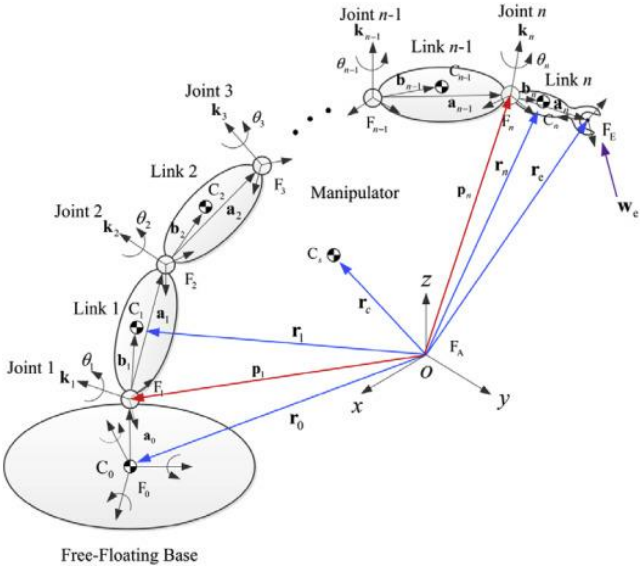


Figure 7 Mathematical model of multi-degree-of-freedom manipulator.

5.3.2 Robot Guild By vision

Rouleau, G propose a series of algorithms [34] aimed at enhancing autonomous robotic grasping in space. The key methods discussed include vision-based motion estimation, trajectory generation, and redundancy resolution for a robotic manipulator. These algorithms are designed to autonomously intercept and grasp moving objects, such as satellites and space debris. The unique aspect of this research lies in its experimental testing facility, which utilizes a helium airship to simulate free-floating objects in space. This innovative approach allows for practical evaluation and refinement of the algorithms under conditions that mimic the challenges faced in outer space.

The proposed methods bring several notable advantages. Firstly, the innovative use of a helium airship as a stand-in for free-floating space objects in the experimental setup is a creative and effective method for realistic testing. The autonomy of the algorithms enhances the capabilities of robots in space applications, crucial for tasks like satellite servicing and space debris removal. Furthermore, the practical testing of these algorithms in an experimental setup marks an important step from theoretical research to real-world application, providing a robust platform for demonstrating the feasibility and effectiveness of the technologies.

However, the approach is not without its challenges. The complexity and cost of developing such a sophisticated system, particularly at the experimental stage, are significant. The systems require precise equipment and entail high maintenance costs. Moreover, ensuring real-time effectiveness of these algorithms in the dynamic and unpredictable environment of space poses a substantial challenge. Finally, the transition from simulation-based testing to real-world application can introduce unforeseen technical hurdles and limitations, complicating the deployment and scalability of the solutions in actual space conditions.

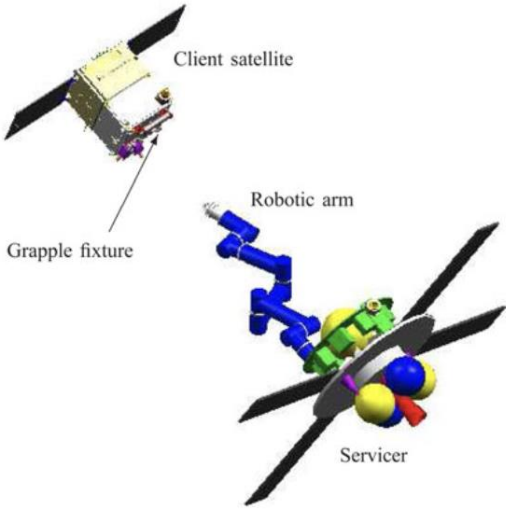


Figure 8 Vision based guild robot catching system.

5.3.3 LSTM-based Flight Trajectory Prediction

In addressing the complexities of flight trajectory prediction within Air Traffic Management (ATM), one study [35] introduces a predictive model based on Long Short-Term Memory (LSTM) networks. The LSTM network is composed of four interactive layers, designed to capture long-term dependencies that are critical for accurate trajectory forecasting. The model employs a sliding window approach to maintain the continuity of the state sequences and to ensure that dynamic dependencies between adjacent states are not compromised.

The predictive capacity of the LSTM model is exploited to forecast both 3-D (time stamp, latitude, longitude) and 4-D (time stamp, latitude, longitude, altitude) trajectories. The data for model training and validation was sourced from Automatic Dependent Surveillance-Broadcast (ADS-B) ground stations, which provide high-fidelity flight data.

The methodology introduced in this study leverages the advanced capabilities of LSTM networks, achieving high accuracy in flight trajectory prediction by capturing long-term dependencies. The use of sliding windows within the LSTM framework maintains the continuity of state sequences and respects the dynamic dependencies crucial for the integrity of the prediction, especially in the complex and variable realm of ATM where precise trajectory forecasting is critical for safety.

However, the LSTM network's computational intensity presents challenges, particularly for real-time applications requiring rapid processing. Moreover, the substantial volume of data required to train the LSTM model effectively may limit its applicability in situations where data is scarce or not sufficiently detailed, potentially impacting the model’s performance and generalization to various flight conditions.

5.3.4 RL-based vision trajectory prediction

Andrea F [36] explores the application of advanced reinforcement learning (RL) techniques, specifically Trust Region Policy Optimization (TRPO) and Deep Q-Network with Normalized Advantage Functions (DQN-NAF), for autonomous

robot control in dynamic workspaces. The research compares these methods against other prominent algorithms like Deep Deterministic Policy Gradient (DDPG) and Vanilla Policy Gradient (VPG) through both simulated and real-world experiments. This comparison focuses on how these algorithms manage to learn manipulation tasks such as reaching random position targets and pick & placing objects. The methods are evaluated based on their ability to learn from scratch, without the need for user demonstrations or task-specific domain knowledge, highlighting their model independence and adaptability to changes in the robot's dynamic and geometric models.

The primary advantage of the RL approaches discussed (TRPO and DQN-NAF) lies in their robustness and adaptability, which are critical in dynamic and unpredictable environments. These algorithms demonstrate the capacity to learn manipulation tasks autonomously, with minimal human intervention. Their model-free nature ensures that they perform well even when there are significant changes to the robot's physical characteristics, such as link lengths, masses, and inertia. This flexibility makes them highly suitable for real-world applications where conditions and tasks can vary dramatically.

However, these algorithms also have notable disadvantages. The complexity of modern robots with high degrees-of-freedom results in large dimensional state spaces that are challenging to learn effectively. This often necessitates the use of example demonstrations to initialize policies and mitigate safety concerns during the training phase. Additionally, when performing dimensionality reduction, it is difficult to fully model all dimensions, which necessitates finding an appropriate representation for the policy or value function to achieve practical training times for physical hardware. These challenges can significantly hinder the scalability and efficiency of deploying these algorithms in real-time applications.

In conclusion, while TRPO and DQN-NAF provide promising avenues for enhancing robot control in dynamic settings, their practical application is limited by the complexities associated with learning large state spaces and the need for careful policy representation and initialization.

