Project Report : CS 7643

Daniel Crawford
Georgia Institute of Technology
dcrawford@gatech.edu

Abstract

Time-Optimal Satellite Maneuvering is crucial to the usage of artificial satellites and is an important problem in control theory. Here, I build off of the established success of applying machine learning to this problem by framing it as a Deep Reinforcement Learning Problem. By implementing an Actor-Critic model that learns the distribution of movements of three independent adjusters to orientation, I show not only successful maneuvers, but time optimal maneuvers, for orienting a satellite in both a close-distance and far-distance rotation. Further, future steps are discussed to refine this process and make the model more robust.

1. Introduction/Background/Motivation

This work is an attempt to use Deep Reinforcement Learning methods to train a model to accurately and efficiently orient a satellite in three-dimensional space. The positioning and orientation of a satellite is important to its function. I am seeking a model that can not only orient the satellite to a correct position, but can do so in a time-optimal manner. This is another important aspect of the work.

Currently, the methods to orient satellites are done manually. That is, information is transmitted to a ground support team, which send maneuver instructions back, which the satellite then executes. Audo et al. have found success in using LSTM modeling to accomplish this. This takes advantage of the squential nature of the problem of control theory based movements.

The manual approach that is currently being used is not optimal because it makes large use of the limited computing power on most satellites. If I am able to show that there is promise in automating this procedure by simply placing a model in the satellite to reduce the amount of required manual maneuver time, then this will allow for more capability to be put onto the satellite by increasing its computational capacity.

Because I will be attempting to solve this with Deep Reinforcement Learning, the 'data' that will be used is a simulation of the environment that a satellite operates within. This means that I set up the environment where the satellite was oriented in a particular position, and then was directed to take an action, adjusted itself according to the prescribed movements, and the returned the new state. This simulation is the data used for a reinforcement learning project.

The orientation of a satellite was represented as a quaternion, as described by Bilimoria et al. [2] This accounts for the three dimensions of movement, and for the extra dimension of rotation. The actions were represented as a vector of length 3, which each entry from -1 to 1, reflecting the force with which each thruster was pushing the satellite. Using integration over this force and the quaternions I was able to reflect the new orientation of the satellite.

2. Approach

In order to increase the baseline capabilities of automatically orientating a satellite, I set up a training environment and training process for the model to learn. To solve this problem I took the first steps in performing a simple maneuver. This means that I was training the model to be able to map the actions of adjusting positions to states that lead to a target orientation. To begin, I utilized a 10 degree adjustment of the satellite. This will be the first step in a multi step learning process for the satellite for the model.

Utilizing machine learning to solve this problem has already been explored in Audo et al. [1] They were able to achieve success with LSTM. This approach takes the novel direction of using Deep Reinforcement learning to solve the same problem. The disadvantage of this would lie in the difficulty in instantiating the simulation and securing enough computation to train for all possible maneuvers and target distances. However, the increased flexibility of Deep Reinforcement Learning, and robustness to changes in requirements makes this approach a good candidate for success.

I am utilized an Actor-Critic Model (with similar set up and configuration to Assignment 5). This this is a control problem having the model attempt to learn the distributions of movements of 3 independent thrusters. Therefore, I will set the model up to learn these distributions of the movements based off of input form the current state (orientation and current force applied by thrusters). In for the actor to become accurate, I will optimize on the loss function

$$L_P(\theta) = -\mathbf{E}_{(s,a)\sim\pi}\left[\sum_{t=1}^{T} log\pi_{\theta(a_t|s_t)A_t}\right], \tag{1}$$

where

$$A_t = G_t - V_\phi(s_t) \tag{2}$$

is the advantage, calculated with return:

$$G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k \tag{3}$$

To increase the accuracy of prediction of the Critic Model, I will also optimize on the loss function

$$L_V = \mathbf{E}_{s_t\sim\pi_\theta}\left[(G_t - V(s_t)^)\right] \tag{4}$$

and to encourage exploration, I will also seek to minimize the opposite of entropy:

$$L_E(\theta) = \mathbf{E}_{(s,a)\sim\pi}\left[-H_\theta(a|s)\right] \tag{5}$$

Note that $\pi_\theta(a|s)$ reflects the distribution of of actions to be taken given state s with parameters θ. This set up is taken from assignment 5, and as I was able to achieve success there, wanted to try and apply what I learned to a new problem here.

