

Learning attitude control

K. Djebko¹, F. Puppe¹, S. Montenegro², T. Baumann², M. Faisal²

¹ Julius-Maximilians-Universität Würzburg, Chair of Computer Science VI, Artificial Intelligence and Knowledge Systems, Am Hubland, 97074 Würzburg, Germany
Phone: +49 931 31 - 86405, Mail: kirill.djebko@uni-wuerzburg.de

² Julius-Maximilians-Universität Würzburg, Chair of Computer Science VIII, Aerospace Information Technology, Emil-Fischer-Str. 70, 97074 Würzburg, Germany

Abstract: Attitude control algorithms are unintuitive and difficult to master. As a result, many pico- and some nano-satellites are built that have attitude control hardware, but still cannot stably control their attitude. Parameterizing attitude control algorithms is done through lengthy manual testing and tuning. The recent advances in the field of Artificial Intelligence (AI) and machine learning open new possibilities of controller design, by applying deep reinforcement learning to train a controller instead of manually fine tuning it. Such an approach would enable a more efficient attitude controller development process.

Our work applies the Proximal Policy Optimization (PPO) Deep Reinforcement Learning (DRL) algorithm in combination with the Basilisk Astrodynamics Simulation Framework, to train an attitude controller in the form of an AI-agent with a realistic simulator. We evaluated our approach using two spacecraft configurations, the 4 kg InnoCube CubeSat which is in development and a hypothetical 750 kg spacecraft, and outline the differences in behavior regarding training and execution as well as general lessons learned from designing the observation space and the reward function. In particular we tackle the problem of variations in the inertia tensor of the respective spacecraft to increase the robustness of the controller.

1. INTRODUCTION AND RELATED WORK

Reinforcement Learning [1] has seen a renaissance with neural network based Deep Reinforcement Learning as it gained widespread attention across various domains since 2013 [2-4]. Besides the application for games like Atari [2, 3] and Go [5, 6] there has recently been increased interest in DRL in the aerospace community [7, 8] and especially the training of UAV [9-11] and spacecraft attitude controllers [12-15]. Some work focuses on discrete actions, e.g., n discrete torques from which the controller can choose [13], while others employ continuous action spaces [14]. Most work however assumes fixed moments of inertia. In case the spacecrafts assumed moments of inertia do not match its actual ones or change due to i.e., fuel consumption, the controller performance may decline.

We therefore investigate the effect of changing moments of inertia on a DRL attitude controller. We employ the Stable-Baselines3 [16] implementation of Proximal Policy Optimization DRL algorithm [17] within the OpenAI Gym framework [18]. The PPO algorithm is a popular DRL algorithm used to realize various DRL tasks ranging from video games to fine-tuning complex language models like the recently released ChatGPT by OpenAI¹. Despite great progress in the area of DRL, sample inefficiency is still a major challenge. Therefore, a simulator is necessary for training that can generate enough training samples in reasonable time. As simulator the Basilisk Astrodynamics Simulation Framework² was chosen. It provides the tools necessary to customize the simulated

¹ <https://openai.com/blog/chatgpt/>

² <https://hanspeterschaub.info/basilisk/>

spacecraft and comes with a Python interface for seamless integration into the Python based DRL environment.

2. MODELS

Two models were considered. The first model is based on the 4 kg InnoCube Cubesat [19, 20], jointly developed by the Julius-Maximilians-Universität Würzburg, the TU-Braunschweig and the TU-Berlin. The moments of inertia were computed using a CAD model of the satellite. The second model is a hypothetical 750 kg spacecraft (“LargeSat”) with high inertia to contrast InnoCube. Both models feature three reaction wheels with a max torque of 0.002 Nm for InnoCube and 0.2 Nm for LargeSat. The initial moments of inertia, given by their diagonals, were (0.0427, 0.0427, 0.0068) kg·m² for InnoCube and (900, 800, 600) kg·m² for LargeSat. The resolution of the simulation and therefore the duration of one time step was set to 1 second.

3. HYPERPARAMETERS AND OBSERVATION SPACE

The reinforcement learning procedure requires the design of a so called observation space, that is the input the AI-agent receives, an action space, defining which actions the AI-agent can take, and a so called reward function which is to be defined and optimized. During training the agent receives an observation and predicts an action that is iteratively optimized to maximize the reward. The default network architecture was replaced by a three-layer network with 128 neurons each and tanh activation function for the actor and the critic respectively. Early experiments showed catastrophic forgetting/unlearning during training, likely stemming from occasional large policy updates. To stabilize the training process, the `max_grad_norm` and `target_kl` parameters were adjusted to limit the magnitude of policy change during training. Additionally the learning rate was annealed quadratically. The hyperparameters used are listed in Table 1.

Table 1: Hyperparameters used during the training of the AI-agents

n_steps	gamma	max_grad_norm	target_kl	learning_rate	batch_size
2048	0.95(InnoCube) 0.99(LargeSat)	0.1	0.05	Quadratically annealed. Init. value $5e^{-4}$ ($1e^{-5}$ for post-training)	64

The final observation space was chosen to be a 25-element vector consisting of $(err_quat_{t_{i-1}}, err_quat_{t_i}, a_{t_{i-1}}, err_rr_{t_{i-1}}, err_rr_{t_i}, att_quat_{t_{i-1}}, att_quat_{t_i})$, with err_quat being the 4-element attitude error quaternion, err_rr the 3-element spacecraft rotation rate error, a the 3-element vector of commanded torques as agent actions and att_quat the 4-element attitude quaternion. The subscript t_i indicates the current timestep and t_{i-1} the previous one. This observation space allows inferring moments of inertia implicitly given two consecutive states with the action that caused the state transition.

4. REWARD FUNCTION

The value for the reward function RF_{t_i} at timestep t_i is evaluated as follows:

$$\begin{aligned}
 err_att_{t_i} &= 1 - |err_quat_{t_i}[0]| \\
 \Delta angle_{t_i} &= err_att_{t_i} - err_att_{t_{i-1}} \\
 p_{t_i} &= -norm_{t_i} = -\|err_rr_{t_i}\| \\
 RF_{t_i} = reward_{t_i} &= p_{t_i} + r_{t_i}
 \end{aligned}
 \quad
 r_{t_i} = \begin{cases}
 9 + \frac{1}{norm_{t_i} + 0.01} & , \text{if } err_att_{t_i} < 3.8e^{-5} \\
 e^{-\frac{err_att_{t_i}}{0.14}} & , \text{else if } err_att_{t_i} < err_att_{t_{i-1}} \\
 e^{-\frac{\Delta angle_{t_i}}{0.14}} \cdot 0.1 & , \text{else if } \Delta angle_{t_i} < \Delta angle_{t_{i-1}} \\
 e^{-\frac{err_att_{t_i}}{0.14}} - 1 & , \text{else}
 \end{cases}$$

with $err_quat_{t_i}[0]$ being the scalar component of the error quaternion. In the following, the subscript t_i is omitted due to brevity. It is to be noted, that the vector of spacecraft rotation rate errors err_rr is scaled by a spacecraft-configuration specific scalar, so that no value of the rate vector exceeds 1 during training. For InnoCube this scalar is 300 and 4 for LargeSat. The reward function used is an extension of the reward function from [13] and conditions the agent to achieve the goal-attitude while subtracting the regularization term p , in order to prevent drastic rate changes. The reward is maximized, when the goal-attitude is achieved as fast as possible and then held steady.

Since light spacecraft, unlike their inert and slow counterparts, may suffer from oversteering, we introduced a modification to the training process for this type of spacecraft. We replaced the first condition of the r term with $\frac{1}{norm+0.01} - 91$. This modified reward function is called RF_{pt} with subscript pt for post-training, which refers to a second training run using the initially trained agent as base and RF_{pt} as reward function. The post-training reward function however is unsuited for initial training as it greatly increases the sample inefficiency due to the harsh rate penalty reducing the incentive of attaining the goal-attitude during early training.

5. TRAINING AND RESULTS

Most works on AI attitude controllers assume fixed moments of inertia. This however may not always be the case in reality. Changes in the moments of inertia may lead to a reduction in the quality of the controller performance [12]. The authors of [21] report that an agent trained on specific moments of inertia is applicable to larger inertia but struggles for lower inertia. The latter most likely stems from oversteering, which small and light satellites are especially prone to. To tackle this issue, during training, random moments of inertia were generated by first taking the minimum and maximum values from the original moments of inertia and generating a new random diagonal by drawing from the interval $[min \cdot 0.6, max \cdot 1.4]$ for each value independently.

Both the InnoCube- and the LargeSat-configuration were trained for 200 million steps each, which took about 1.5 days using a parallelization factor of 8 on an Intel i7 9600. Since the InnoCube-configuration features low inertia but can produce relatively large torques, the satellite is prone to drastic oversteering if the agent makes a mistake. We therefore applied the aforementioned post-training to the InnoCube-configuration. While LargeSat did not suffer from this problem due to its high inertia, the increased episode length of 1000 as compared to 50 for InnoCube significantly reduced the amount of sampled moments of inertia during training.

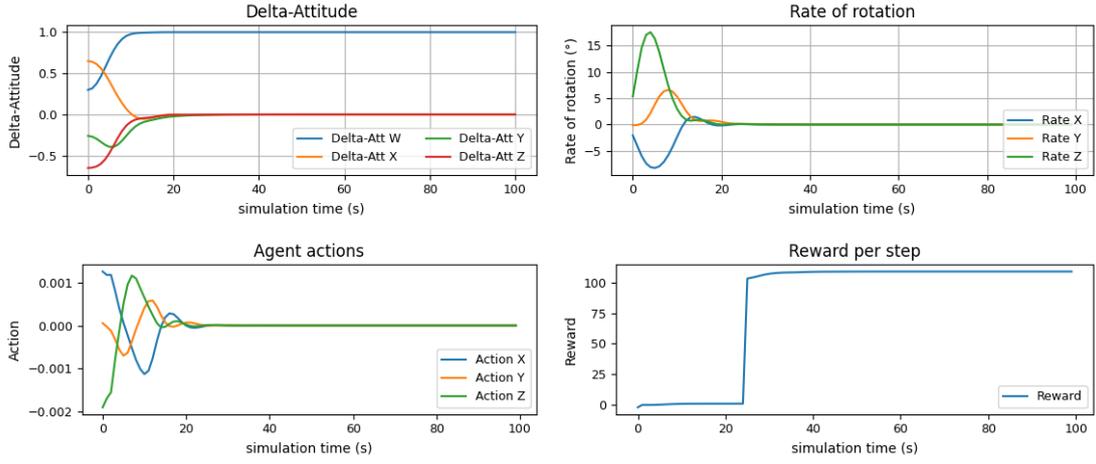


Figure 1: Attitude (top left), rotation rates (top right), agent actions (bottom left) and reward per step (bottom right) for an example run with random inertia for the post-trained InnoCube-agent.

Figure 1 shows an example run with random moments of inertia for the post-trained InnoCube-agent representative for the general performance of the controller. The agent was able to reliably attain the goal-attitude fairly quickly and stabilize the spacecraft.

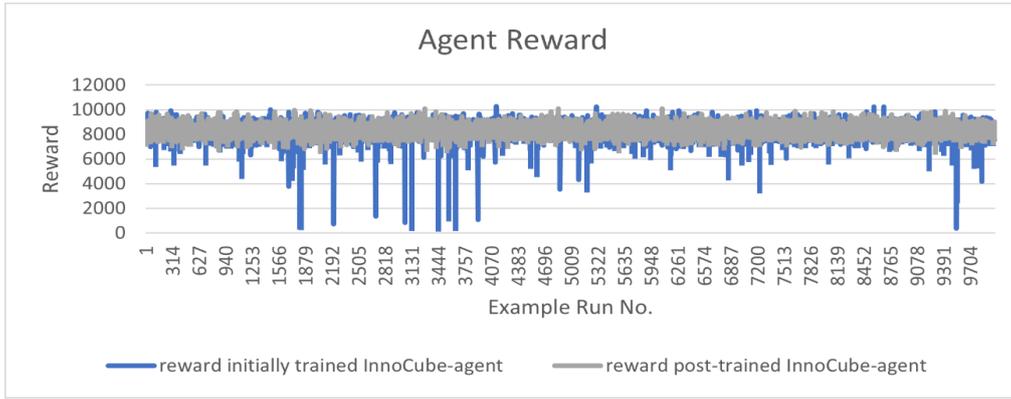


Figure 2: Comparison of the reward for 10000 runs with random initial-attitude, goal-attitude and moments of inertia for the originally trained (blue) and the post-trained (gray) InnoCube-agent evaluated on the original reward function.

Figure 2 shows the reward for 10000 runs with random initial-attitudes, goal-attitudes and moments of inertia for the initially and post-trained InnoCube-agent. The occasional drops in reward show the oscillatory behavior of the originally trained agent. These outliers have been eliminated using post-training.

Table 2: Mean reward and reward standard deviation for the originally trained and the post-trained InnoCube-agent based on 10000 runs with random initial-attitudes, goal-attitudes and moments of inertia.

Agent	Reward mean	Reward standard deviation
InnoCube No post-training	8427.79	541.29
InnoCube Post-trained	8313.54	394.70

Table 2 shows the mean reward and reward standard deviation for 10000 random runs of the InnoCube-configuration. The post-trained agent outperformed the initially trained

one, since it reduced oscillatory oversteering after attaining the goal-attitude, which is consistent with the observations from Figure 2Figure 2.

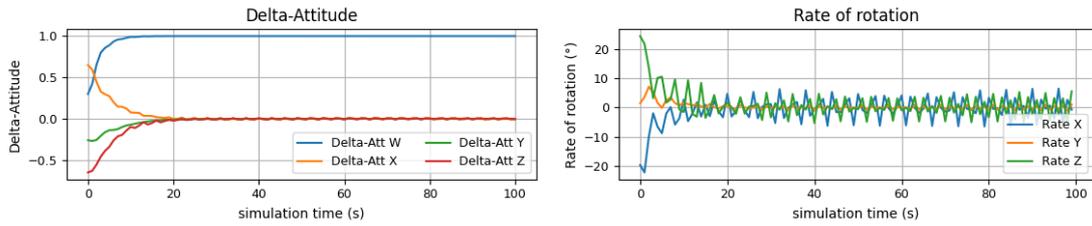


Figure 3: Attitude (left) and rates of rotation (right) for a bad example run of the initially trained InnoCube-agent with random moments of inertia, random initial attitude and random goal-attitude.

Figure 3 shows a concrete example for the oscillations of the originally trained InnoCube-agent and Figure 4 shows the improvement from applying the post-trained InnoCube-agent to the same initial-attitude, goal-attitude and moments of inertia. Even though such behavior is rare even for the initially trained agent, it is undesired behavior which is to be eliminated when the controller is to be used in a real world scenario. By applying post-training, the agent was corrected successfully.

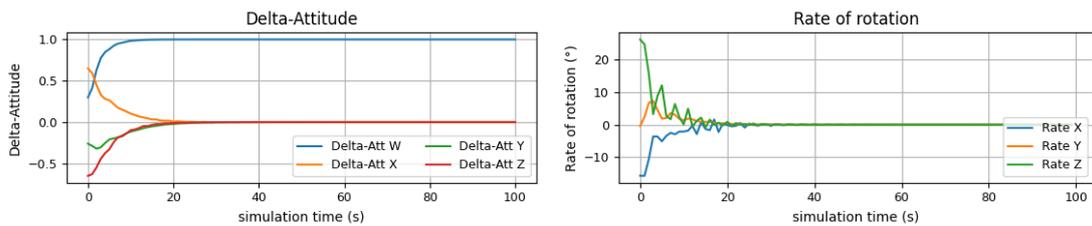


Figure 4: Attitude (left) and rates of rotation (right) for the setup from Figure 3 after post-training with reward function RF_{pt} .

To contrast the light InnoCube, the training was repeated for the LargeSat-configuration. Due to the increased episode length, the discounting factor gamma was increased from 0.95 to 0.99, causing the agent to discount immediate rewards stronger in favor of future rewards. The underlying attitude-control problem was easier in nature as the high inertia made the spacecraft less prone to oversteering.

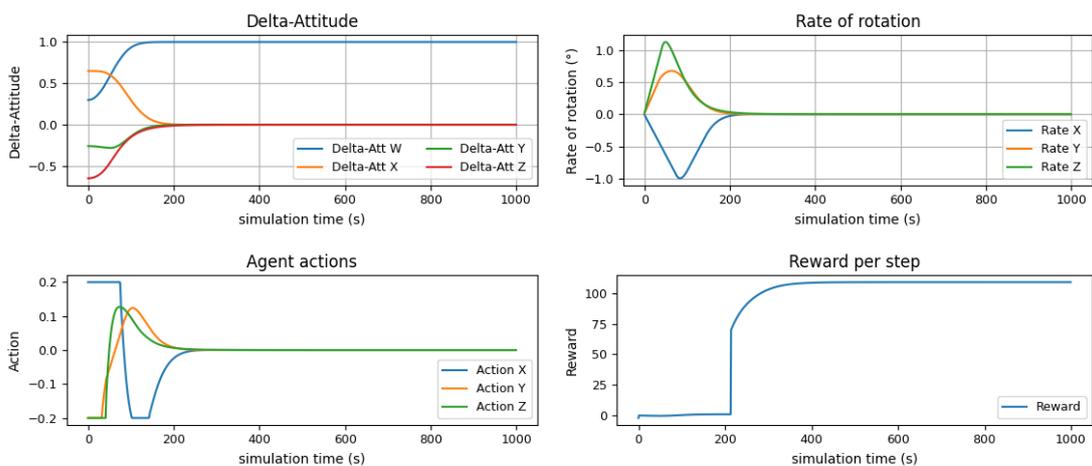


Figure 5: Attitude (top left), rotation rates (top right), agent actions (bottom left) and reward per step (bottom right) for an example run with random moments of inertia for the LargeSat-agent.

Figure 5 shows the result of an example run for the LargeSat-agent for random moments of inertia and with the same initial- and goal-attitude as Figure 1 for better comparison. The agent was again able to reliably and swiftly attain the goal-attitudes.



Figure 6: Reward for 10000 runs with random initial-attitude, goal-attitude and moments of inertia for the LargeSat-agent evaluated on the original reward function.

Figure 6 shows the reward for 10000 random runs for the LargeSat-agent. No significant drops in reward can be observed, indicating stable behavior.

Table 3: Mean reward and reward standard deviation for the LargeSat-agent based on 10000 runs with random initial-attitudes, goal-attitudes and moments of inertia.

Agent	Reward mean	Reward standard deviation
LargeSat	82019.30	6016.68

Table 3 shows the mean reward and reward standard deviation for 10000 random runs of the LargeSat-configuration. The reward standard deviation is low compared to the mean reward, confirming reliable attainment of the goal-attitudes. The increase in absolute numbers for both values stems from the increased episode length.

For completeness, training of the InnoCube-agent was also performed using a gamma value of 0.99 and training of the LargeSat-agent was performed using a gamma value of 0.95. For the InnoCube-configuration the now too far sighted agent showed a longer initial exploration phase during training and a decrease of quality in regards to mean reward and reward standard deviation ($\mu=6848.22$ | $\sigma=2285.14$). It also suffered from strong oversteering and often failed to attain the goal-attitude. The post-trained 0.99 gamma InnoCube-agent (RF_{pt}) performed slightly better but still suffered from oscillatory behavior ($\mu=7146.30$ | $\sigma=574.85$), as the focus on a longer time horizon seems to lead to a reduction of quality in regards to immediate responses. For the agile InnoCube-configuration this results in oversteering.

For the LargeSat-agent trained using 0.95 gamma, the training stability and mean reward decreased while the reward standard deviation increased ($\mu=76611.33$ | $\sigma=11365.79$). Further post-training using RF_{pt} was performed for the 0.95 and 0.99 gamma LargeSat-agents respectively. The results were that while the 0.95 gamma agent showed minor but

insufficient positive change ($\mu=76161.58|\sigma=10341.88$), the 0.99 gamma agents performance slightly decreased ($\mu=76632.90|\sigma=6378.11$). This stems from RF_{pt} enforcing a slow approach to the goal-attitude while the far-sighted 0.99 gamma agent quickly increases its rates and just as quickly decreases them when approaching the goal-attitude. Additionally the post training was less stable initially, indicating that quite some re-learning took place.

6. DISCUSSION

We have presented an AI attitude controller that accommodates for changes in the spacecrafts moments of inertia by employing Deep Reinforcement Learning. Our controller showed to be robust to both increases and decreases in the moments of inertia within a wide range. We investigated AI attitude controllers for two different types of spacecraft, light satellites with low inertia and heavy spacecraft with high inertia, and discussed the changes in the type of training. Our results show, that light spacecraft require different focus, as the problem of oversteering can degrade the quality of the controller.

7. FUTURE WORK

Building on the lessons learned from considering a light and heavy spacecraft we want to investigate measures to decrease the training time of the AI-controller. This includes an investigation into using an agent trained on one type of spacecraft as a base for post-training to adapt to another type of spacecraft. Ideally a transfer of the magnitude of InnoCube to LargeSat. This type of transfer learning could speed up the adaption of the AI-controller to vastly different spacecraft types, as there seems to be at least some common structure. Additionally we want to verify our AI-agent in orbit on board the InnoCube satellite.

FUNDING

This work was done as part of the VeriKI project (FKZ: 50RA2012), which is funded by the German Federal Ministry of Economic Affairs and Climate Action (BMWK), based on a decision of the Deutsche Bundestag (The Parliament of the Federal Republic of Germany).

ACKNOWLEDGMENTS

We thank the members of the German Space Agency DLR for their support of this work, namely (in alphabetical order) Markus Bibus, Sabine Philipp-May, Christian R. Prause, Bernd Sommer and Klaus Vollmann.

REFERENCES

- [1] R. Sutton, F. Bach, and A. Barto, Reinforcement learning: An introduction. Massachusetts: MIT Press Ltd, 2018.
- [2] V. Mnih, K. Kavukcuoglu, D. Silver, et. al., "Playing Atari with Deep Reinforcement Learning," NIPS Deep Learning Workshop 2013, 2013.
- [3] V. Mnih, K. Kavukcuoglu, D. Silver, et. al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015.

- [4] T. Lillicrap, J. Hunt, A. Pritzel, et. al., “Continuous control with deep reinforcement learning,” arXiv preprint arXiv:1509.02971, 2015.
- [5] D. Silver, A. Huang, C. Maddison, et. al., “Mastering the game of go with deep neural networks and Tree Search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- [6] D. Silver, J. Schrittwieser, K. Simonyan, et. al., “Mastering the game of go without human knowledge,” *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
- [7] B. Gaudet, R. Linares, and R. Furfaro, “Deep Reinforcement Learning for six degree-of-freedom planetary landing,” *Advances in Space Research*, vol. 65, no. 7, pp. 1723–1741, 2020.
- [8] A. Harris, T. Thibaud, and H. Schaub, “Spacecraft decision-making autonomy using deep reinforcement learning,” 29th AAS/AIAA Space Flight Mechanics Meeting, pp. 1–19, 2019.
- [9] E. Bohn, E. Coates, S. Moe, et. al., “Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization,” 2019 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 523–533, 2019.
- [10] W. Koch, R. Mancuso, R. West, et. al., “Reinforcement learning for UAV attitude control,” *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 2, pp. 1–21, 2019.
- [11] X. Qiu, C. Gao, K. Wang, et. al., “Attitude control of a moving mass-actuated UAV based on Deep Reinforcement Learning,” *Journal of Aerospace Engineering*, vol. 35, no. 2, 2022.
- [12] D. Gao, H. Zhang, C. Li, et. al., “Satellite attitude control with deep reinforcement learning,” 2020 Chinese Automation Congress (CAC), pp. 4095–4101, 2020.
- [13] J. Elkins, R. Sood, and C. Rumpf, “Autonomous spacecraft attitude control using deep reinforcement learning,” 71st International Astronautical Congress, 2020.
- [14] J. Elkins, R. Sood, and C. Rumpf, “Adaptive continuous control of spacecraft attitude using deep reinforcement learning,” 2020 AAS/AIAA Astrodynamics Specialist Conference, pp. 420–475, 2020.
- [15] A. Balossino, G. Reverberi, C. Menegazzo, et. al., “Artificial Intelligence for CubeSats: an application for underactuated attitude control,” *The 4S Symposium: Small Satellites, Systems and Services*, 2022.
- [16] A. Raffin, A. Hill, A. Gleave, et. al., “Stable-baselines3: reliable reinforcement learning implementations,” *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 12348–12355, Jul. 2022.
- [17] J. Schulman, F. Wolski, P. Dhariwal, et. al., “Proximal Policy Optimization Algorithms,” arXiv preprint arXiv:1707.06347, 2017.
- [18] G. Brockman, V. Cheung, L. Pettersson, et. al., “OpenAI Gym,” arXiv preprint arXiv:1606.01540, 2016.
- [19] B. Grzesik, T. Baumann, T. Walter, et. al., “InnoCube—a wireless satellite platform to demonstrate Innovative Technologies,” *Aerospace*, vol. 8, no. 5, p. 127, 2021.
- [20] S. Montenegro, T. Baumann, E. Dilger, et. al., “InnoCube Der erste Drahtloser Satellit,” *Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V.*, Nov. 2022.
- [21] K. Ito and T. Yanao, “Adaptive Attitude Control of Spacecraft via Deep Reinforcement Learning with Lyapunov-Based Reward Design,” *The 31th Workshop on JAXA Astrodynamics and Flight Mechanics*, 2021.