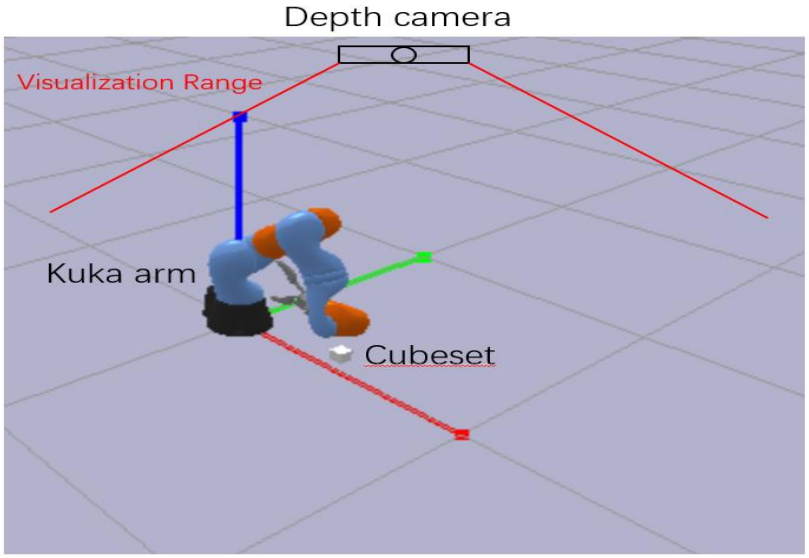


Figure 9 RL-based simulation environment.

5.3.5 Comparison of different methods

Here we use weight table to make selection. In the evaluation of robotic control algorithms, we compared Traditional Algorithm, Robot Guided by Vision, Trajectory Prediction using LSTM, and Reinforcement Learning, considering factors such as response speed, accuracy, repeatability, cost, and overall performance.

Table 1 Evaluation of Robotic Control Algorithms

Plan	Characteristics [1-10]				
	Response speed Weight 0.35	Accuracy Weight 0.35	Repeatability Weight 0.2	Cost Weight 0.1	Overall
Traditional Algorithm	4	5	4	8	4.85

Robot Guild By vision	6	7	6	8	6.65
Trajectory prediction(LSTM)	7	7	7	6	6.95
Reinforcement Learning	9	9	9	4	8.15

From the table above, we can find that the Traditional Algorithm, with its reliance on analytical models and complex mathematical equations, tends to lag in response speed and struggles with adaptability in dynamic or uncertain environments, although it offers high accuracy and low operational costs once established.

The Vision-Guided Robot performs better in terms of response speed and accuracy due to its use of Kalman filtering for rapid state estimation, but its repeatability may suffer in complex, changing conditions despite relatively low operational costs. Trajectory Prediction with LSTM excels in handling complex time series data, providing precise predictions and quick processing after initial training, but it requires significant data and computational resources, impacting initial cost and scalability. Reinforcement Learning showcases superior performance in response speed, accuracy, and repeatability, making it highly suitable for real-time applications. It can self-optimize through interaction with the environment but comes with high initial costs due to the need for substantial computational power and training.

In conclusion, while each method has its own merits, Reinforcement Learning emerges as the most adaptable and capable, promising significant long-term benefits for dynamic and real-time applications, provided the initial cost and resource demands can be accommodated.

5.4 Different of ending effector

Absolutely, the end effector plays a pivotal role in the robotic arm's ability to capture space debris. Its design is critical because it must be versatile enough to handle a variety of shapes and sizes, robust enough to withstand the rigors of the space environment, and sensitive enough to apply the correct amount of force to grasp objects securely without causing damage.

5.4.1 Net End-effector Prototype

The Net End-effector Prototype employs a deployable net designed to capture space debris by enveloping it, which can then be drawn back towards the robotic arm for containment and disposal. This method stands out for its adaptability to the irregular shapes and sizes of debris found in orbit. It simplifies the capture process by reducing the precision needed for direct contact and grasp, making it particularly suitable for objects that are tumbling or have an undefined structure.

The advantages of such a system include its versatility and safety. The flexible nature of the net allows for the capture of objects that might otherwise be difficult or dangerous to handle with a rigid end-effector. It also reduces the potential for generating further debris, as there is less risk of accidentally breaking the object during capture. Additionally, the net can be designed to fold compactly when not in use, saving valuable space on the robotic arm.

However, the design does have its drawbacks. One of the primary concerns is the potential for the net to become entangled with the object or the arm itself, which could complicate capture and retrieval. The deployment mechanism must be highly reliable; any failure could result in the loss of the debris and the net, adding to the problem. There are also challenges related to controlling the movement of the net in a zero-gravity environment and ensuring that once the debris is captured, it can be securely retained during the transport back to the robotic arm.

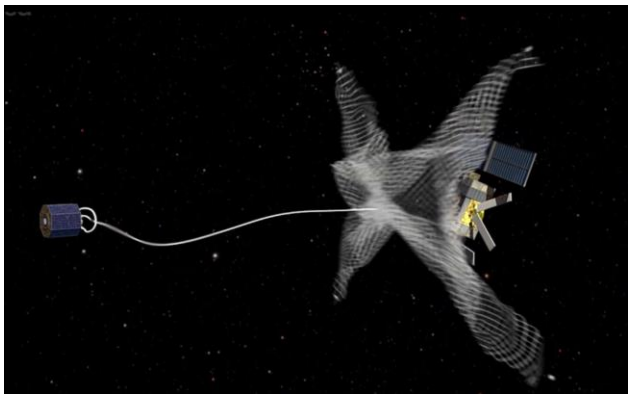


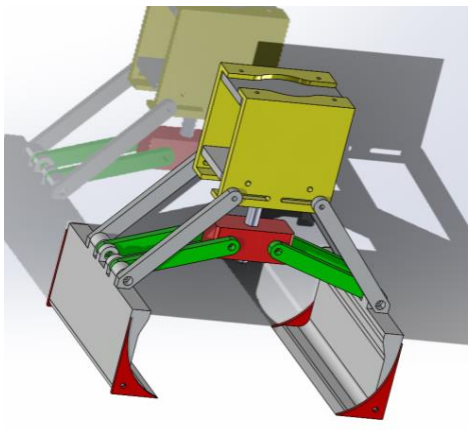
Figure 10 ESA's active debris removal mission: e. Deorbit.

5.4.2 Design of the edge of the gripper

The deployment of a gripper-based end effector for space debris cleanup missions involves using a mechanical hand or claw designed to grasp and secure debris directly. This approach typically relies on precision movement and control to manipulate the robotic arm and open and close the gripper around the targeted debris. Such systems are often designed to emulate the dexterity of a human hand and are engineered to exert the precise amount of pressure needed to maintain a secure grip without damaging the debris.

The primary advantage of using a gripper end effector is the high level of control and precision it offers. Grippers can apply a specific force to an object, enabling the handling of both sturdy and delicate items. This makes the method particularly effective for controlled operations where the size, shape, and orientation of the space debris are known and consistent. Additionally, a well-designed gripper can be highly reliable with low risks of malfunction, especially if redundancy is built into the control systems.

