TITLE

A. Balossino [1], G. Reverberi [2], L. Sarti[3], C. Menegazzo[4], S. Ciaglia [5]

[1] Argotec S.r.l, Via Cervino 52, 10155 Turin Italy, alessandro.balossino@argotecgroup.com
[2] Argotec S.r.l, Via Cervino 52, 10155 Turin Italy, gianmarco.reverberi@argotecgroup.com
[3] Argotec S.r.l, Via Cervino 52, 10155 Turin Italy, lorenzo.sarti@argotecgroup.com
[4] Argotec S.r.l, Via Cervino 52, 10155 Turin Italy, carlo.menegazzo@argotecgroup.com
[5] Argotec S.r.l, Via Cervino 52, 10155 Turin Italy, sarah.ciaglia@argotecgroup.com

Artificial Intelligence for CubeSats: an application for underactuated attitude control

ABSTRACT

In the last few years, there has been an increased interest in performing missions in deep space with small satellites: despite their reduced size, they can provide valuable scientific and technological returns.

Limited redundancy and communication lags may pose significant threats to the mission continuity in case of non-nominal conditions. Endowing satellites with greater autonomy to handle off-nominal situations is therefore a very promising solution.

Allowing small satellites to independently handle their attitude in a robust way, even in critical situations, is an enabling factor towards full autonomy: for this reason, we focused on the attitude control problem as starting use-case. Here, we present our approach towards safe use of AI technology onboard spacecrafts, describing our solution devised to tackle the peculiar condition of under-actuation.

The final system, based on Neural Network algorithms, in the frame of Deep Reinforcement Learning, uses a control policy trained to reach a target attitude starting from any initial position, with performances useful to maintain mission operations pursuance even in under-actuation conditions.

In the paper we will describe the development flow that will lead to the in-orbit validation of the system on ArgoMoon, a 6U satellite that will be launched in the frame of the Artemis-1 mission.

1 INTRODUCTION

It is common for small satellites to have a limited level of redundancy to comply with the challenging mass and volume budget: aside of the redundancy at subsystem level, which is often impossible, it is even challenging to have a redundant set of actuators and sensors.
In the missions Argotec is currently working on (ArgoMoon [1] and LICIACube [2]), that foresee the utilization of 6U CubeSats in deep space, this aspect is even more challenging given the difficulty in communicating with the spacecraft and the impossibility of using actuators that exploit the Earth’s magnetic field.
Both ArgoMoon and LICIACube have a primary mission phase which is relatively short, however it is very likely in the near future there will be missions implemented with Cubesats/microsats in deep space with a long duration and, in that case, developing systems capable of dealing with off-nominal events becomes more appealing to guarantee the success of the mission.
The Attitude Determination and Control Systems (ADCS) installed on both the missions feature three reaction wheels installed on the three axes of the satellites and the failure of one actuator has been identified as an interesting use-case to analyze the potential of applying AI-based solutions

to control one of the key functions of the satellite during off-nominal events.
In case of failure of one of the actuators, the satellite becomes underactuated, which is a condition that cannot be handled easily by exploiting analytical solutions [3][4]. On the other hand, Machine Learning (ML) – and in particular Reinforcement Learning – have been identified as interesting technique for dealing with this particular application and to start demonstrating that this approach can provide valuable results that go beyond the application of the attitude control problem.

2 REINFORCEMENT LEARNING

Reinforcement Learning (RL) is a Machine Learning (ML) paradigm that deals with the problem of learning the optimal behavior to act in an unknown environment. A RL agent collects data through the interaction with the environment, observing the effect of its own behavior. It learns, by trials and errors, how to map a particular observation to the appropriate action to maximize a payoff signal. This is comparable to how humans and other biological agents learn, experiencing the consequences of their interactions with the environment and gaining knowledge about how to influence what happens.