2.1. Environment, Task, and Metrics

Naturally one of the most difficulty parts of of this project was the coding of the environment. This required much study and careful design. However, I was able to leverage the previous work and translate the simulation (environment) to python. The environment was a deterministic control environment where the input form each thruster was a step (length 3 vector) and using quaternion integration was able to simulate the movement over a time step. Therefore, the metric used was success (binary), did it make it to within 5%of the target?

2.2. Sparse Reward Problem

The second, and biggest challenge that I anticipated was the sparse reward problem. Since this project was about navigating a satellite to a desired target, the reward had to be shaped beyond the binary, achieved or not achieved. This is an inherent and well studied problem in (Deep) Reinforcement learning. I tried multiple reward systems, both continuous functions and discrete rewarding system. Eventually I found a system of rewards that seemed to work: for every 10% closer to the target orientation, the reward for that step doubled. This was developed through an iterative process and the realization that in order for the model to learn what the target was, and how to move there, the rewards must grow exponentially, not linearly, to signify stronger reward. While linear may indeed work, I conjecture that exponentially increasing reward for reach check point would perform better.

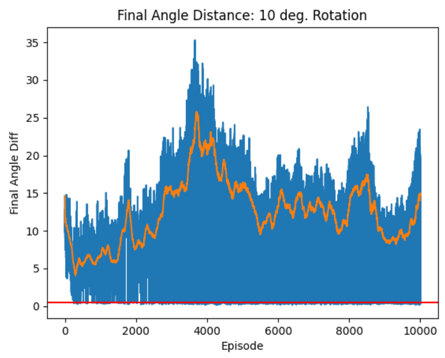
Another problem I had to address in the implementation was the concern of time optimality. I am seeking to orient the satellite in a time-optimal fashion, so, model must, in some way account for the reward. I did try and use both another loss function, and to add a time component to the reward. I struggled to find success with these, but was able to account for time by summing the reward of each step and creating the episodic reward. Instead of training the model to adjust the satellite and only be rewarded at the final position, I calculated the reward from each time step and summed together. This, then, rewarded the model for reaching the target position quicker (as it would accrue more of an exponentially higher reward) but also stay there (as opposed to randomly swinging in and out of the target position, again to gain more reward.)

3. Experiments and Results

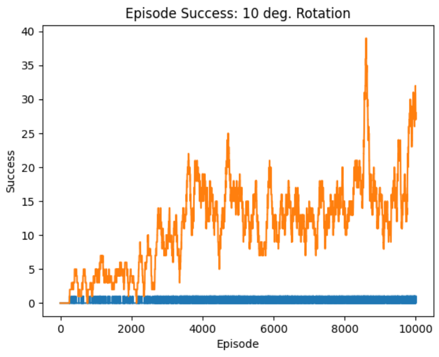
Success was measured first by achieving the correct orientation. Secondly, the time taken (number of steps done) was accounted for and considered when measuring optimality. These two components make up a successful run.

While it will be be a good extensions to run a set of trial simulations with the trained networks, the scope of this project, we will stick to analyzing the results of the training. As a test simulation will be the same as the training simulation (we will not present the model with a novel learning scenario) I will use the moving average of the most recent 100 episodes to convey overall success of the model. Both a small (10 degree) and big (45 degree) maneuver will be be shown here.