On the downside, gripper-based systems have limitations in terms of adaptability. They require accurate information about the debris' shape and trajectory for successful capture, which is not always possible with irregular or spinning objects. Furthermore, the need for precise alignment and approach increases mission complexity and can be challenging in the dynamic space environment. There is also a higher risk of generating additional debris if the gripper inadvertently crushes or breaks the object during the capture process. Moreover, the physical constraints of a gripper may limit the size and type of debris it can handle.



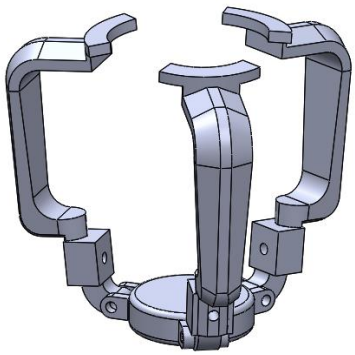


Figure 11 grippe prototype.

We've designed this gripper with a versatile and robust claw mechanism that's tailored for securing a wide array of space debris. Its articulate claws are engineered to conform to irregular shapes, ensuring a secure capture without applying excessive force, which is crucial for the delicate task of debris removal in space. The construction is sturdy, reflecting our commitment to durability and reliability in the unforgiving environment of space. This gripper is a testament to our team's innovative approach to tackling the complexities of space debris mitigation.

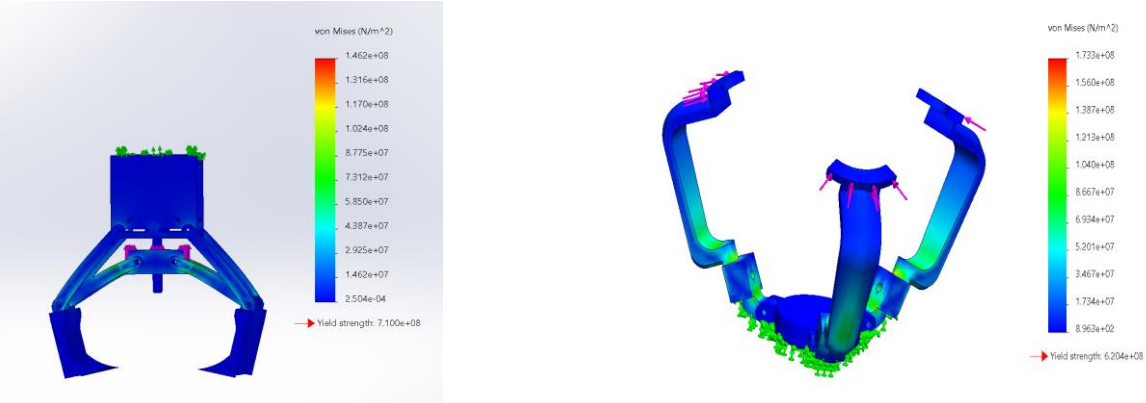


Figure 12 Finite element analysis of grippers.

Finite element analysis was performed on both different grippers. The analysis results show that our design is up to the task.

6 Proposed Design method

6.1 Principle of Reinforcement learning

Reinforcement learning is an important branch of machine learning algorithms. The important idea of using reinforcement learning to solve problems is to give specific rewards or punishment measures to individuals. Every time an individual performs an action, he will receive the corresponding reward or punishment, and then let the individual himself to find strategies for executing actions to maximize the fatigue gained during the interaction between oneself and the environment.

reward or minimum punishment. Such an interactive process is the process of individuals searching for optimal strategies. In the above reinforcement learning process, the person who performs the action and receives rewards and punishments is called the agent, and the other party that rewards the agent after making the action is called the environment. Every time the agent makes an action, rewards will be received, and punishments (Reward) given by the environment. At the same time, the environment state (State) will also change. The learning process of an agent is to aim at maximum reward (minimum punishment) and find a one-to-one correspondence between its own actions and changes in environmental state.

6.1.1 Markov decision process，MDP

In reinforcement learning, the agent performs an action, receives rewards and punishments, and changes the environmental state. When the agent continuously takes actions, the environmental state will continuously change. Such a state change is a Markov process, and the continuous state is a state sequence, which is called a Markov chain. When the last state of the sequence is the termination state, it is an event (episode). Similarly, the state is also given a value, which is called harvest G_t. It is defined as the sum of all rewards from state S_t, starting sampling until the end state. The mathematical expression of state S_t harvest is as follows:

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Among them, γ is the attenuation coefficient with a value range between $[0,1]$.

The value of the state is the expectation of state gain throughout the Markov process, which is defined as:

$$v(s) = \mathbb{E}[G_t | S_t = S]$$

Value can accurately reflect the importance of a certain state. If there is a function that can obtain the value corresponding to a given state, then the function is called a value function. Expanding the value gives:

$$\begin{aligned} v(s) &= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t = S] \\ &= \mathbb{E}[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \cdots) | S_t = S] \\ &= \mathbb{E}[R_{t+1} + \gamma G_t | S_t = S] \\ &= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = S] \end{aligned}$$

And finally, we have:

$$v(s) = R_s + \gamma \sum_{s' \in S} P_{ss'} v(s')$$

Which $P_{ss'}$ is the probability of transition between states

The above equation is called the Bellman equation. This formula illustrates that the value of a state is jointly composed of the reward of the state and the sum of subsequent state values according to the probability distribution and a certain attenuation ratio. Once the agent knows the value of each state, it can transition toward the state with the highest value by comparing the values of subsequent states. Then consider choosing an action from all actions that can be performed in a state that leads to the state of greatest value. Such an action selection behavior is a Markov decision process, which is defined as:

$$MDP = [S, A, P, R, \gamma]$$

In the formula, S is a set of all states in an environment, A is a set of actions that an agent can choose, P is a matrix composed of the probability of transition between any two states, R is the reward function, and γ is the attenuation factor. In the Markov decision-making process, an individual can select an action from all the actions that can be selected and execute it based on the knowledge of the state of the environment, thereby changing the state of the environment.

The basis for defining an agent's choice of an action from selectable actions in a certain state is called a policy. Represented by the letter π. The purpose of the reinforcement learning algorithm is to find a strategy that can maximize the cumulative reward obtained during the interaction between itself and the environment in the Markov decision-making process. This strategy becomes the optimal strategy. Combined with the formula, the optimal strategy optimization goal is:

$$\pi^* = argmaxQ(s, a) = argmax\mathbb{E}[\sum_t R(s_t, a_t)]$$

Various algorithms of reinforcement learning are formed based on the above theory. In general, the ultimate optimization goal of the algorithm is to find a strategy to select actions during the state transition process so that the agent can obtain the maximum cumulative reward after going through a complete sequence.

6.1.2 SAC (Soft actor and critic)

The development of reinforcement learning algorithms so far has mostly focused on optimizing these two points, how to improve sample utilization efficiency and avoid suboptimal strategies during the training process. The Soft Actor-Critic algorithm is a reinforcement learning algorithm based on the Actor-Critic (AC) framework developed by BAIR and Google Brain in 2018. It is an off-policy actor-critic algorithm based on the maximum entropy model framework. Among them, off-policy means that the state transition samples generated during the training process have nothing to do with the policy. This feature can improve the sample utilization efficiency during the training process and increase the training speed. Second, the AC framework is adopted, that is, two neural networks are used for policy generation and policy evaluation respectively. At the same time, the AC framework makes the algorithm a continuous space in the selection of actions and states, providing a wider range of choices for actions and states. Finally, and the most important point of the SAC algorithm is that when optimizing the strategy, we must not only maximize the cumulative reward, but also maximize the entropy of action selection as much as possible. The use of entropy allows the exploration and utilization of samples to be dynamically adjusted. At the same time, it can also help prevent the policy from prematurely converging to form a local optimum.