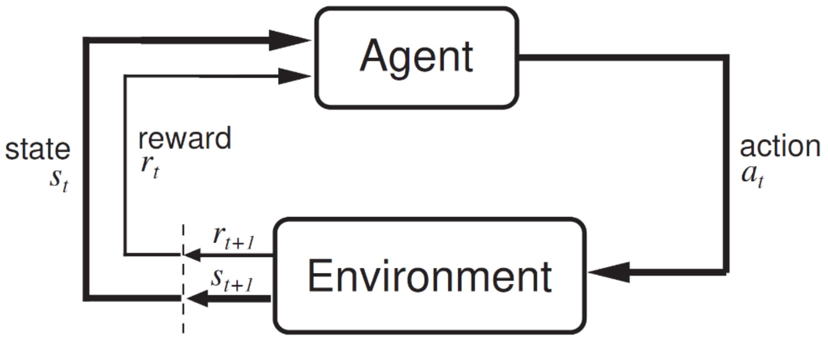


Figure 1 Reinforcement Learning scheme

The generality and flexibility of this approach is obviously one of its most interesting properties, unfortunately this often has a price in terms of sample complexity and learning stability. For these reasons, simulation is key to successfully apply RL to real-world problems.

Combining the flexibility of the Reinforcement Learning paradigm with the empirical power and generality of Deep Neural Networks is certainly appealing and the key to build truly general learning agents. Deep Reinforcement Learning (DRL), in fact, has led to exciting results ([5], [6]) and real leaps in AI research [7].

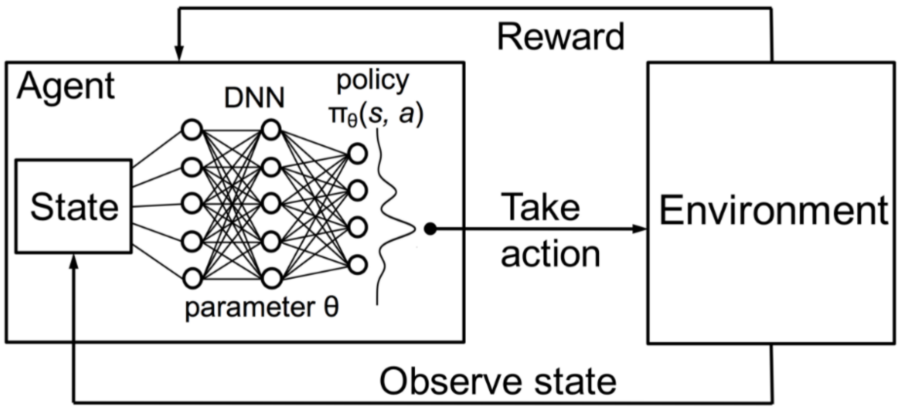


Figure 2 Deep Reinforcement Learning scheme

A possible approach to use neural networks in RL is to parametrize a policy with a Deep Neural

Network, as shown in Figure **2**. DRL agents based on Policy Gradient, such as DDPG [8], A3C [9], TRPO [10], PPO[11], SAC [12] are currently the state-of-the-art for RL applied to continuous control problems, thanks to their empirically high performances and ability to handle continuous action spaces

In particular, PPO (Proximal Policy Optimization) has been proven to be able to reach high performance in complex tasks while being relatively fast to train, simple to tune and not overly complex to implement. PPO has become the "go to" DRL algorithm used by many industry leaders such as OpenAI. PPO does not impose particular limitations on the networks' architecture or environment characteristics and it is easy to customize.

3 ATTITUDE CONTROL MATHEMATICAL MODEL

In order to be able to develop a simulation and training environment, it was necessary to define the mathematical model of the problem. Attitude control is modeled by defining the orientation of a body in space and by the transition from one orientation to another.

There are two main ways to describe the orientation of a rigid body (and therefore the attitude of a satellite):

- **Euler Angles:** any orientation can be achieved by composing three elemental rotations, i.e. rotations about the axes of a coordinate system. Euler angles can be defined by three of these rotations. However, this representation suffers from a problem called gimbal lock.
- **Quaternions**: they are a number system that extends the complex numbers. Quaternions are generally represented in the form (a, b, c, d) : a + b**i** + c**j** + d**k**, where **i**, **j**, **k** are imaginary axes.