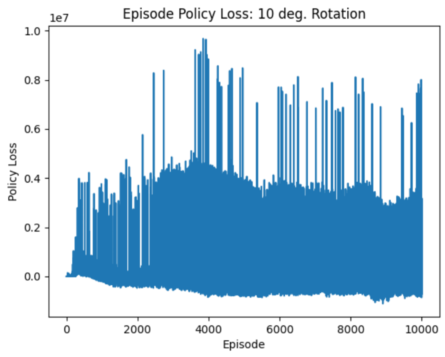
3.1. Small Degree Maneuver

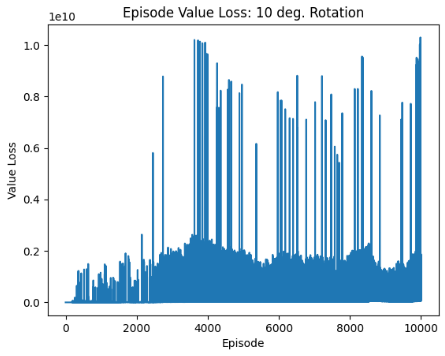


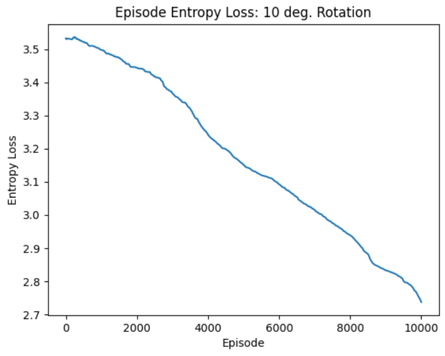
This figure shows the individual and average trend for 10,000 episodes of the 10 degree adjustment of the distance from target at the final episode. Note that the way I have the simulation set up is by having the satellite start 10 degrees away form the target and propel itself to where it needs to go. We can see that after about 4000 episode, there is a general decrease in the average final angle distance. The red horizontal line reflect the 5% of target distance that we are setting as the goal. (This will be discussed later.) Note there are two main reasons that that the trend is not lower given all of the success (as shown on the next plot). First, this is the FINAL position distance, so the actor may have moved the satellite after achieve a successful movement. Second the movements are so fine tuned that it will be difficult for the actor to move, due to physical restraints, the satellite only a fraction of a degree.



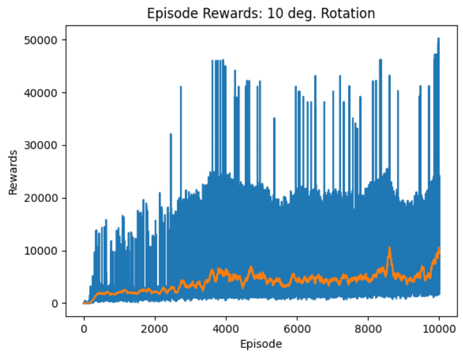
The plot of successes here (orange indicate sum of last 100 rows, with each success valued at 1) shows a positive trend, which suggests that the agent is learning over time to complete this task. With respect to time optimality, I have set the known time (more on this later too) as the max time, so any success is within the current state-of-the-art bounds. It seems that with more training more success will be reached.





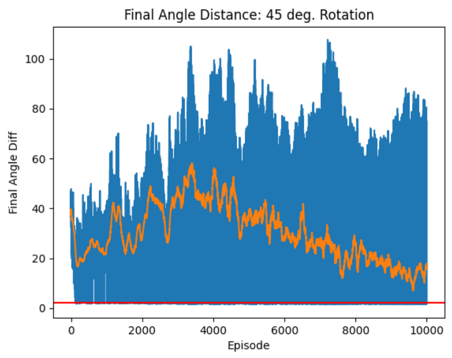


These three plots show the loss function values (weighted). Note the correlations in spikes. These reflect the large rewards that are given for finding and staying at the desired orientation. The general downward slope the the entropy loss suggests that exploration decreased as training went on, which is of course desirable as to create a more stable model.

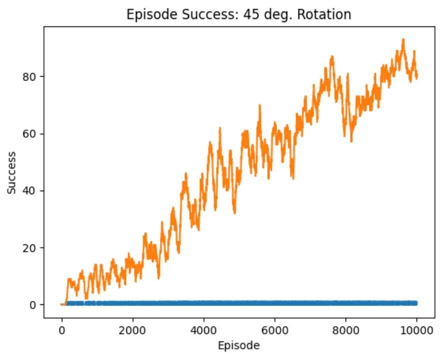


The rewards, as discussed earlier, were set up to be exponentially increasing in value for linearly decreasing distance from the target orientation.

