Autonomous Space Debris Collection using Deep Reinforcement Learning

James May
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, NY
jlm7092@rit.edu

ABSTRACT

This paper proposes a method of autonomously collecting debris from orbit around our planet, for deorbiting. Deep reinforcement learning is used to teach an arm robot platform to recognize optimal grasp locations on a variety of debris objects similar to what would be found in orbit. Proximal Policy Optimization is used to generate a policy capable of determining proper joint and gripper poses to grasping a floating object. Simulated dead CubeSat satellites are included in the training process. A policy is learned to grasp the object that achieves a 80% accuracy on colliding with 3U CubeSat objects.

I. PROBLEM STATEMENT

This paper proposes an approach to autonomously collect space debris.

As we begin to become a truly space faring species, we must acknowledge the errors of past space projects, and work to correct them. One such error has been the accumulation of space debris around our planet, which presents a multitude of risks to any space vehicle. Being struck by a piece of space debris would not only cripple any space vehicle available today, but most likely also produce more space debris through the impact, leading to a potential exponential growth in risk which would render us unable to leave the planet. To resolve this, we need to create systems capable of cleaning up the space debris present around our planet, and ensure a clean future for space travel.

II. LITERATURE SURVEY

A. Space Debris Removal

The problem of space debris removal has been increasingly discussed in recent years. The Kessler Syndrome, proposed by Donald Kessler in 1978 [1], is a prediction that with enough space debris in orbit, the system becomes unstable in that one collision could start a cascade that would riddle the atmosphere with an exponentially increasing number of high velocity debris objects. Especially in light of recent Soyuz capsule damages due to space debris impact, many researchers are looking towards this problem and potential solutions.

[2] provides an excellent summary of the problem from not only the perspective of debris accumulation, but also the increasing number of government and private bodies who wish to put satellites & other bodies into orbit, and how this can be done so sustainably.

Tether space robots have been proposed in a number of papers as a potential solution to this problem. [3], [4], [5], & [6] all propose tether-based robots for space debris collection. Tether robots have an end effector attached to the primary robot via a tether. This tethered end effector would grab onto target debris & tow it to a safe deorbiting altitude. However, these proposed robotic systems are still theoretical, and do not have demonstrated reliability in space, unlike a rigid robotic arm.

Some researchers have begun to apply machine learning to different aspects of the problem. [7] provides an optimization algorithm for planning space debris removal missions, for both single object capture and multiple object capture. [8] provides a space debris dataset for researchers to conduct supervised learning on. This dataset enables supervised learning of space object detection & removal, and even includes orbital data for proper positioning of objects in a model. [9] also tackles mission planning similar to [7] but produces an algorithm by employing reinforcement learning.

B. End Effectors

The act of grasping a potentially large, adversarial object such as a satellite in a space environment is a daunting task. Some researchers have begun to look into novel end effector designs that could cope with such tasks.

[10] proposes using a set of “caging” robotic arms, which would grasp a large object at geometrically optimal points to hold it. They provide an algorithm for positioning the arms to solve this problem. This may be computationally intensive to implement in an actual mission. This has the benefit of using robotic arms which are a proven system in space.

[11] proposes a net system with multiple maneuverable units. The units would maneuver the net around the object to collect it, then direct the whole assembly to a deorbiting

Identify applicable funding agency here. If none, delete this text box.

velocity. This is a novel approach, but would need testing before viability could be determined.

Unfortunately, most of these systems are purely theoretical, and will most likely take longer to adapt into a feasible system, compared to a more standard and already space-proven system such as an electric gripper.

C. Grasp Detection

Grasp detection is a core sub-problem, since in order to deorbit debris, you must first be able to catch it. The majority of research on grasp detection focuses on parallel grippers & uses one or more of a set of well-established grasp datasets, including the Cornell & Jacquard datasets.

[12] demonstrates a reinforcement learning approach towards grasp detection & execution. It employs a double-DQN architecture & uses a multi camera setup comprised of a single RGB camera and a single RGB-D camera. It demonstrates accuracies of greater than 90% on certain objects for 4DoF grasps.

[13] also limits to 4DoF grasps, and uses a single RGB-D camera. Supervised learning is employed using common grasp datasets, like the Cornell grasp dataset. Accuracy of 98% is achieved on single objects, and an accuracy of 93% is achieved on clusters of objects.

A transformer-based approach to grasp detection can be seen in [14]. This approach uses a single RGB-D camera, and can achieve accuracies around 95% on common grasp datasets.

[15] uses a single RGB-D camera input, and generates grasp detections for cluttered & stacked environments. Its EGNet can produce accuracies of 94% & 70% in cluttered and stacked environments, respectfully.

The challenge of 6DoF pose estimation is less common, since less datasets provide 6DoF grasps, and the challenge is more difficult. [16] proposes a CNN-based 6DoF grasp detector that can achieve 93% accuracy.

[16] proposes a generative CNN-based solution which gives accuracies of 99% & 96% on the Cornell and Jacquard datasets, respectively.

[17] proposes a 4DoF grasp detection algorithm using parallel grippers with a real world accuracy of 92%.

[18] uses single RGB-D images to generate grasp poses. Their SymmetryGrasp algorithm achieves state-of-the-art accuracy on the GraspNet-1-Billion dataset.

[19] uses a Feature Pyramid Network (FPN) on single RGB-D images to produce grasp detections. It achieves a 93% accuracy & 90% accuracy on the Cornell and Jacquard datasets, respectfully.

[20] performs RGB-D fusion, which uses both RGB-D and RBG camera data to perform grasp detections, similar to [12]. It is able to achieve accuracies of 99% & 94% on the Cornell and Jacquard datasets, respectfully.

As can be seen, there exists a plethora of grasp detection methodologies and approaches, the vast majority of which have over 90% accuracy. This means there is an abundance of existing research to draw inspiration from in order to apply grasp estimation to the task of space debris collection.

III. Proposed Approach

In order to solve the problem of grasping space debris, a deep learning approach is proposed. This method would allow a robotic arm to learn optimal methods for grasping these adversarial objects and maneuvering them into a goal position. The Sawyer research robot platform was chosen for this task. The robot's parallel end effector will be used to manipulate the debris, and the two onboard cameras will allow the robot to observe the environment. The head camera is placed on the head of the robot and has a good perspective of the total environment. The wrist camera moves with the arm and provides a good close perspective of the object to be grasped. Using both cameras will allow the robot to perceive depth, and through deep reinforcement learning learn to correlate the observations with the state of the environment.

This system would be part of a greater mission including an orbiting satellite for the system to attach to, equipped with a holding mechanism for the collected debris, as well as a plan to deorbit either the debris itself, or the entire system. This project assumes that the whole system would have an RCS or equivalent system to bring the arm close enough to perform a grasp. The grasp detection will be trained on objects with varied position and rotation, to improve the robustness of the system and ensure that the exact position and rotation of the debris does not have to be matched, which will save fuel.

A. Simulation

CoppeliaSim was chosen to be the simulator for this task. CoppeliaSim provides an easy to use drag-and-drop interface, as well as the Sawyer robotic platform as a built in robot. CoppeliaSim also provides robust control over the simulated environment through its remote API. An environment including the robot can be built, and the program allows for the camera feeds of the robot to be taken, which allows realistic partial observation of the environment.

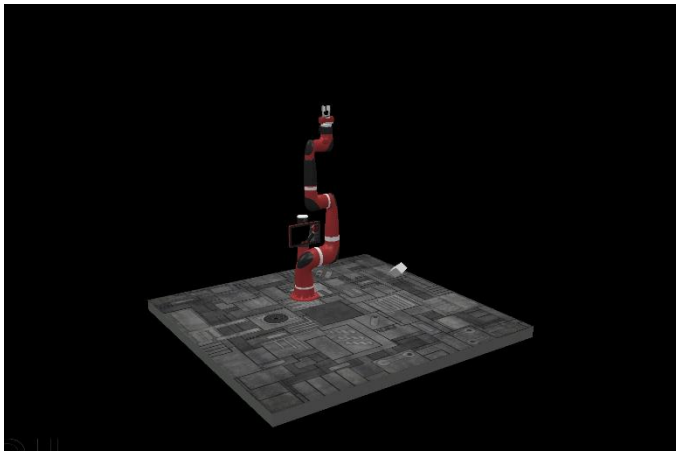

Figure 1: CoppeliaSim Environment

B. Reinforcement Learning Approach

From available RL algorithms, Proximal Policy Optimization (PPO) was chosen. PPO tends to converge well compared to other common RL algorithms, and is able to handle

continuous action spaces. The Stable Baselines3 implementation of PPO was used, in order to rely on a stable implementation of the algorithm. A Python script will interact with the CoppeliaSim simulation through the remote API provided, which will allow the full control over the environment necessary for deep reinforcement learning to occur.

The architecture of the policy is as follows. The policy was a Stable Baselines3 MultiInputPolicy, with two Nature Atari CNNs as feature extractors. They produce a latent vector of size 256 each. There is also a third input which represents the current position of the Sawyer robot, which is a vector of size 8. These 3 inputs are then fed into two separate actor and critic fully connected networks, each with 3 hidden layers with 256 neurons. Tanh activation functions were applied to all the hidden layers to inject nonlinearities. The actor network had an output size of 8 neurons, corresponding to the desired joint positions of the robot. The critic network had an output size of 1, which is the value of the state. The hyperparameters used were as follows:

n_steps=512

gamma=0.99

learning_rate=0.0003

batch_size=64

Training was conducted so that every 100 iterations a model was saved.

C. Environment

In order to perform Reinforcement Learning, a custom environment was required. This environment required 4 critical components:

1. Connection to CoppeliaSim for monitoring and controlling the simulation.
2. A reward function to evaluate the current state of the simulation.
3. An action space to define what robot parameters were under control of the learned policy.
4. An observation space to define what observations the policy could make from the simulation.

A custom environment was written as a Python class, which provides all these requirements, as well as some helper functions for debugging and controlling the simulation. This class was designed to be Gymnasium-compliant, which is both a requirement for Stable Baselines3 and helps make the environment more adaptable to future projects.

The reward environment was defined to help the robot converge to a successful grasp. At each time step, a reward of -0.01 was applied to encourage the agent to explore the environment. If Sawyer's gripper hits the CubeSat, a reward of +1 is given each time step it is touching. If the left or right gripper pad touches the CubeSat, a reward of +10 is given per time step for each pad touching. This means a total reward of 20.99 is attainable each time step, if the object is grasped. If any part of the robot collides with the floor (barring the mounted base), a reward of -10 was applied and the run was ended, to punish the agent and ensure it would not hit the table if tested on the real robot.

The action space was a vector of size 8, with each value corresponding to the joint position of one of Sawyer's joints. The first 7 values are the normalized angular positions of the joints of Sawyer, and the 8th value is the gripper position.

The observation space is comprised of 3 components. The first two are the camera feeds from Sawyer's head camera and wrist camera. The head camera is RGB with a resolution of 360x640, and the wrist camera is RGB with a resolution 480x640. The third component is a robot joint state vector of size 8, which corresponds to the setup of the action space mentioned above. These 3 observations are the sole observations the agent can use.

D. Test Setup

Once the agent proves to be successful in simulation, then a test on a real robot is to be attempted. The Sawyer robot will be used to match with the simulated robot. Objects from the test dataset will be printed out & hung with fishing line in front of the robot, and the agent will be allowed to control the robot to see if a successful grasp can be achieved. Black poster board will be placed around the robot to simulate space & not confuse the agent.

IV. Expected Results

From this project, the primary expected result is a trained model that can coordinate the Sawyer robot to grasp a 3U CubeSat object in a low gravity environment. The desired accuracy on test objects would be greater than 90% in simulation, and greater than 80% in real testing. This model would first be demonstrated in the CoppeliaSim simulation environment, and then demonstrated in real life with the setup described above.

The minimum viable result for this project should be a success rate of greater than 80% on test objects in simulation. Transitioning from simulation to reality may present problems which prove to be insurmountable in a semester of work, in which case descoping the project to simulation only may be required.

V. Results

After training the model sporadically over the course of a week, the agent was able to successfully avoid the floor and hit the object roughly 80% of the time. The agent is also able to attempt a grasp, and can often get one of the gripper pads to connect with the object, but it does not successfully grasp.

Unfortunately, the training of the simulated robot proved to be enough of a challenge that actual testing was not able to be completed. This was fortunate, however, due to a mistake made in the definition of the reward function. Since hitting the floor was set to be an end state, the policy learned that if the block was no longer reachable, whether through hitting it away or simply not seeing it with the cameras, that the best approach would be to hit the floor as quickly as possible. The minor penalty for each time step means that even though the floor is a reward of -10, it

is preferrable to take the penalty and end quickly rather than wait and take the cumulative penalty if no object is present.

VI. Conclusion

Overall this project was an excelled learning opportunity. While the agent can not successfully grasp, it shows some level of intelligence and understanding of its environment. Given enough training time, it may actually successfully learn to grasp. Regardless, this project verifies that the approach may be feasible, it just needs architecture adjustments and more consistent training time.

There are plenty of opportunities for future improvements in this project. First and foremost would be to test the policy on the actual robot based on the test setup described previously, and analyze the results. Another future improvement would be to remove the negative end state from the reward function, so that the robot would have no desire to hit the floor. Including other objects beyond CubeSats would be another improvement, which would help the policy to generalize to any foreseeable object in space. Adding small velocities and torques to the floating objects would also help the generalization, since expecting perfect velocity matching between the system and the object is unreasonable.

Acknowledgment

There are a number of people who helped make this project that I'd like to acknowledge. I'd like to thank Dr. Ferat Sahin for the course structure, feedback, and support throughout this project. I'd also like to thank Karthik Subramanian for the advice, technical tips, and access to the necessary compute power to make this project a success.

References

[1] "Kessler syndrome," *Wikipedia*, Feb. 07, 2020. https://en.wikipedia.org/wiki/Kessler_syndrome

[2] A. Murtaza, S. J. H. Pirzada, T. Xu and L. Jianwei, "Orbital Debris Threat for Space Sustainability and Way Forward (Review Article)," in *IEEE Access*, vol. 8, pp. 61000-61019, 2020, doi: 10.1109/ACCESS.2020.2979505.

[3] Y. Zhao, F. Zhang and P. Huang, "Dynamic Closing Point Determination for Space Debris Capturing via Tethered Space Net Robot," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 5, pp. 4251-4260, Oct. 2022, doi: 10.1109/TAES.2022.3159626.

[4] P. Huang, F. Zhang, J. Cai, D. Wang, Z. Meng and J. Guo, "Dexterous Tethered Space Robot: Design, Measurement, Control, and Experiment," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 53, no. 3, pp. 1452-1468, June 2017, doi: 10.1109/TAES.2017.2671558.

[5] J. Kang, Z. H. Zhu and L. F. Santaguida, "Analytical and Experimental Investigation of Stabilizing Rotating Uncooperative Target by Tethered Space Tug," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 4, pp. 2426-2437, Aug. 2021, doi: 10.1109/TAES.2021.3061798.

[6] Q. Chen, G. Li, Q. Zhang, Q. Tang and G. Zhang, "Optimal Design of Passive Control of Space Tethered-Net Capture System," in *IEEE Access*, vol. 7, pp. 131383-131394, 2019, doi: 10.1109/ACCESS.2019.2939518.

[7] H. Li and H. Baoyin, "Optimization of Multiple Debris Removal Missions Using an Evolving Elitist Club Algorithm," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 1, pp. 773-784, Feb. 2020, doi: 10.1109/TAES.2019.2934373.

[8] Z. Zhang, C. Deng and Z. Deng, "A Diverse Space Target Dataset With Multidebris and Realistic On-Orbit Environment," in *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 15, pp. 9102-9114, 2022, doi: 10.1109/JSTARS.2022.3203042.

[9] J. Yang, X. Hou, Y. H. Hu, Y. Liu and Q. Pan, "A Reinforcement Learning Scheme for Active Multi-Debris Removal Mission Planning With Modified Upper Confidence Bound Tree Search," in *IEEE Access*, vol. 8, pp. 108461-108473, 2020, doi: 10.1109/ACCESS.2020.3001311.

[10] X. Zhang, J. Liu, J. Feng, Y. Liu and Z. Ju, "Effective Capture of Nongraspable Objects for Space Robots Using Geometric Cage Pairs," in *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 1, pp. 95-107, Feb. 2020, doi: 10.1109/TMECH.2019.2952552.

[11] Z. Meng, P. Huang and J. Guo, "Approach Modeling and Control of an Autonomous Maneuverable Space Net," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 53, no. 6, pp. 2651-2661, Dec. 2017, doi: 10.1109/TAES.2017.2709794.

[12] S. Joshi, S. Kumra and F. Sahin, "Robotic Grasping using Deep Reinforcement Learning," *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, Hong Kong, China, 2020, pp. 1461-1466, doi: 10.1109/CASE48305.2020.9216986.

[13] D. Wang, C. Liu, F. Chang, N. Li and G. Li, "High-Performance Pixel-Level Grasp Detection Based on Adaptive Grasping and Grasp-Aware Network," in *IEEE Transactions on Industrial Electronics*, vol. 69, no. 11, pp. 11611-11621, Nov. 2022, doi: 10.1109/TIE.2021.3120474.

[14] S. Wang, Z. Zhou and Z. Kan, "When Transformer Meets Robotic Grasping: Exploits Context for Efficient Grasp Detection," in *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8170-8177, July 2022, doi: 10.1109/LRA.2022.3187261.

[15] Y. Yu, Z. Cao, Z. Liu, W. Geng, J. Yu and W. Zhang, "A Two-Stream CNN With Simultaneous Detection and Segmentation for Robotic Grasping," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 2, pp. 1167-1181, Feb. 2022, doi: 10.1109/TSMC.2020.3018757.

[16] Y. Li, Y. Liu, Z. Ma and P. Huang, "A Novel Generative Convolutional Neural Network for Robot Grasp Detection on Gaussian Guidance," in *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1-10, 2022, Art no. 2517510, doi: 10.1109/TIM.2022.3203118.

[17] H. Cheng, Y. Wang and M. Q. . -H. Meng, "A Robot Grasping System With Single-Stage Anchor-Free Deep Grasp Detector," in *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1-12, 2022, Art no. 5009712, doi: 10.1109/TIM.2022.3165825.

[18] Y. Shi, Z. Tang, X. Cai, H. Zhang, D. Hu and X. Xu, "SymmetryGrasp: Symmetry-Aware Antipodal Grasp Detection From Single-View RGB-D Images," in *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12235-12242, Oct. 2022, doi: 10.1109/LRA.2022.3214785.

[19] H. Cheng, Y. Wang and M. Q. . -H. Meng, "A Vision-Based Robot Grasping System," in *IEEE Sensors Journal*, vol. 22, no. 10, pp. 9610-9620, 15 May15, 2022, doi: 10.1109/JSEN.2022.3163730.

[20] H. Tian, K. Song, S. Li, S. Ma and Y. Yan, "Lightweight Pixel-Wise Generative Robot Grasping Detection Based on RGB-D Dense Fusion," in *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1-12, 2022, Art no. 5017912, doi: 10.1109/TIM.2022.3196130.