Every rotation θ around a vector **v** can be represented by the quaternion:

$$(\cos\left(\frac{\theta}{2}\right), v_x \sin\left(\frac{\theta}{2}\right), v_y \sin\left(\frac{\theta}{2}\right), v_z \sin\left(\frac{\theta}{2}\right))$$

Quaternions are generally preferred over Euler Angles because they do not suffer from gimbal lock and compositions of rotations result to be easier, as they are simple multiplications of quaternions.
In order to rotate a vector **w** of a quaternion rotation **q**, we have to consider it as a quaternion with a null real part:

$$p = (0, w_x, w_y, w_z)$$

Then the rotation can be obtained with the conjugation of **p** by **q**:

$$p' = q * p * q^{-1}$$

Dynamics Equations
We modeled the satellites as rigid bodies whose dynamics is described by Euler equations:

$$\begin{aligned} I_x\,\dot{\omega}_x &+ (I_z - I_y)\omega_y\omega_z = M_x \\ I_y\,\dot{\omega}_z &+ (I_x - I_z)\omega_z\omega_x = M_y \\ I_z\,\dot{\omega}_z &+ (I_y - I_x)\omega_x\omega_y = M_z \end{aligned}$$

We used the Euler method to solve numerically the equations.

The total angular momentum of the system (satellite body + reaction wheels) about the center of mass of the system can be written as:

$$H_{system} = H_{sat} + H_{wheels}$$

And the complete body's dynamics is then described by the following system of equations:

$$\begin{aligned}
I_x\,\dot{\omega}_x + \left(I_z - I_y\right)\omega_y\omega_z + I_{rw}\omega_{rw_z}\omega_y - I_{rw}\omega_{rw_y}\omega_z + I_{rw}\dot{\omega}_{rw_x} &= M_x\\
I_y\,\dot{\omega}_y + (I_x - I_z)\omega_z\omega_x + I_{rw}\omega_{rw_x}\omega_z - I_{rw}\omega_{rw_z}\omega_x + I_{rw}\dot{\omega}_{rw_y} &= M_y\\
I_z\,\dot{\omega}_z + \left(I_y - I_x\right)\omega_x\omega_y + I_{rw}\omega_{rw_y}\omega_x - I_{rw}\omega_{rw_x}\omega_y + I_{rw}\dot{\omega}_{rw_z} &= M_z
\end{aligned}$$

where I_{rw} is the inertial mass of a reaction wheel (same value for all the RWs), ω_{rw} is the angular velocity of the RWs.

4 DEVELOPMENT FLOW

After the definition of the problem, the analysis of possible solutions and the identification of relevant mathematical models, we made several development steps in order to develop the controller, so that we could deal with the different stages of the project.

First of all, we developed a simulation environment (Argotec Learning Environment - ALE) implementing the mathematical system model and it was used to try different implementation solutions to train the agents. At this stage, we used high level programming languages such as Python experimenting quickly state-of-the-art libraries and solution with a minimal effort.

Once the best promising solutions have been identified, we proceeded with a Hardware-in-the-Loop (HIL) validation phase, that exploited a development board representative of the On-Board Computer of ArgoMoon and LICIACube and a Real Dynamic Propagator (RDP) that provided high-reliability simulations of the operational environment and behavior of the ADCS.

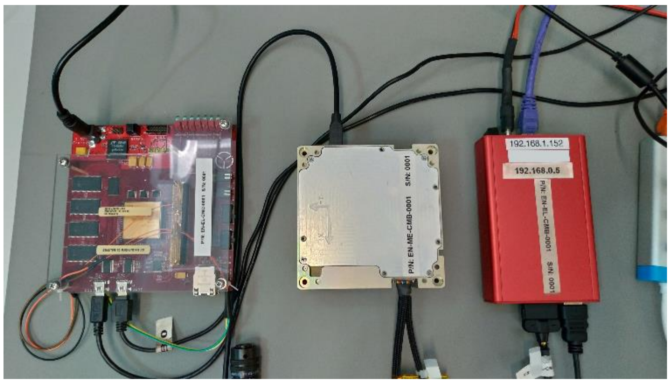


Figure 3 Hardware in the loop setup

Testing the algorithm in the HIL setup required to port the code developed in Python to C, in order to be able to integrate the solution in the software architecture used on board the satellites.

5 CONTROLLER DESCRIPTION

State space

Given the dynamic model of the system, the environment state space, and consequently the agent input, is constituted by the current attitude of the simulated spacecraft, i.e. the orientation quaternion, the angular rates along the three axis and the reaction wheels speed:$s_t =$

$(q_{0_t}, q_{1_t}, q_{2_t}, q_{3_t}, \omega_{x_t}, \omega_{y_t} \omega_{z_t}, ws_{x_t}, ws_{y_t}, ws_{z_t})$

It is also possible to include in the state the information about the working status of the actuators, but being able to access this status is a strong assumption, hardly met in the real world, and should be avoided when training the final model.

In case we want the agent to learn to control the satellite with relevant delays in the actuation, the state should also include the last action, for the environment to be markovian:

$$s'_t = (a_{t-1}, s_t)$$

Action space
The action space is a 3-dimensional vector representing the torques command for the RWs.

$$a_t = (M_x, M_y, M_z)$$

The command is filtered to reflect the physical limitations of the actuators and their working status. It is possible to impose further limitations to the torque being applied using a multiplier.

Reward function
The design of the reward function is a fundamental part of the model since it expresses the objective that the agent has to maximize. We tested various approaches and thoroughly tested their impact measuring performance, tracking their behavior and finally come up with a composite reward structure.

- If the angular speed along any of the 3 axes exceeds a certain threshold the reward is fixed to -1, no matter what the orientation of the satellite is. This ensures that the agent is incentivized to learn how to detumble the satellite.
- In case the angular rate of the satellite is within the threshold the reward is:

$$r = r_{target} - r_{penalty} + r_{bonus}$$

where r_{target} is a reward that is inversely proportional to the distance (in terms of an angle) of the satellite to the target orientation as shown in the figure below.

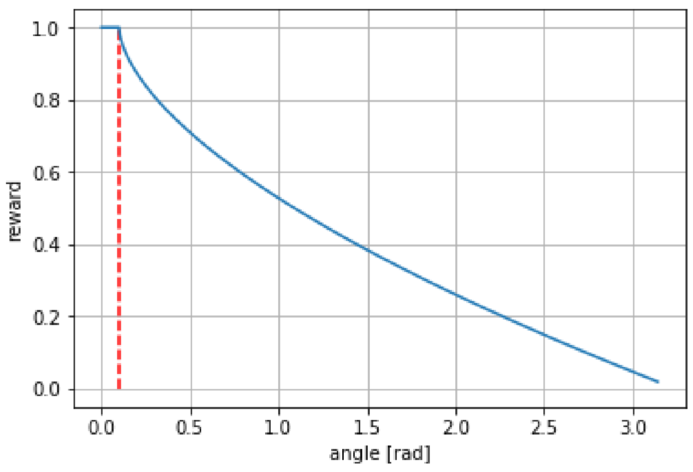


Figure 4 Reward Function

This work aims specifically at solving the problem of underactuated control in the case of the failure of one of the RWs, so we trained 3 specialized controllers for the 3 operative modes of the control system (failure of each RW) and an operative mode with the nominal operation of all the reaction wheels.

It would have been possible to train a single general-purpose controller, even more complex, but, considering the scope of the project, using specialized agents is a good trade-off between computational complexity and performance. Obviously, this approach requires to develop a second component to identify failures and switch to the proper controller. Aiming at building an end-to-end ML solution, we resorted to Supervised Learning to build a failure classifier.

To sum up, the Neural Attitude Control system consists in four controllers each implemented by a neural network activated by a supervisor. The supervisor uses a classifier, fed with a sequence of the last environment observations, that identifies the operative status of the satellite. Each controller has learned to robustly detumble and control the satellite reaching any target attitude.

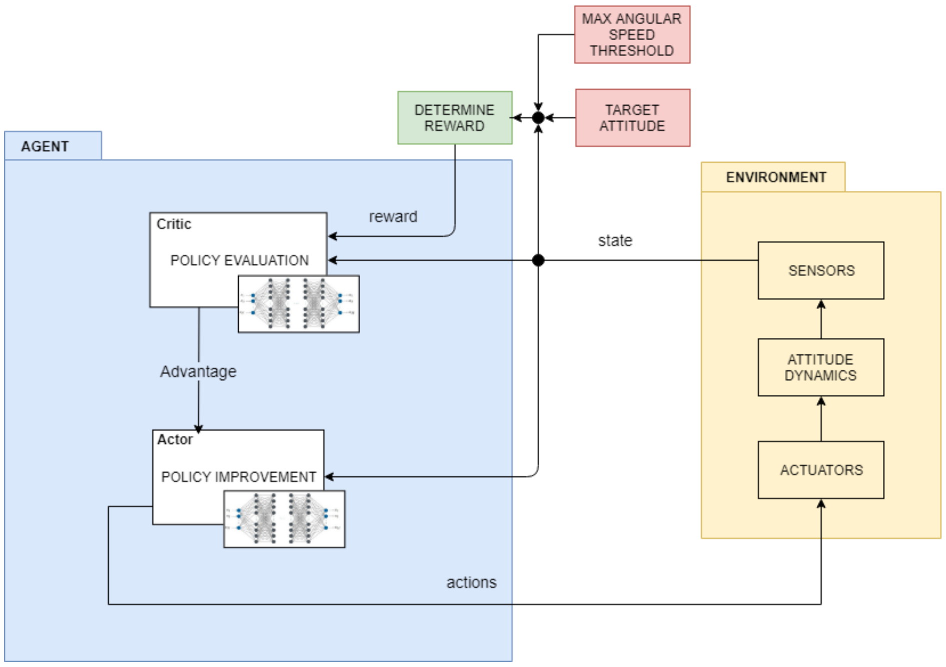


Figure 5.5 Controllers architecture

All the four agents are trained using a custom implementation of PPO. The Neural Networks are standard feed-forward fully connected networks implemented using PyTorch which allows easy optimization with its automatic differentiation tools.

For the nominal case we use a [64, 64] network with ReLU activation, for the underactuated modes we use the same network architectures with the same activation function. We use the same architecture for the actors and the critics and we optimize the networks with Adam [13].

Training

Each agent is trained using a different set of hyper-parameters. In particular, we tested different activation functions and network architectures, several sets of learning rates and batch sizes using training episodes of variable length. We prefer networks with a small number of neurons to keep the computational complexity low.

In order to speed up the data acquisition process and to use more heterogeneous datasets, we trained the agents using 8 parallel actors to gather data from different trajectories. Furthermore, this approach makes the training procedure scale nicely with the number of CPU cores available. At each environment reset a different orientation and speed is sampled as the initial state.

We introduced and tested various modification to the standard implementation of PPO, some of which are well known and documented in the OpenAI implementation, others are inspired by other Policy Gradient algorithms and recent research.

6 RESULTS

We used different methods to assess the performance of the system: the first is an evaluation performed in the simulated environment with over 10.000 different episodes, while the second method evaluates the behavior of the same policies on the target hardware, after that the controllers have been ported and integrated with the real time operating system of the satellite.

Simulation results

To assess the agents performance, we chose challenging maneuvers with starting angles randomly drawn from the range 144-180° about any rotation axis from the initial orientation.