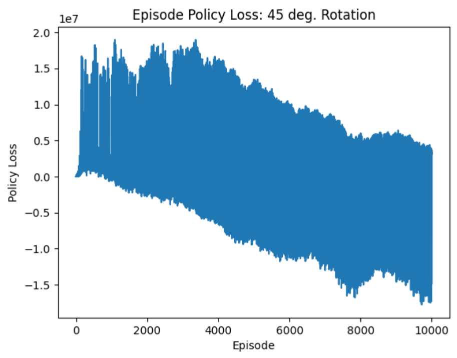
3.2. Large Degree Maneuver

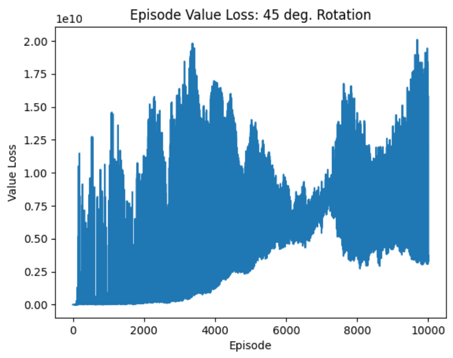


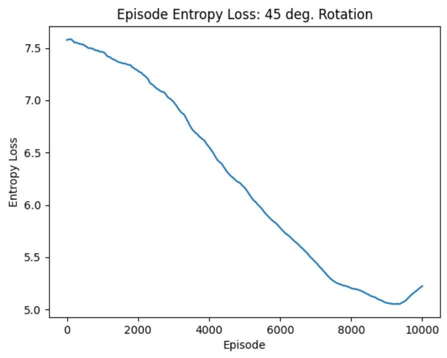
For the large degree maneuver, we see a much more pronounced decrease in the distance of the satellite to the target in the final step. Because the angle of desired rotation was so much larger, the reward was able to take effect of larger span of values and push the satellite to the correct position. Also note how the increase in target movement muted the effects of attempting very fine maneuvers.



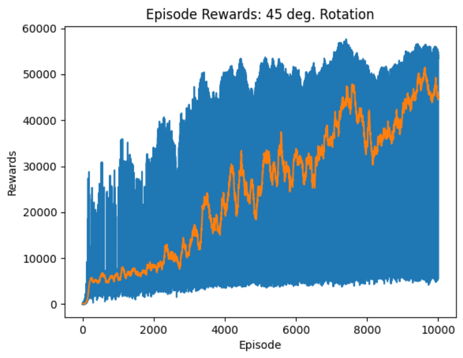
The accumulated count of successes over the previous 100 episodes accumulates dramatically as training goes on. This suggests that there is indeed some kind of learning happening, that the agent is able to reliably make it to the desired orientation (ie not jsut out of chance).







The Policy Loss can be shown to be generally decreasing, which tracks with understanding. The Value loss has an odd convergence in the middle, which I am not really sure how to explain other than chance. The Entropy loss, jsut as in teh 10 Degree maneuver shows decreasing which we would expect to see.



The episodic reward of the simulation increases in line with what would be expected for a learning agent. As such, I am comfortable sating that the agent was indeed learning.

3.3. Discussion of Results

For both the small and big maneuver, I used the same hyper parameters and agent settings to train the two. It appears that the agent did not plateau on its learning and could have achieved higher success rates and greater rewards. I am confident, however, that I accomplished my goal of creating an agent that could learn time-optimal maneuvering.

4. Other Sections

5. Relation to Deep Learning

With any Deep Learning problem, we need to consider what features the network is learning. I think that, for this project, the features that the network is picking up on is the tie between orientation and reward. When I used continuous reward functions, there was not as much success as discrete. So, I think that the model was able to find the features of orientation that correspond, reliable, and exponentially increasing reward. This then allowed it to take the action to acheive that reward.

The structure of the problem was that of a control problem. It was very similar to Assignment 5. Therefore, I sought to optimize the distribution of movements on the thrusters of the satellite to orient it optimally given a current state. This independent control of individual thrusters reflects the control theory aspects of the problem itself and is a good candidate for the reason that this approach worked (over my Deep-Deterministic Policy Gradient approach).