The SAC network model contains two Q networks (used to fit the value function, represented by θ_1, θ_2) and a policy network (used to select actions, represented by $\emptyset$). First, state transition pairs are randomly generated through the environment and stored in the experience pool D. In D, the training process then randomly selects a fixed number of state transition pairs to train the network to iterate the strategy with the goal of maximizing the cumulative reward. According to the formula, the objective function of the Q network can be obtained as:

$$J_Q(\theta) = \mathbb{E}_{(s_t, a_t, s_{t+1})}[\frac{1}{2} Q_\theta(s_t, a_t) - \left(R(s_t, a_t) + \gamma V_{\bar{\theta}}(s_{t+1})\right)^2]$$

The policy network is updated by minimizing the KL divergence:

$$J_\pi(\emptyset) = D_{KL}(\pi_\emptyset(\cdot | s_t) || \exp\left(\frac{1}{\alpha} Q_\theta(s_t, \cdot)\right) - logZ(s_t))$$

To sum up, the Soft Actor-Critic algorithm flow is obtained. The pseudo code is as follows:

Table 2 Soft Actor-Critic algorithm flow

SAC	
1	Input: $\emptyset,\ \theta_1,\ \theta_2$
2	Initialize neural network weights: $\theta_1 \leftarrow \theta_1,\ \theta_2 \leftarrow \theta_2$
3	Initialize an empty experience replay pool: $D \leftarrow \emptyset$
4	for each iteration do
5	for each environment step do
6	Action sampling in policy networks: $a_t \sim \pi_\emptyset(a_t \mid s_t)$
7	Sample state transition sequences from the environment: $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$
8	Store the sampled state transition sequence into the experience replay pool: $D \leftarrow D \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$
9	end for
10	for each gradient step do
11	Update Q network parameters: $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)\ \ i \in \{1,2\}$
12	Update policy network weight: $\emptyset_i \leftarrow \emptyset_i - \lambda_\pi \hat{\nabla}_{\emptyset_i} J_\pi(\emptyset_i)$
13	Update entropy weight parameters: $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$
14	Update target network: $\overline{\theta_i} \leftarrow \tau \overline{\theta_i} \theta_i + (1-\tau)\theta_i$
15	end for
16	end for
17	Output: $\emptyset,\ \theta_1,\ \theta_2$

6.1.3 Reward function design.

The reward function in reinforcement learning is a crucial component that provides the algorithm with feedback on how well it behaves. The function of the reward function is to score the actions taken in a given state and guide the reinforcement learning algorithm to make decisions. It usually has two categories: positive rewards (or rewards), which are used to encourage certain behaviors; negative rewards (or punishments), which are used to prevent or reduce certain behaviors. The design of the reward function is critical to ensuring that the algorithm learns to perform the task efficiently, as it defines the goals and incentives during the learning process.

The importance of the reward function is that it directly affects the performance of the learning algorithm and the quality of the learned policy. A good reward function can guide the algorithm to quickly find effective solutions in complex environments. If the reward function is not set properly, it may lead to low learning efficiency, or even no useful strategies can be learned at all.

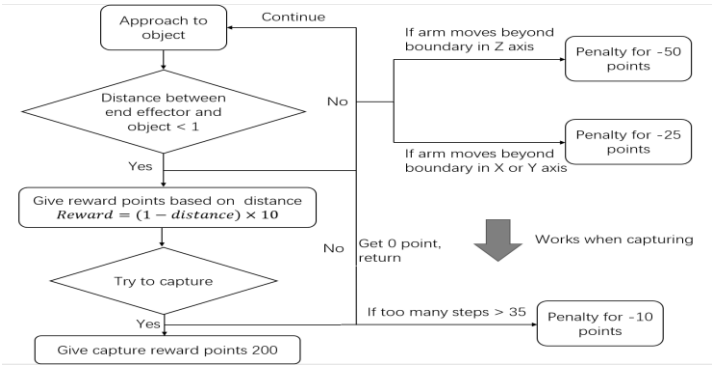


Figure 13 Reward Function

In our reinforcement learning project, we designed a reward function to guide our robotic arm to perform object grasping in a simulated space environment. When the robot arm is close to the target object and the distance to the object is less than 1, our system will calculate the reward score based on the distance and give feedback according to the formula "Reward = (1 - Distance) x 10". If the object is successfully captured, the system will award 200 bonus points. We also introduced a

penalty mechanism to prevent the robot arm from going beyond the predetermined range: if it goes beyond the bounds of the Z-axis, we deduct 50 points; if it goes beyond the bounds of the X-axis or Y-axis, we deduct 25 points. To encourage efficiency, if the robot arm takes more than 35 steps during the grasping process, we will deduct an additional 10 points. Through such a reward and punishment structure, we train the robot arm to complete grasping tasks within spatial constraints in a faster and more accurate manner.

6.2 The RL virtual physical environment

PyBullet is an open-source physics engine developed for robotics simulation, games, and machine learning research. It's particularly known for its efficiency in simulating complex robot locomotion and dexterous manipulation. Using PyBullet as the virtual physical environment for a reinforcement learning (RL) project offers several advantages and specific features worth discussing:

PyBullet provides accurate physics simulation with a focus on real-time applications, making it highly suitable for projects that require the dynamic interaction of objects in a simulated environment. This includes the handling of collisions, friction, and realistic joint dynamics, which are critical for the accurate simulation of a robotic arm in space. The engine supports advanced features such as soft body physics and fluid dynamics, which can be crucial for simulating the intricate behaviors needed in complex scenarios like space debris manipulation. One of PyBullet's key strengths is its seamless integration with machine learning libraries and frameworks, particularly those used in reinforcement learning like TensorFlow and PyTorch. This compatibility allows for the direct application of RL algorithms to control simulated robotic systems, facilitating the development and testing of models like the SAC (Soft Actor-Critic) within a realistically simulated environment. We can train their models using data generated from the simulations, iterate quickly, and observe the effects of their algorithms on robot behavior in real-time.

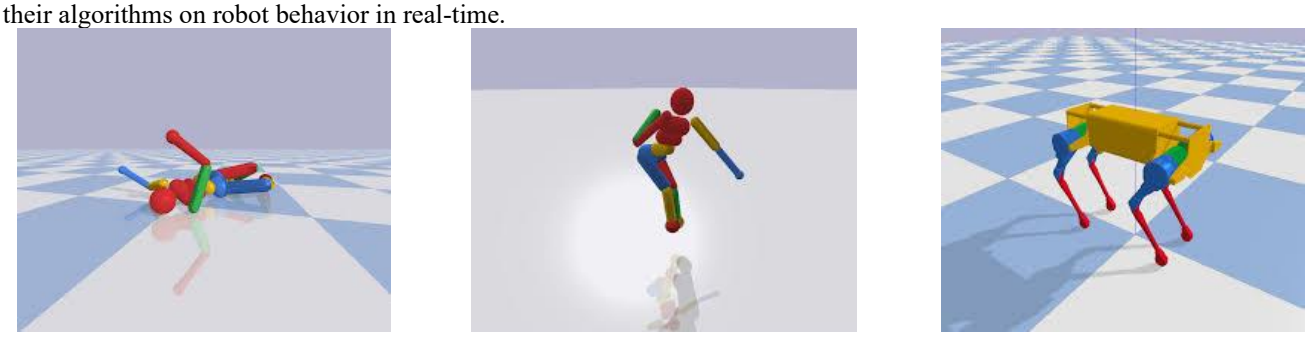


Figure 14 Simulations in Pybullet

PyBullet excels in providing real-time visualization tools that help us observe the simulation without the need for additional software. This is particularly useful in debugging and fine-tuning the robot's actions based on the visual feedback directly from the simulation. The debugging tools allow users to draw debug information or change simulation parameters on the fly, which is invaluable for developing and testing complex robotic systems such as those involved in your project.

Dealing with space debris, PyBullet provides an excellent platform to simulate the zero-gravity conditions that are typical in space. The engine can simulate reduced or zero gravity environments, allowing the RL algorithms to train under conditions like those they will face in actual space missions. This is crucial for ensuring the algorithms can cope with the unique challenges of operating in space, such as handling objects that move in unpredictable ways due to the lack of atmospheric drag and gravity.

6.3 The transformation of coordinate between camera and real world

In the working space of the robot arm, the position of the target grasping end point is represented by coordinates based on a coordinate system. Then in actual robot grasping experiments, the position of the object is often detected by the camera first. At this time, the target detection algorithm outputs the position coordinates are based on the camera coordinate system, but the coordinate system that the reinforcement learning control system designed previously refers to the mobile grasping target position coordinates is often the robot base coordinate system. In this case, the target needs to be based on the camera

coordinate system. The coordinates are converted into coordinates in the robot base coordinate system. This process is coordinate conversion.

First, assume that there is a coordinate system $\{A\}$ in space as the reference coordinate system, and then establish a coordinate system {B} for rigid body B in this space. Under this assumption, the essence of the pose description is the spatial coordinate system of rigid body B. The relationship between the transformation relative to the datum coordinate system {4}. For the selection of the coordinate origin of the coordinate system $\{B\}$, the center of gravity, geometric center, etc. of the rigid body is usually used as the origin of the coordinate system.

Use the position vector "P." to describe the position of the origin of the motion coordinate system $\{B\}$ in the datum reference coordinate system $\{A\}$ with respect to its motion coordinate axis. The rotation matrix ${}^{A}_{B}R$ can be used to describe it. For the rotation matrix, it is defined as: the projection of the x-axis of the coordinate system $\{B\}$ on the x, y, and z-axes of the coordinate system $\{A\}$ respectively can form the first column of the third-order rotation matrix. Similarly, the x, y of the coordinate system $\{B\}$, the projections of the z-axis on the x, y, and z-axes of the coordinate system $\{A\}$ respectively constitute the first, second, and third columns of the rotation matrix. From this we can get the reference coordinate system $\{B\}$ pose of rigid body B as shown in Equation below.

$$\{B\} = \{{}^{A}_{B}R \ {}^{A}P_{B}\}$$

If a certain coordinate system moves in the space of the reference coordinate system with a constant attitude, the spatial transformation in this case is called a pure translation transformation. The attitude angle under pure translation transformation will not change, that is, the rotation matrix to be discussed later is the identity matrix, so this transformation in space will only change the position of the origin of the motion coordinate system relative to the base reference coordinate system.

The position of the motion coordinate system $\{B\}$ in the datum reference coordinate system $\{A\}$ can be described by the position vector ${}^{A}P$. From this, the position of the coordinate system $\{B\}$ relative to the coordinate system $\{A\}$ can be obtained by the vector addition formula The vector, as shown in the equation, is called the coordinate translation equation:

$${}^{A}P = {}^{B}P + {}^{A}P_{B}$$

When describing the expression form of coordinates, the description of the pose relationship of the general motion coordinate system $\{B\}$ in the reference coordinate system $\{A\}$ can be compounded by the left multiplication of the pose matrix under the motion coordinate system $\{B\}$ (including translation and rotation) The transformation matrix is obtained. In the pure translation transformation, the attitude of the coordinate system $\{B\}$ does not change, so the position transformation matrix T can be expressed as the following equation:

$$T(x,y,z) = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where P_x, P_y and P_z are the three components of the pure translation vector d relative to the x, y, z axes of the reference coordinate system. The rotation matrix represented by the first 3 columns of the matrix remains unchanged because the projections are all 1 (that is, the identity matrix), and the last column represents the position of the origin of coordinate system $\{B\}$ relative to coordinate system $\{A\}$. This homogeneous matrix is composed of a 3×3 rotation matrix and a 3×1 position matrix, plus the row vector of $[0\ 0\ 0\ 1]$ in the fourth row.

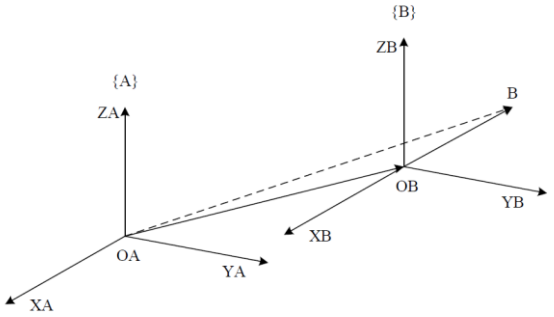


Figure 15 Translation transformation.

For the derivation of pure rotation transformation, we can first understand it by simply rotating around the x, y, and z axes. First, assume that the motion coordinate system $\{B\}$ is located at the origin of the datum reference coordinate system

$\{A\}$, and the x, y, z axes of the two coincide with each other. A deeper understanding of this simple example can be generalized to other cases of rotation or combinations of translation and rotation.

Under the above premise, let the coordinate system $\{B\}$ rotate around the x-axis of the datum reference coordinate system $\{A\}$ by a certain angle 0, and assume that there is a point B on the rotating coordinate system {B}. This point is relative to the motion coordinate system The coordinates of $\{B\}$ are P_x, P_y and P_z and the coordinates relative to the base reference frame are P_a, P_b and P_c. When the motion coordinate system {B} rotates at a certain angle around any coordinate axis on it, point B on the motion coordinate system $\{B\}$ is transformed in the same motion manner.

Before the rotation, the x, y, and z axis coordinate vectors of point B in the motion coordinate system and the base reference coordinate system are coincident. After the rotation, the position and attitude of point B remain unchanged in the rotating coordinate system $\{B\}$, but the values of P_a, P_b and P_c in the base reference coordinate system $\{A\}$ have changed. Therefore, it is necessary to find the new coordinates of point B relative to the fixed reference coordinate system $\{A\}$ after the motion coordinate system $\{B\}$ is rotated.

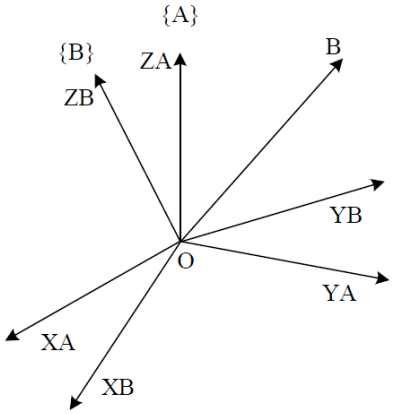


Figure 16 Rotation transformation.

At this time, the coordinates of point B relative to the reference coordinate system $\{A\}$ after rotation can be expressed in the form of a matrix.

$$\begin{bmatrix} P_a \\ P_b \\ P_c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}$$

If the motion coordinate system $\{A\}$ is translated multiple times along the x, y, z axes and rotated around these three axes, then this transformation is called a combination of the above two transformations and is called a composite transformation. Composite transformation needs to be carried out in sequence, as shown in the figure.

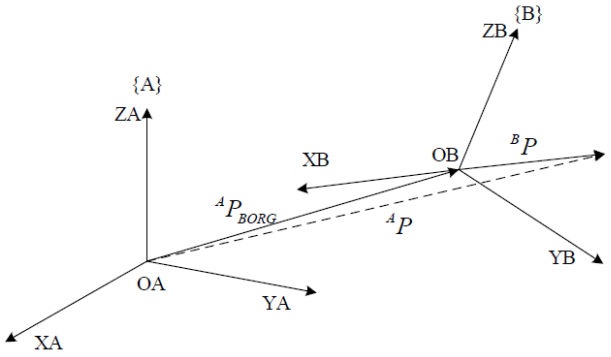


Figure 17 Compound transformation with translation and rotation.

Compound transformation is a combination of translation transformation and rotation transformation. The transformation matrix is as shown in the formula:

$$^{A}P = {}^{A}_{B}T\,{}^{B}P = \begin{bmatrix} {}^{A}_{B}R & P_{BORG} \\ 0\ \ 0\ \ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix}$$

P_{BORG}, ${}^{A}_{B}R$ are the translation matrix and rotation matrix between coordinate system $\{B\}$ and coordinate system $\{A\}$ respectively.

7 Results of the experiment.

Grabbing Fixed Objects:

In the initial phase of the project, the SAC (Soft Actor-Critic) model was tasked with controlling the robotic arm to grasp fixed objects, a foundational step essential for laying the groundwork for more complex tasks. This stage primarily focused on perfecting the basic control and manipulation capabilities of the robotic arm within a controlled environment, where variables such as object movement and changing trajectories were absent, allowing for a high degree of success. By concentrating on static targets, the model could thoroughly learn and refine the essential mechanics of object recognition and gripping, such as calculating grip strength, aligning the arm's approach, and ensuring stable contact. This mastery of basic skills was critical, ensuring that the SAC model developed a robust understanding of spatial relationships and the physical properties of objects it interacted with, thereby establishing a solid foundation for subsequent stages involving more dynamic and unpredictable conditions. This step not only demonstrated the model's potential in a simplified setting but also set the stage for advancing to more challenging tasks, progressively building towards handling the complexities of moving objects in space.

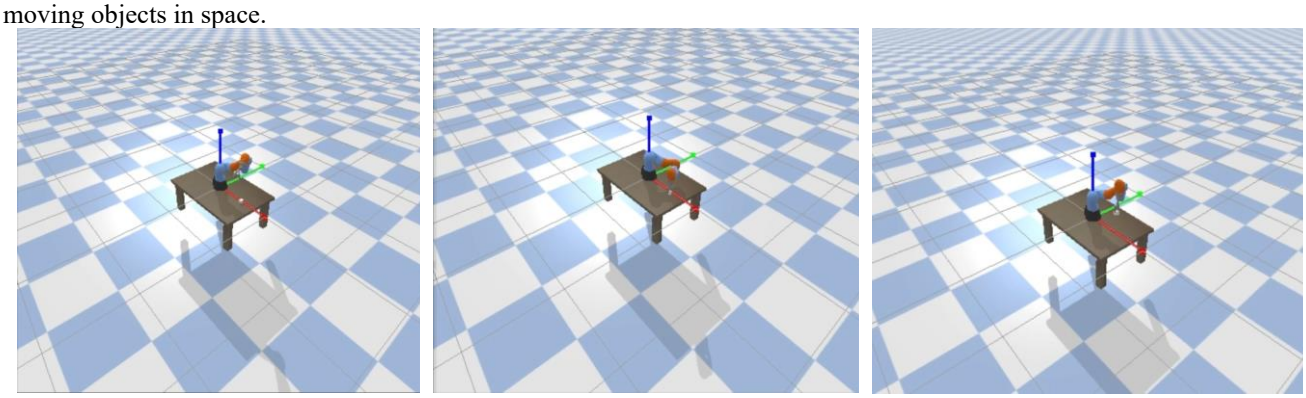


Figure 18 Catching Fixed Object in Pybullet Simulation Environment

Grabbing Moving Objects in 2D Space

The transition from manipulating fixed objects to handling moving objects within a two-dimensional framework marked a critical escalation in the project's complexity, presenting new dynamic challenges that tested the adaptability and responsiveness of the SAC model. As objects began to move, the model was required to rapidly adjust to continuously changing variables such as speed and direction, demanding accurate real-time predictions and calculations of trajectories to achieve successful grasping. This phase significantly tested the SAC model's capacity for dynamic decision-making and required enhancements to its learning algorithms to better understand and predict the motion patterns of various objects. Over time, with extensive training and iterative tuning, the model's ability to swiftly interpret and react to these dynamic elements saw considerable improvement. The success rate gradually increased as the SAC model honed its capabilities in tracking moving targets, predicting their future positions with greater precision, and timing the movements of the robotic arm to align perfectly with the trajectory of the objects, thereby refining its overall performance in a controlled yet increasingly complex setting.

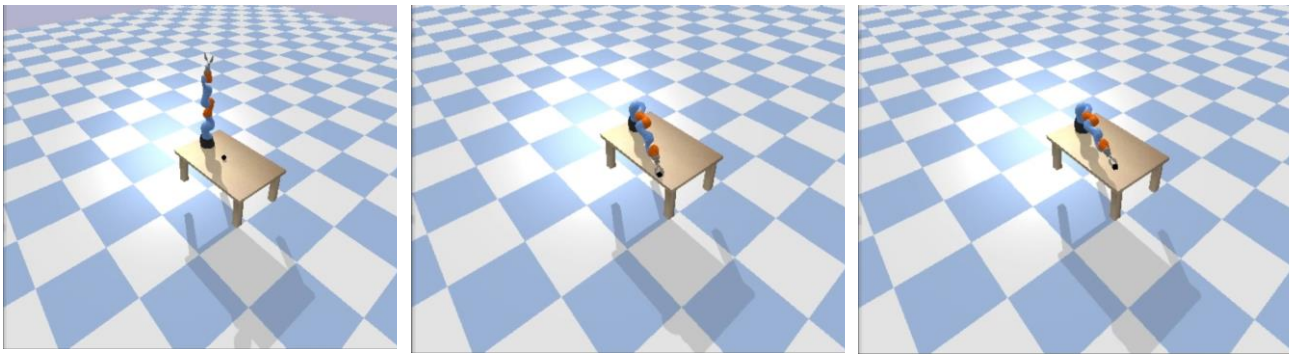


Figure 16 Catching 2D Moving Object in Pybullet Simulation Environment

Grabbing Moving Objects in 3D Space

Moving to the task of grappling with objects in three-dimensional space posed the greatest challenge yet for the SAC model, representing a significant leap in the project's complexity and a test of its technological frontier. In this advanced stage, the SAC model was required to interpret a much more complex environment, dealing with additional variables such as the depth perception of objects, their varying speeds, complex and non-linear trajectories, and their orientation in space—all of which necessitate a more sophisticated approach to prediction and movement execution. The lower success rate in this phase reflects the substantial increase in difficulty, as the model must perform intricate calculations and make real-time spatial adjustments to align the robotic arm precisely with the moving targets. This requires not only a deeper understanding of three-dimensional dynamics but also a rapid processing capability to manage continuous changes in the environment. The challenges here are immense, as each aspect of the operation—from detecting and tracking objects to executing a successful grasp—demands precision and adaptability, testing the limits of current AI technologies and learning frameworks in an unforgiving and complex arena.

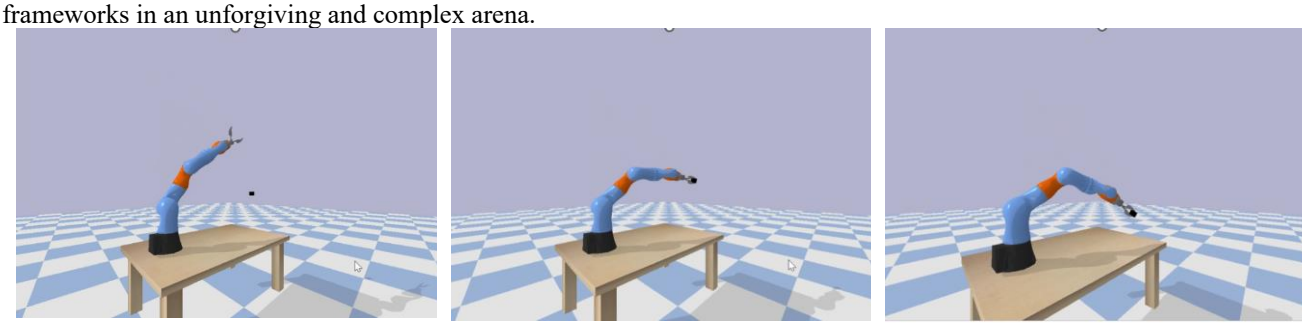


Figure 17 Catching 3D Moving Object in Pybullet Simulation Environment

Limitations:

The difficulties in advancing the SAC model's capability to grasp moving objects in 3D space are multifaceted, with one significant hurdle being sensor limitations. The accuracy with which a robotic system can perceive and interact with its environment is heavily dependent on the quality of its sensors, particularly in terms of depth perception and spatial resolution. In the context of space, where depth perception is critical due to the absence of gravitational references, any inaccuracy in sensor readings can lead to miscalculations in object positioning. Enhanced sensor technologies with higher resolution and better depth accuracy are crucial for improving the model's ability to accurately localize objects in the vast and complex space environment. This improvement would directly contribute to increasing the precision of the robotic arm's movements and its success rate in object retrieval tasks.

Another critical area requiring attention is the enhancement of the SAC algorithm itself. While the SAC model is known for its effectiveness in reinforcement learning tasks requiring a balance of exploration and exploitation, its application in a three-dimensional, dynamic environment presents unique challenges. The variability and unpredictability of object movements in space require the algorithm to make highly precise predictions and execute complex maneuvers in real-time. This necessitates further tuning of the algorithm to enhance its predictive accuracy and responsiveness. Enhancements might include refining the model's neural architecture, improving its ability to learn from fewer examples, and adjusting its reward functions to prioritize crucial factors like speed and trajectory adjustments. These improvements are essential for ensuring that the SAC model can effectively manage the increased complexity and dynamics of a 3D space environment.

The quality and variety of training data, along with the fidelity of the simulation environment used for training the SAC model, play a pivotal role in its real-world performance. Training data must encompass a wide range of scenarios that the robotic arm could potentially encounter in space, including various object sizes, shapes, and motion patterns. The simulation environment needs to mimic the space conditions as closely as possible to ensure that the model learns and adapts to the realities of space operations. Limitations in these areas can lead to a model that performs well in simulated or controlled environments but fails to translate these results into real-world effectiveness. Therefore, expanding the training dataset with more diverse examples and enhancing the simulation's realism are critical steps towards preparing the SAC model for successful deployment in space debris management tasks.

8 Conclusion

In this project, we took a leap forward in robotics, integrating cutting-edge machine learning to develop a robotic arm capable of autonomously identifying and capturing objects in a simulated outer space environment. By fusing the Yolo object detection system with the Soft Actor-Critic algorithm, we push the boundaries of practical applications of artificial intelligence, especially in space-like conditions.

We have made substantial progress in enhancing the algorithm by customizing the reward function, which significantly improves the performance of the SAC model. This also improves the system's ability to adapt and perform in different scenarios. The robustness of our approach was tested in demanding simulations running on the Pybullet physics engine, where our robotic arm was shown to be able to grasp objects in zero gravity, mimicking the challenges of space. Our model continuously evolves, learns, and improves through iterative training on a variety of objects, highlighting the adaptability of our algorithm.

The insights we gathered throughout our journey were invaluable. They emphasize the importance of iterative testing to create systems that operate reliably under large uncertainty spaces. Achieving the right balance between system complexity and operational performance is critical to optimizing cost efficiency and functionality. Our team's innovative and collaborative spirit, bringing together expertise in engineering and machine learning, is key to our ability to solve problems quickly and innovatively. The project not only marks a breakthrough in solving the critical problem of space debris, but also pioneers the integration of advanced artificial intelligence-driven systems into future space endeavors.

9 Future work

Future work on the robotic arm project, particularly aimed at tackling space debris, is set to expand, and enhance the capabilities and effectiveness of the system. A key area of focus will be the optimization of the Soft Actor-Critic (SAC) network. The SAC algorithm, known for its efficiency and robustness in decision-making tasks, will undergo further refinements to boost its performance specifically in the more challenging scenarios of space debris management. This involves enhancing the algorithm's ability to identify and grasp a variety of debris types under a range of conditions that mimic the unpredictable dynamics of space accurately and reliably. Such improvements could significantly increase the success rate of the robotic arm in performing critical cleanup tasks, thereby contributing to safer orbital environments.

Another significant step forward involves transitioning from simulations to real-world testing. The project plans to conduct tests aboard the International Space Station (ISS), marking a pivotal shift from theoretical models to practical applications. This phase will allow the team to evaluate the robotic arm's functionality and performance in an actual zero-gravity environment, providing invaluable data on its operational capabilities and limitations. Testing in such a unique and challenging environment will also help refine the system's design and operation, ensuring it can withstand the rigors of space and effectively perform its intended tasks. This real-world testing is crucial for moving towards operational deployment, aiming to mitigate the growing concern of space debris.

In addition to refining and testing the current system, there is an ambitious plan to scale the technology for broader applications. The project aims to develop and deploy multiple robotic systems capable of managing different types of space debris and adapting to various operational scenarios. This expansion is geared towards creating a more comprehensive debris management system that can handle the increasing complexity and quantity of debris in orbit. By scaling up the technology, the initiative could provide a robust solution to ensure the long-term sustainability of space operations,

protecting valuable satellites and spacecraft from potential collisions and contributing to the overall safety and cleanliness of our near-Earth space environment.

References

[1]. Park T H, Lee B H. An approach to robot motion analysis and planning for conveyor tracking[J]. IEEE transactions on systems, man, and cybernetics, 1992, 22(2): 378-384.

[2]. Kimura H, Mukai N, Slotine J J E. Adaptive visual tracking and Gaussian network algorithms for robotic catching [J]. Advances in robust and nonlinear control systems, 1992: 67-74

[3]. Chen Y, Watson LT. Optimal trajectory planning for a space robot docking with a moving target via homotropy algorithms [J]. Journal of robotic systems, 1995, 12(8): 531-540

[4]. 王树国、严艳军，赵晓东。基于轨迹规划的自由漂浮空间机器人抓取运动物体的研究[J].宇航学报，2002,23(3):48-51

[5]. Aghili F. A prediction and motion-planning scheme for visually guided robotic capturing of free-floating tumbling objects with uncertain dynamics [J]. IEEE Transactions on Robotics, 2012, 28(3): 634-649.

[6]. Riley M, Atkeson C G. Robot catching: Towards engaging human-humanoid interaction[J]. Autonomous Robots, 2002, 12(1): 119-128.

[7].] Senoo T, Narniki A, Ishikawa M. Ball control in high-speed batting motion using hybrid trajectory generator [C]/Robotics and Automation. 2006. 1CRA 2006. Proceedings 2006EEE International Conference on. IEEE. 2006: 1762-1767.

[8]. Houshangi N. Control of a robotic manipulator to grasp a moving target using vision[C]//Robotics and Automation, 1990. Proceedings. 1990 IEEE International Conference on. IEEE, 1990: 604-609.

[9]. Allen P K, Timcenko A, Yoshimi B, et al. Automated tracking and grasping of a moving object with a robotic hand-eye system [J]. IEEE Transactions on Robotics and Autonation.1993, 9(2): 152-165.

[10]. Bing W, Xiang L. A simulation research on 3d visual serving robot tracking and grasping a moving object[C]//2008 15th International Conference on Mechatronics and Machine Vision in Practice. IEEE, 2008: 362-367.

[11]. Asada M, Tanaka T, Hosoda K. Adaptive binocular visual serving for independently moving target tracking [C]//Robotics and Automation, 2000. Proceedings. ICRA.IEEE International Conference on. IEEE, 2000, 3: 2076-2081.

[12]. Su J, Xi Y. Path planning for robotic hand/eye system to intercept moving object[C]//Decision and Control, 1999. Proceedings of the 38th IEEE Conference on IEEE,1999, 3: 2963-2968

[13]. Mehrandezh M, Sela M N. Fenton R G, et al. Proportional navigation guidance for robotic interception of moving objects [J]. Journal of Robotic Systems, 2000, 17(6): 321-340

[14]. Keshmiri M, Xie W F. Catching moving objects using a navigation guidance technique in a robotic visual serving system [C]//American Control Conference (ACC), 2013. IEEE,2013: 6302-6307.

[15]. Agah F, Mehrandezh M, Fenton R G, et al. On-line robotic interception planning using a rendezvous guidance technique [J]. Journal of Intelligent and Robotic Systems. 2004, 40(1): 23-44.

[16]. Levine S, Pastor P, Krizhevsky A, et al. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection[J]. The International Journal of Robotics Research, 2018, 37(4-5): 421-436.

[17]. Zeng A, Song S, Welker S, et al. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning[C]//2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018: 4238-4245

[18]. Berscheid L, Meißner P, Kröger T. Robot learning of shifting objects for grasping in cluttered environments[J]. arXiv preprint arXiv:1907.11035, 2019.

[19]. Quillen D, Jang E, Nachum O, et al. Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods[C]//2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018: 6284-6291.

[20]. Kalashnikov D, Irpan A, Pastor P, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation[J]. arXiv preprint arXiv:1806.10293, 2018.

[21]. Zhu H, Gupta A, Rajeswaran A, et al. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost[C]//2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019: 3651-3657

[22]. Haarnoja T, Zhou A, Hartikainen K, et al. soft actor-critic algorithms and applications[J]. arXiv preprint arXiv:1812.05905, 2018.

[23]. Yahya A, Li A, Kalakrishnan M, et al. Collective robot reinforcement learning with distributed asynchronous guided policy search[C]//2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017: 79-86.

[24]. Akkaya I, Andrychowicz M, Chociej M, et al. Solving Rubik's cube with a robot hand[J]. arXiv preprint arXiv:1910.07113, 2019.
[25]. Jeong R, Aytar Y, Khosid D, et al. Self-supervised sim-to-real adaptation for visual robotic manipulation[C]//2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020: 2718-2724.
[26]. Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779-788.
[27]. International Organization for Standardization. (2014). ISO 13482:2014 Robots and robotic devices — Safety requirements for personal care robots.
[28]. Institute of Electrical and Electronics Engineers. (2015). IEEE 1872-2015: Standard for Ontological Standard for Ethically Driven Robotics and Automation Systems.
[29]. International Organization for Standardization. (2022). ISO 14302:2022, Space systems - Electromagnetic compatibility.
[30]. International Organization for Standardization. (2006). ISO 14624-5:2006, Space systems — Safety and compatibility of materials — Part 5: Determination of reactivity of system/component materials with aerospace propellants.
[31]. Denavit J, Hartenberg R S. A kinematic notation for lower-pair mechanisms based on matrices[J]. Trans of the Asme journal of Applied Mechanics, 1955, 22: 215-221
[32]. Jahanshahi, H., & Zhu, Z. H. (2024). Review of Machine Learning in Robotic Grasping Control in Space Application. Acta Astronautica.
[33]. Jianjun Luo, Lijun Zong, Mingming Wang, Jianping Yuan, Optimal capture occasion determination and trajectory generation for space robots grasping tumbling objects, Acta Astronautica Volume 136,2017, Pages 380-386
[34]. Rouleau, G., Verma, S., Sharf, I., & Martin, É. (2005). Vision-based tracking and trajectory generation for robotic capture of objects in space. In AIAA Guidance, Navigation, and Control Conference and Exhibit (p. 6446).
[35]. Z. Shi, M. Xu, Q. Pan, B. Yan and H. Zhang, "LSTM-based Flight Trajectory Prediction," 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 2018, pp. 1-8
[36]. Andrea F, Elisa T, Nicola C, et al. Robotic arm control and task training through deep reinforcement learning [EB/OL]. (2020-05-06) [2020-09-01]