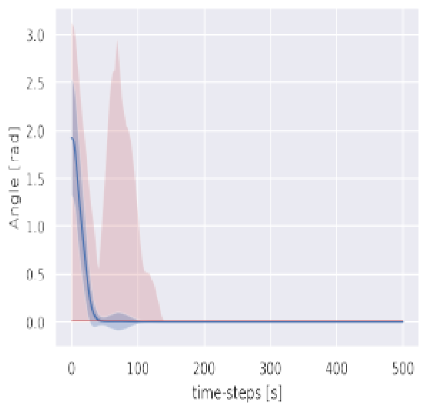
The results are summarized in the following table.

Table 1 Controllers performance

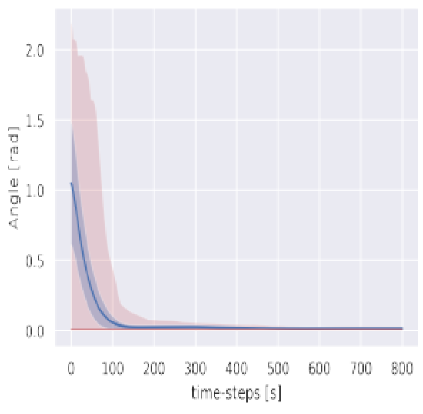
Failure	Align axis	Accuracy [rad]	Conv. time [s]	Horizon [s]
x	x	0.01	72.95	500
	y	0.05	1367.84	800
	z	0.05	987.06	800
y	x	0.05	1218.91	800
	y	0.01	75.37	500
	z	0.05	1185.04	800
z	x	0.05	1463.03	800
	y	0.05	1510.54	800
	z	0.01	142.71	500

The achieved accuracy depends on the failed RW axis and on the required axis alignment. It shall be noted that the performance varies across the underactuated axes because of the different inertias of the satellites along the different axes.

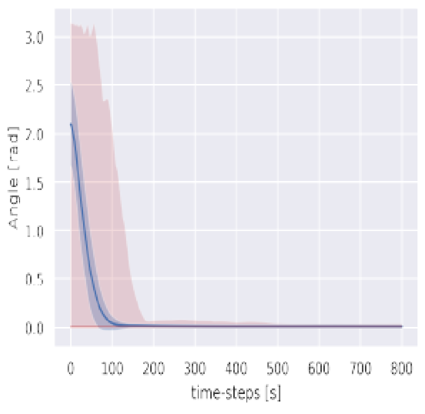
The following image represents the behavior of the controllers in the simulated environment in a way to highlight the average performance (blue line), the variance area (blue area) and the worst controller (red area) by simulating 10.000 episodes for each of the 9 controllers.



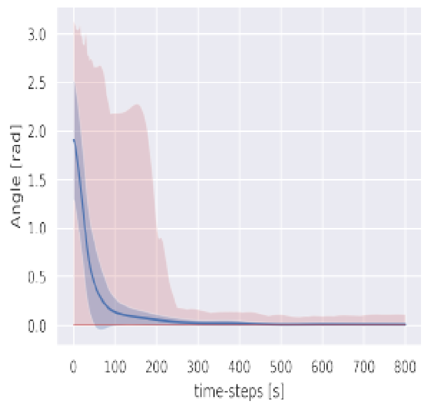
(a) Failure on x-axis, align x



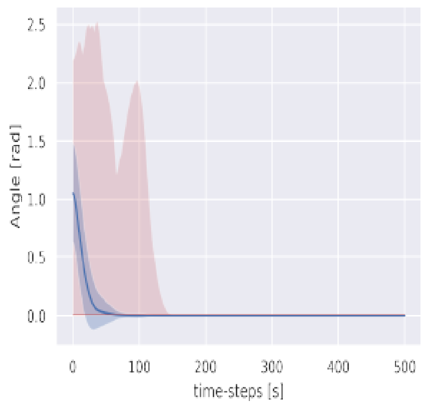
(b) Failure on x-axis, align y



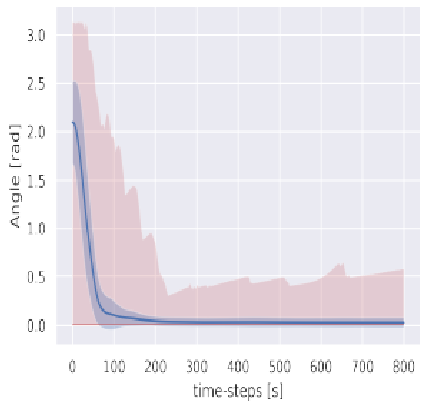
(c) Failure on x-axis, align z



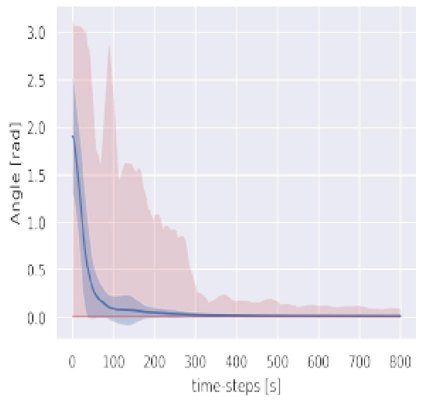
(d) Failure on y-axis, align x



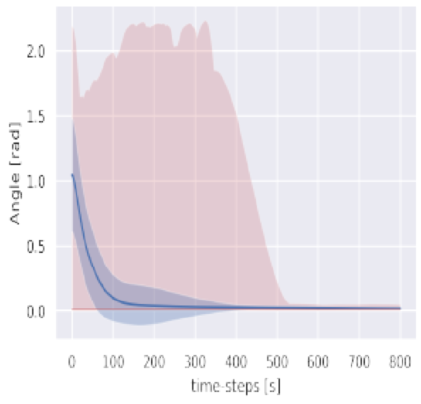
(e) Failure on y-axis, align y



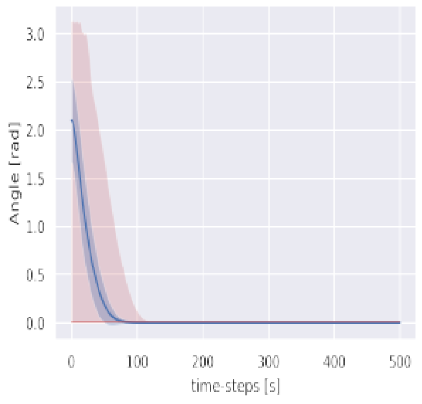
(f) Failure on y-axis, align z



(g) Failure on z-axis, align x

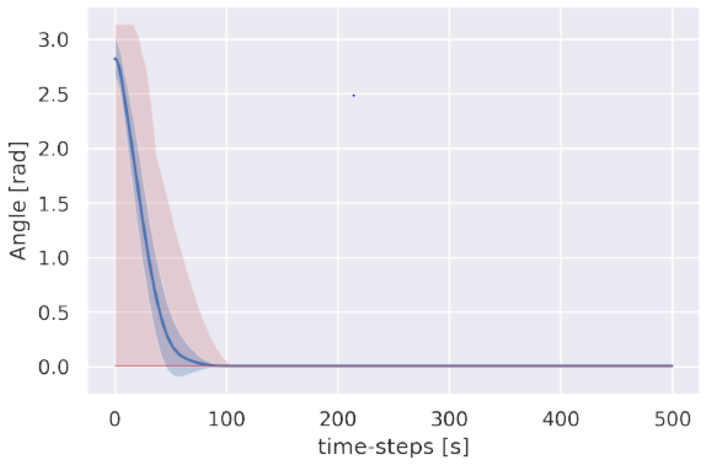


(h) Failure on z-axis, align y



(i) Failure on z-axis, align z

It was performed also an evaluation of the controller in nominal conditions, whose results can be seen in the figure below.



As expected, in this case there’s a much lower variance than in the off-nominal case and even the worst scenario diverges significantly less than the ones in underactuated scenarios.

After the assessment of the performance of the controllers in the simulated environment, we tested the system in the HIL setup to validate the system. Experiments were run in real time conditions and were stopped once either the target attitude accuracy was reached or when other exit conditions were met.

The results obtained in the simulation phase were confirmed, and the system proved to be robust also against non-ideal conditions, such as the non-deterministic actuation time, thanks to the injection of random delays in the training phase.

7 CONCLUSION AND WAY FORWARD

We used Reinforcement Learning to create a flight attitude controller for a small satellite actuated by means of three reaction wheels.

We created a system which has good performance both in the underactuated and nominal scenarios and that is robust to external disturbances and perturbations. Moreover, the controller is implemented in a very compact neural network that is compatible both with the processing power limitations of the satellites and the real-time needs of attitude control.

The behavior of the system has been validated by means of tests on a representative hardware in a HIL setup.

The controller has been uploaded on ArgoMoon and we are planning to execute on orbit tests in the second phase of the mission, that is scheduled to take place in summer 2022.

During the in-orbit tests, we will stop one of the reactions wheels and make the satellite reach a pre-defined target attitude. We plan to prepare different conditions and scenarios to gather information verifying the correct functioning of the controller in a real-world environment and to validate the simulations we made.

8 REFERENCES

[1] Valerio Di Tana, Gabriele Mascetti, Simone Pirrotta, Carlo Fiori, Riccardo Rinaldi, Simone Simonetti, Biagio Cotugno, "ArgoMoon: Challenges And Design Solutions For The Development Of A Deep Space Small Satellite," Proceedings of the 69th IAC (International Astronautical Congress) Bremen, Germany, 1-5 October 2018, paper: IAC-18.B4.8.1, URL: https://iafastro.directory/iac/proceedings
[2] P. Tortora and V. Di Tana, "LICIACube, the Italian Witness of DART Impact on Didymos," 2019 IEEE 5th International Workshop on Metrology for AeroSpace (MetroAeroSpace), 2019, pp. 314-317, doi: 10.1109/MetroAeroSpace.2019.8869672.
[3] P. Tsiotras and V. Doumtchenko. Control of spacecraft subject to actuator failures: state-of-the-art and open problems. J. Guid. Control Dyn., 7, 2000.
[4] H. Gui, L. Jin, and S. Xu. Attitude maneuver control of a two-wheeled spacecraft with bounded wheel speeds. Acta Astronaut., 88:98–107, July 2013.
[5] V. Mnih et al., «Human-level control through deep reinforcement learning», Nature, vol. 518, n. 7540, pagg. 529–533, feb. 2015
[6] N. Heess et al., «Emergence of Locomotion Behaviours in Rich Environments», ArXiv170702286 Cs, lug. 2017.
[7] D. Silver et al., «Mastering the game of Go without human knowledge», Nature, vol. 550, n. 7676, pag. 354–359, ott. 2017.
[8] T. P. Lillicrap et al., «Continuous control with deep reinforcement learning», ArXiv150902971 Cs Stat, set. 2015.
[9] V. Mnih et al., «Asynchronous Methods for Deep Reinforcement Learning», ArXiv160201783 Cs, feb. 2016.
[10] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, e P. Abbeel, «Trust Region Policy Optimization», ArXiv150205477 Cs, feb. 2015.Fdfdsdfs
[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, e O. Klimov, «Proximal Policy Optimization Algorithms», ArXiv170706347 Cs, lug. 2017.Fdfdsfds
[12] T. Haarnoja, A. Zhou, P. Abbeel, e S. Levine, «Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor», ArXiv180101290 Cs Stat, gen. 2018.Dfsfdfd
[13] D. P. Kingma e J. Ba, «Adam: A Method for Stochastic Optimization», ArXiv14126980 Cs, dic. 2014.