The model itself consisted of an input layer, 2 fully connected layers, and an output layer. Each of these had learned parameters. These were learned with ADAM optimizer, done in PyTorch. (I opted at have all of the loss functions be optimized together.) The hyper parameters were tuned manually and not learned. I found greatest success with hidden layer size of 128 nodes, γ of 0.99999, and a learning rate of 0.0001

The neural network expected a vector of length 7 as its input: the first 4 entries reflected the current orientation (quaternion) and the second 3 reflected the current force (scaled) applied by the thrusters. All of these value were in $[-1, 1]$. There was not so much data to pre-process but building the simulation was a crucial task. I had to ensure that the simulation was valid and accurately reflected real-world movements. The loss functions, which were previously discussed consisted of policy loss, to perform more optimal movements, value loss, to assess the value of movements more accurately, and entropy loss to encourage exploration were used. The loss function optimized on was a weighted sum of these three.

Over-fitting was not a concern here, and the model did appear to generalize well to both small and large movements. A validation run will be needed to be more confident in the results, however the aim to create an agent that can learn was accomplished.

What hyper parameters did the model have? How were they chosen? How did they affect performance? What optimizer was used?

The learning rate used was 0.0001. I found that a larger learning rate produced an unstable model which would rise and fall unpredictably in reward and performance. The γ value was perhaps the most important value I had to tune. IT was only with high value (> 0.999) that I was bale to achieve best success. It appears that for this problem a large amount of weight is needed across recent actions.

The deep learning framework PyTorch was used to complete this project.

What existing code or models did you start with and what did those starting points provide? All code (Environment, Metrics, and Learning) were coded solely by myself. I was able to leverage code in MatLab Provided by the authors of Audo et al. [1] for this project, however needed to rewrite everything in order understand and have it work with my code base.

Briefly discuss potential future work that the research community could focus on to make improvements in the direction of your project's topic.

Future research for this project is ongoing. The next step would be to run many test simulations. I also thing that expanding the number different targeted degree movements is important. A system of Curriculum Learning Could be set up in order to have the agent learn to orient to multiple different target points.

6. Nature of the Project

This project was born out of a professional network that I have with the Naval Post-Graduate School, out of Monterey, California. LTC Brian Wade, Ph.D, and Professor Mark Karpenko were able to provide me with the initial problem. From there, I was able to leverage the MatLab code of Dr. Karpenko for the basic simulation, however all code used for this project was written solely by by self and for this course.

However, from them I able able to ascertain two things. The first was the maximum possible time it would take a satellite to orient itself. In order to ensure time-optimal maneuvers, I cross referenced the max number of steps in the simulation environment that I would allow the agent to take with the theoretical maximum. This ensured that the agent was learning in a time dependent fashion. A future area of research would be trying to do this without the artificial cap. Second, as subject-matter-experts, both of my point of contacts suggested that the agent need only to be within 5% of the target orientation to be deemed successful. They stated that from there, manual maneuvers would be possible and efficient. More of their work, along with LT Vincent Audo, can be found in the referenced papers.

7. Work Division

All work was done by Daniel Crawford for the purposes of this course: Table 1.

Student Name	Contributed Aspects	Details
Daniel Crawford	Entire Project	Coded simulation in Python, modified code from Assignment 5 to fit specifications, trained the agent, hypertuned, sculpted reward function, captured results.

Note that this project was taken individually, with the explicit written permission of Prof. Kira, https://edstem.org/us/courses/15925/discussion/1170203

Table 1. Contributions of team members.

7.1. References

[1] Audo, V., Karpenko, M., Wade, B. M. (2020, October). IMPLEMENTATION OF TIME-OPTIMAL ATTITUDE MANEUVERS USING LONG-SHORT TERM MEMORY NEURAL NETWORK. Naval Postgraduate School.

[2] Bilimoria, KD., Wei, B., (1993, June). Time-Optimal Three-Axis Reorientation of a Rigid Spacecraft. JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS.