33RD CONGRESS
OF THE INTERNATIONAL COUNCIL
OF THE AERONAUTICAL SCIENCES
STOCKHOLM, SWEDEN, 4-9 SEPTEMBER, 2022

ICAS 2022 SWEDEN

DEEP REINFORCEMENT LEARNING FOR EVTOL HOVERING CONTROL

Danilo Sartori Alarcon[1] & Jorge Henrique Bidinotto[1]

[1]University of São Paulo

Abstract

The objective of this paper is to use Deep Reinforcement Learning (DRL) to control the vertical speed and pitch angle of an eVTOL during hovering flight. A simple 6DoF model of the Airbus Vahana is implemented in Open AI Gym workspace and the PPO tested, with different hyperparameters evaluation. Then, the PPO method is also compared with DDPG and SAC methods, using Open AI SpinningUp implementation, and also compared with a PID controller baseline. The result showed that the PPO was the best algorithm for this task, both in execution time and control performance, with returns slightly better than a conventional PID controller

Keywords: deep reinforcement learning; eVTOL; UAM

1. Introduction

In recent years the electrical Vertical Takeoff and Landing (eVTOL) market for Urban Air Mobility (UAM) is evolving in fast steps, gaining a big value among different companies. Although this market relies only in future estimations, the years of 2021 and 2022 delivered a reality: companies were listed in the stock exchange markets worthing billions of dollars each (Table 1). Although some devaluation in the stock prices has occurred since the initial offer, the market yet look to the companies with a high expectation.

Table 1 – eVTOL manufacturers initial market value

COMPANY	INITIAL MARKET VALUE [US$ billions]	MONTH OF INITIAL STOCK TRADE
Archer	1.7	Sep, 2021
Eve	2.4	May, 2022
Joby	6.6	Aug, 2021
Lilium	3.3	Sep, 2021
Vertical	5.4	Dec, 2021

Sources: [1], [2], [3], [4], [5]

Numerous aircraft manufactures plan to deliver their aircraft for commercial use by around 2025. Among the diverse list of challenges to be surpassed, one of them is the aircraft control, specially for aircrafts with tilted wings or rotors. The high nonlinearities involved in the propeller/wing interference and the multiple inputs for control (rotor tilt and rotation) may impose big challenges for the aircraft reliable control in diverse situations. Azar et al [6] and Bøhn [7] in their work with UAV cite that these nonlinearities are a big difficult for classical control designs.
In order to deal with these problems, a possible alternative is the use of Deep Reinforcement Learning (DRL) methods to create a robust control that can deal with the nonlinearities and optimize the control with multiple inputs. DRL methods are a recent evolution as well as the eVTOL, dating mainly in the

2010's, and their application up to now are highly concentrated in robots control and, in the case of air vehicles, concentrated in path planning, navigation and trajectory optimization.
However, a few works applying DRL for attitude and speed control of small unmanned air vehicles (UAV) are available, such as the work made by Koch [8], which implemented a quadcopter model to test DRL methods, such as Proximal Policy Optimization (PPO), Trust Region Policy Optimization (TRPO) and Deep Deterministic Policy Gradient (DDPG). This study indicated that, for the attitude control problem, the PPO method delivered the best results.
Other studies such as Xu [9] and Bøhn [7] used PPO to control the attitudes and speed of fixed wing aircrafts, and the review done by Azar et al [6] indicated that PPO is the more indicated method for attitude and speed control.
These initial work were a good starting point for the DRL application for aircraft controlling. But further questions shall be asked or the future: are other methods applicable to the controllers? Are the methods applicable for vertical-horizontal flight transition? How is the robustness for real life application in manned air vehicles, such as eVTOLs ?
The work in this paper aims to start to answer these questions. Here the Airbus A³ Vahana is modeled (Section 3.) and the PPO algorithm is tested with Neural Network (NN) and hyperparameters variation, in order to find the best cases (Section 4.3) and compare the results with a PID baseline. Finally, the PPO method is compared with Deep Deterministic Policy Gradient (DDPG) and Soft Actor-Critic (SAC) in Section 4.4

2. Deep Reinforcement Learning Methods

Reinforcement Learning methods rely on the Markov Decision Process (MDP) as expressed in Figure 1. The iterative process define the relation between the states, agent, action, environment and reward. Based on the states in time-step t, the agent takes the action A_t according to a given policy π. This action interacts with the environment, resulting in the next state S_{t+1} and a reward signal R_{t+1}. The reward measures 'how good' the state is, based on a pre-defined reward equation.

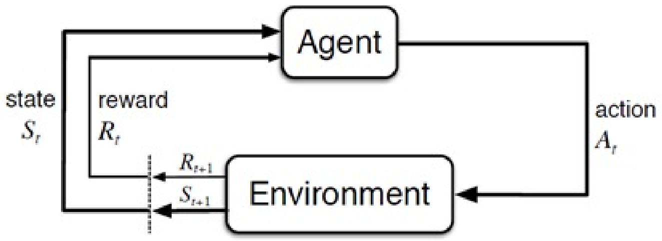


Figure 1 – Markov Decision Process (MDP) schematic [10]

The agent objective is to generate a policy that takes good actions in order to improve the rewards. However, a new reward is obtained for every time-step and it is not feasible to optimize a big set of values. Instead, the accumulated rewards are used in the optimization, called Return (G). In general, the return in time-step t is the accumulated reward for future time-steps, considering the discount factor γ, as calculated in Equation 1. In general, the return for the first time-step (G_0) is used as the optimization objective.

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma \cdot R_{t+2} + \gamma^2 \cdot R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k \cdot R_{t+k+1} \\ G_t &\doteq R_{t+1} + \gamma \cdot G_{t+1} \end{aligned} \tag{1}$$

The policy is a parameterized function that calculates actions based on current states. The Reinforcement Learning main goal is to optimize a policy, by changing its parameters to increase a given return. The policy function may be of different types, but in the 2010's, Neural Networks (NN) were used as policy approximation functions, beginning the era of Deep Reinforcement Learning (DRL) methods for controlling process.

In this study, three DRL methods will be used for policy optimization: Proximal Policy Optimization (PPO) [11], Deep Deterministic Policy Gradient (DDPG) [12] and Soft Actor-Critic (SAC) [13]. The algorithms used are the one implemented at the OpenAI SpinningUp framework [14]

3. Aircraft Model

The model used for the experiments was based on the Airbus A³ Vahana aircraft (Figure 2). This aircraft was chosen due to two reasons: the first is because it has a tilt-wing and tilt-rotor configuration, what will be used for vertical to horizontal flight transition in future work (and evaluate DRL in this scenario) and the second reason is that it uses propellers and rectangular wings, which are easier to be modeled when comparing to complex planforms or ducted fans instead of propellers.

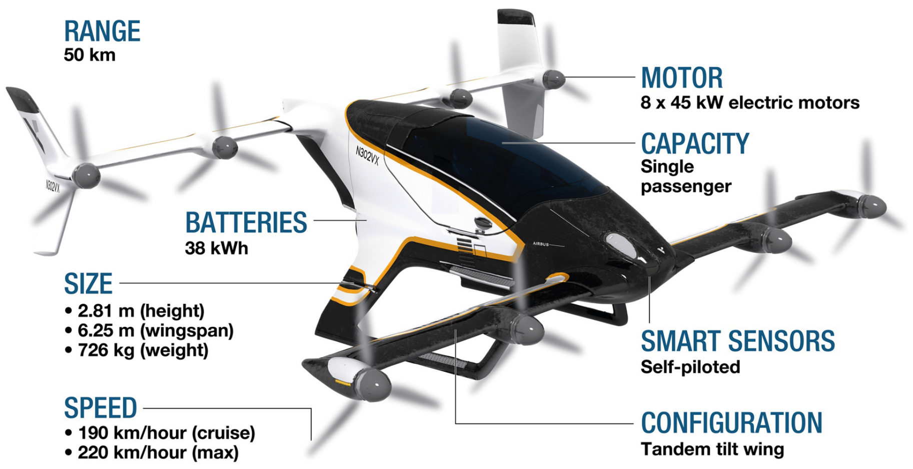


Figure 2 – Vahana general technical specifications [15]

The aircraft was modeled in the Open AI Gym [16] framework. A 6-DoF model was used, following the flowchart logic of Figure 3.
The aerodynamic block calculates the six aerodynamic coefficients in stability axis (the three forces: CLS, CDS, CYS; and three moments: CRS, CMS, CNS - term 'S' means stability axis) of the wing and fuselage, convert them to Body Axis and then calculate the aerodynamic body forces and moments, considering surface and CG positions.

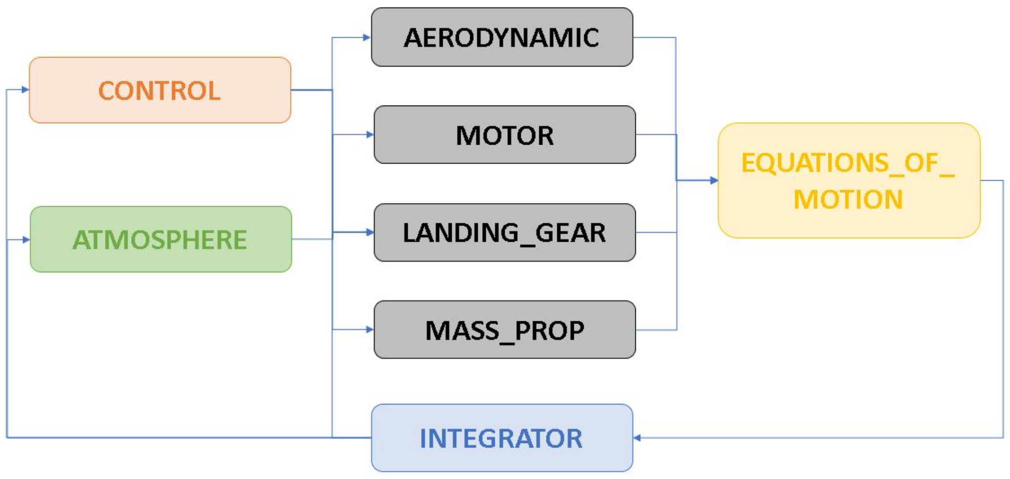


Figure 3 – Aircraft model flowchart

As the wing is subjected to high angles of attack, up to 180°, at this moment a simple flat plate model was used, following the proposed by Xu et al. [9], but modified to consider sideslip angle:

$$\begin{aligned}
CDS_{wing} &= 2 \cdot \sin^2 \alpha_{wing} \cdot |\cos \beta_{wing}| \\
CLS_{wing} &= 2 \cdot \sin \alpha_{wing} \cdot \cos \alpha_{wing} \cdot |\cos \beta_{wing}| \\
\\
FXS_{wing} &= -CDS_{wing} \cdot S_{wing} \cdot q \\
FZS_{wing} &= -CLS_{wing} \cdot S_{wing} \cdot q
\end{aligned} \tag{2}$$

Where:

S_{wing}	wing reference area [m²]
q	Dynamic Pressure [Pa]
FXS/FZS	Force at X and Z axis, Stability reference frame [N]

On the other hand, the fuselage has a drag and lateral force model modeled as:

$$\begin{aligned}
CDS_{fus} &= (0.10 \cdot \cos \beta_{fus} + 0.40 \cdot \sin \beta_{fus}) \cos \beta_{fus} \\
CYS_{fus} &= -(0.10 \cdot \cos \beta_{fus} + 0.40 \cdot \sin \beta_{fus}) \sin \beta_{fus} \\
\\
FXS_{fus} &= -CDS_{fus} \cdot S_{fus} \cdot q \\
FYS_{fus} &= CYS_{fus} \cdot S_{fus} \cdot q
\end{aligned} \tag{3}$$

Where:

S_{fus}	fuselage reference area, being calculated as cross section area: πr^2 [m²]
q	Dynamic Pressure [Pa]
FXS/FYS	Force at X and Y axis, Stability reference frame [N]

The motor block calculates the propulsive forces, the Landing Gear the ground reaction forces (not used in the hover simulations of this study) and the Mass Properties block calculates the current mass, center of gravity and inertia tensor. Then, the Equations of Motion block calculates the states derivatives following the equations:

$$\begin{aligned}
X\dot{Y}Z_E &= LB2E \cdot V_B \\
E\dot{U}L_B &= \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)/\cos(\theta) & \cos(\phi)/\cos(\theta) \end{bmatrix} \cdot \omega_B \\
\dot{V}_B &= \frac{FT_{BA} + W_{BA}}{m} - \omega_B \times V_B \\
\dot{\omega}_{BODY} &= I^{-1} \cdot [MT_{BODY} - \omega_{BODY} \times (I \cdot \omega_{BODY})]
\end{aligned} \tag{4}$$

Where:

XYZ_E	Vector of Body position in Earth axis
V_B	Vector of Body linear speeds in body axis
EUL_B	Vector of body Euler angles (ϕ, θ, ψ)
ω_B	Vector of body rotational speeds (p, q, r)
FT_{BA}	Total External Force, Body Axis at CG
MT_{BA}	Total External Moment, Body Axis at CG
m	Aircraft Mass, in [kg]

g	9.806 m/s. Gravitational Acceleration
I	Body Inertia Matrix tensor, in [kg.m²]
W_{BA}	Weight force in body axis ($LE2B \cdot \begin{bmatrix} 0 & 0 & mg \end{bmatrix}^\top$)
$LE2B$	Rotation Matrix, from Earth Axis to Body Axis
$LB2E$	Rotation Matrix, from Body Axis to Earth Axis

The integrator model used a second-order Euler method with 0.05 second of time step and the Atmosphere block uses the ISA Model [17].
The control block receive as inputs the normalized (from -1 to +1) pitch (u_{pitch}) and vertical commands ($u_{vertical}$), that will be the Reinforcement Learning agent actions (in this study, roll and yaw are fixed in the Equations of Motion block). Then, the control block firstly allocate these commands u to thrust commands of each one of the engines:

$$\begin{aligned} Control_{Thrust} = u_{pitch} \cdot \left[+1,+1,+1,+1,-1,-1,-1,-1\right] \\ +(u_{vertical}+1)/2 \cdot \left[+1,+1,+1,+1,+1,+1,+1,+1\right] \end{aligned} \tag{5}$$

Then, this command shall be saturated from -1 to +1:

$$SatControl_{Thrust} = \min\left(N_{i=1:8} = 1, \max\left(N_{i=1:8} = -1, Control_{Thrust},\right)\right) \tag{6}$$

The command is then re-scaled from 0 to 1, as there is no negative rotation considered in the motors:

$$ScaleControl_{Thrust} = 0.5(SatControl_{Thrust} + 1) \tag{7}$$

And finally, in order to consider the quadratic proportionality between RPM and Thrust, the RPM_p control is calculates as:

$$RPM_p = \sqrt{ScaleControl_{Thrust}} \tag{8}$$

This command (RPM_p) will then be an input of the motor function.

4. Tests and Results - Vertical Speed and Pitch Control

4.1 Observations, Reward, Action and Terminal State

The first part of the work was to control the vertical body speed and pitch angle. The reinforcement learning agent used the body vertical speed and pitch angle as observed states, and their derivative and integral values. These states are normalized according to Table 2. The use of the integral values as observations to improve the control was based on the work of Xu et al. [9].

Table 2 – Vertcal Speed / Pitch Angle Control Model Observation Normalization Factor

STATE	NORM FACTOR
w_{BODY}	$1/20$
$w_{B\dot{O}DY}$	$1/20$
$\int w_{BODY}$	$1/100$
θ	$1/0.5\pi$
q	$1/0.5\pi$
$\int \theta$	$1/2\pi$

The reward for each state used the quadratic error of the normalized vertical speed and pitch, following equation 9. In this case, the highest possible reward is 20 and the lowest is 0. The normalized error was motivated by the work of Azar et al. [6] and Koch [8], that indicated that a normalized reward helps to speed up the training.

$$R_t = 10 \cdot \left(1 - \sqrt{\min\left(1, \overline{w_{BODY}}^2\right) + \min\left(1, \overline{\dot{w}_{BODY}}^2\right)}\right) + 10 \cdot \left(1 - \sqrt{\min\left(1, \overline{\theta_{deg}}^2\right) + \min\left(1, \overline{\dot{\theta}_{deg}}^2\right)}\right) \tag{9}$$

Where:

$$\overline{w_{BODY}} = \frac{w_{BODY}}{10} \qquad \overline{\theta_{deg}} = \frac{\theta_{deg}}{45}$$
$$\overline{\dot{w}_{BODY}} = \frac{\dot{w}_{BODY}}{10} \qquad \overline{\dot{\theta}_{deg}} = \frac{\dot{\theta}_{deg}}{45} \tag{10}$$

Another strategy applied was the terminal state, when reaching a pitch angles greater than $\pm$ 45 °. This was useful in order to stabilize the training as, when exploring the environment, the aircraft could reach high attitude angles that could lead to loss of control. When reaching this terminal state, the episode is stopped and no additional reward is accumulated. Hence, episodes ending in a terminal state will lead to a lower return.
When using a terminal state it is important to avoid a condition of local minimum. In the case of negative rewards (i.e. a penalty when far from the target values), when reaching a terminal state the negative rewards are not accumulated anymore. This way, an agent that reaches the terminal state in a few steps (1 second, for example) would have a better return than an agent that reaches this terminal state in more time (2 seconds, for example). However, it is clear that an agent that takes more time to reach the terminal state is better, and will continue to improve. So, when using negative rewards a terminal state penalty shall be introduced, proportional to the time it takes to reach the state. A simple solution found was the to use positive rewards and no additional penalty when reaching the terminal state.

4.2 PID Baseline

The first step was to determine a PID control baseline, using the gains of Table 3. Although a complete optimization was not done to maximize the return with the PID, a manual trial and error process was executed and these gains were considered close to optimum for the PID controller.

Table 3 – Gains for the PID controller baseline

	u_{vertical}	u_{pitch)
K_P	2.1	1.0
K_I	0.0015	0.0
K_D	0.0	0.5

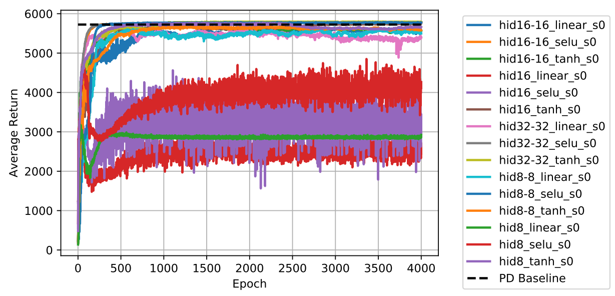
The return for the PID controller is expressed in Table 4. The simulation was of 15 seconds (300 steps), initiating at $\pm$ 10 m/s and $\pm$ 10°.

Table 4 – PID controller return

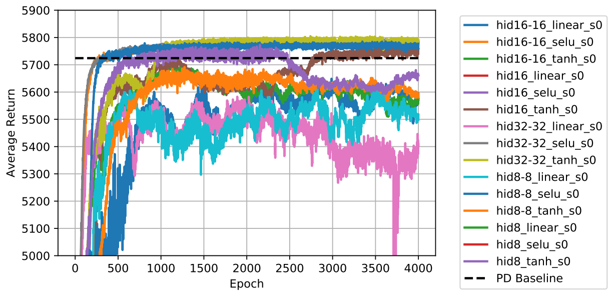
Initial w_{BODY}	Initial θ	Return
+10 m/s	+10 deg	5709.6
+10 m/s	-10 deg	5709.6
-10 m/s	+10 deg	5739.0
-10 m/s	-10 deg	5739.0
AVERAGE		**5724.3**

4.3 PPO Method Results

The PPO method implementation used the OpenAI SpinningUp algorithms [14]. Each episode started at an aleatory condition in the four combinations of vertical speed of $\pm$ 10 m/s and pitch angle of $\pm$ 10°. The goal is to stabilize both parameters at zero. The episode length was of 300 steps (15 seconds), except when the terminal state is achieved.
The first hyperparameter tested was the variation of the Neural Network architecture (perceptrons and activation functions). Architectures with 8, 16, 8-8, 16-16 and 32-32 perceptrons were used, with both linear, tanh and selu activation. Figure 4 shows the average return for this test.



(a)



(b) Y-axis zoom

Figure 4 – PPO results - grid test for Neural Network architecture

In this case study, smaller NN (such as 1 layer with 8 or 16 perceptrons) tend to have worst result and be dependant on the activation function to avoid local minimum. The cases with two layers (8-8, 16-16, 32-32) have good results, similar to PID baseline, with the three activation functions tested (tanh, selu, linear). However, the SeLu activation represented the most stable return evolution, with similar results for the three cases with two layers, as seen in Figure 5.

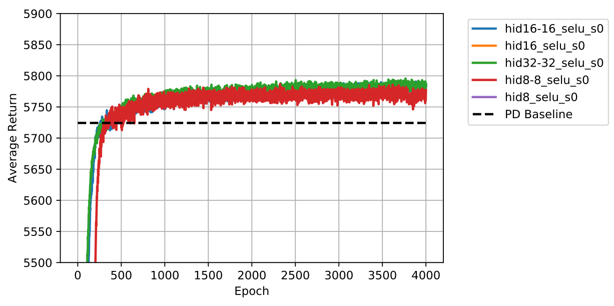


Figure 5 – PPO results - Selu activation

As the training resulted in good reward, higher than the PID baseline, other PPO hyperparameters were also analyzed. The NN architecture was fixed with two layers of 16 elements (16-16) with the SeLu activation, which was the best configuration with the default hyperparameters.
The first parameter studied was the Clip Ration (Figure 6a). The variation of 0.1 and 0.3 around the default value of 0.2 did not resulted in better return after 4000 epochs, just a faster learning in the first 500 epochs. On the other hand, when varying the Target KL-divergence (Figure 6b), the increase to 0.02 resulted in a fast learning in the beginning, but lower final return, and the opposite for reducing it.
Finally, the Lambda (for General Advantage Estimation, Schulman et. al. [18]) is varied (Figure 6c) and higher values (0.99 between the tested ones) resulted in better performance, both for faster learning and final return.
Fixing the Clip Ratio at 0.2, Target KL at 0.01 and Lambda at 0.99, the PPO was applied in 15.000 epochs, with the return history at Figure 6d.

Simulating the final agent of Figure 6d the result is shown in 7. The simulation shows that the agent is capable of stabilize the controlled variables related to the reward (w_{BODY},$\dot{w}_{BODY}$,θ,q), starting from

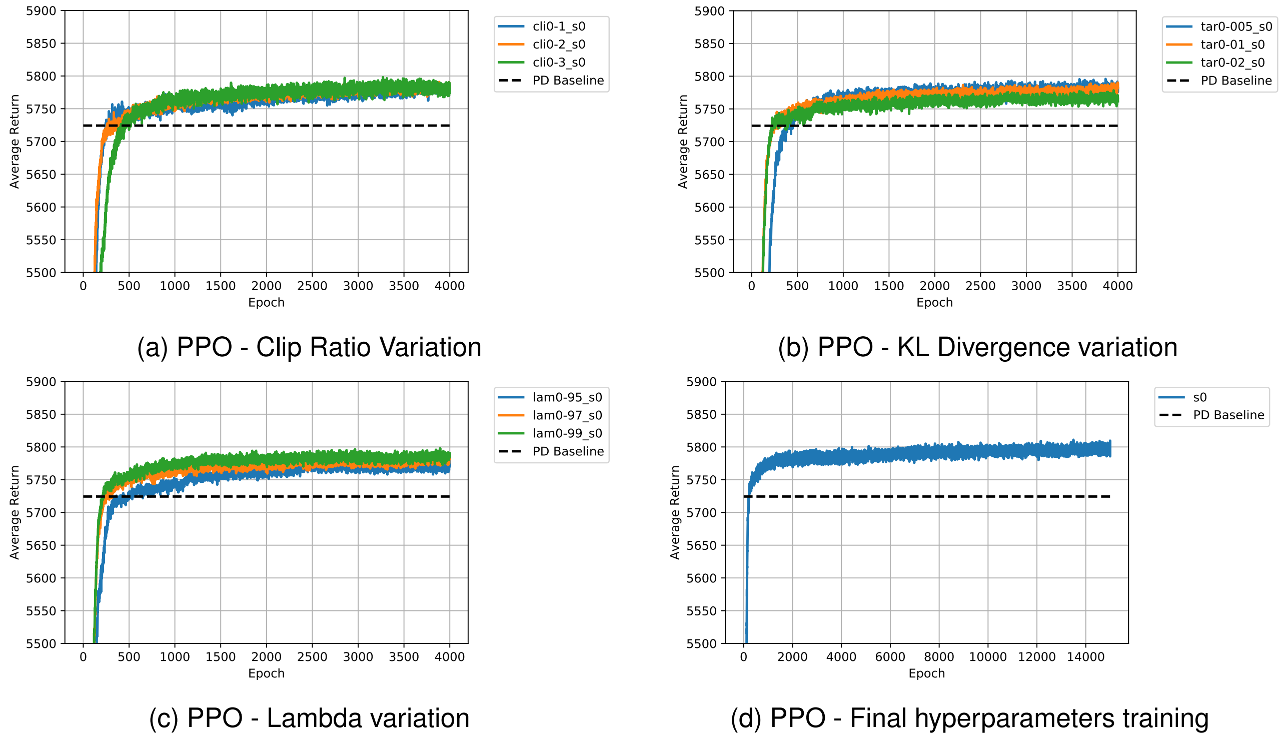


Figure 6 – PPO results - Hyperparameter variation

a vertical speed of -10 m/s and pitch angle of 10 deg. It can also be observed that the longitudinal body speed (U) is stabilized at a value different from zero, as the controller does not aim to pursuit this error. For future works, the Pitch Angle controller can be used as a inner loop controller for the longitudinal speed controller, or even train the agent to pursuit a target longitudinal speed, instead of Pitch Angle. It is important to note that Pitch angle and Longitudinal Speed are dependant on each other, hence it is better to control just one of them, as the controller may not be possible to zero both, as their relation varies with wind speed, propeller inclination and other factors.

Although the PPO generates a stochastic policy, the Figure 7 shows no apparent stochasticity in the body actions. This is due to the stochasticity reduction along the training epochs (the agent reduces exploration and increases exploitation). The Figure 6d shows a rapid increase in the average return in the first 1000 epochs, and then a slight increase until the end of training. This continuous increase is due to the stochasticity decrease. This can be confirmed by the history of the simulations of the agents for different epochs in Figure 8, where the pitch rate (q) is plotted for different training epochs. Although right in the 200th epoch the system already show a "stabilization" around zero, the commands are still very noisy. However this noise is reduced along the epochs and after thousands of iterations, the policy is almost deterministic.
Another way to look at the stochasticity is to plot the action versus the observation parameters. Note that, in the first epoch of Figure 9, the policy is almost a random number between -1 and +1. In the 200th epoch there is already a trend in the behavior, but some points may be fuzzy colored. From epoch 500 and on, the stochasticity is almost imperceptible looking to these chart.
Finally, we can check the actions response versus other observations parameters in Figures 10 and 11. The first one shows that the Pitch Command acts almost like a PID, being dependent mainly on θ and q, with a minor dependency on $\int \theta$, but with a non-linearity around zero. On the other hand, the Pitch Command almost has no dependency on the vertical speed and acceleration.
However, the Vertical Command depicted in Figure 11 is a little more complex. Relative to the basic parameters of vertical speed and acceleration, it also has a PD shape. But it has also a big dependency on the pitch angle and acceleration, in order to keep good stabilization even when the aircraft is leveled.

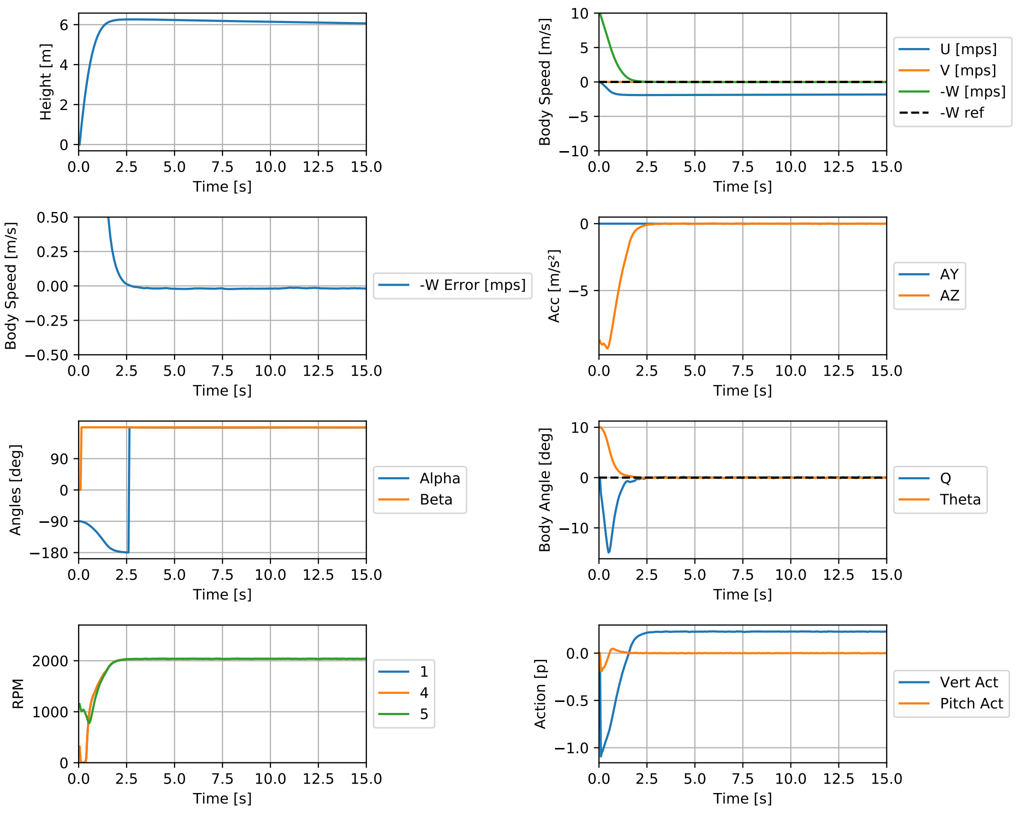
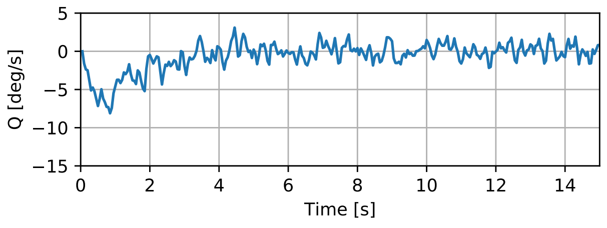
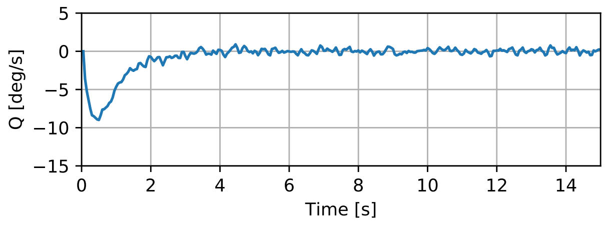


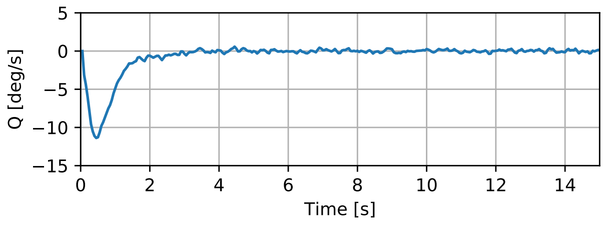
Figure 7 – Aircraft model simulation, with best agent of Figure 6d



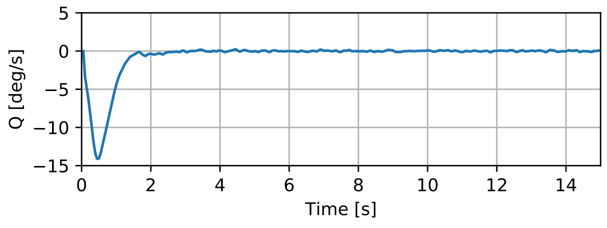
(a) after 200 epochs



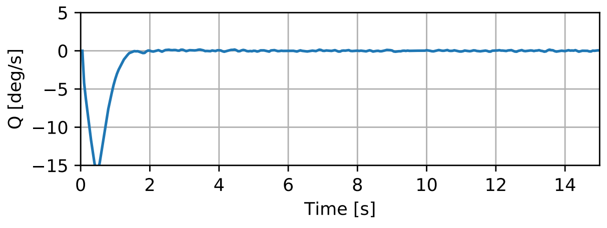
(b) after 500 epochs



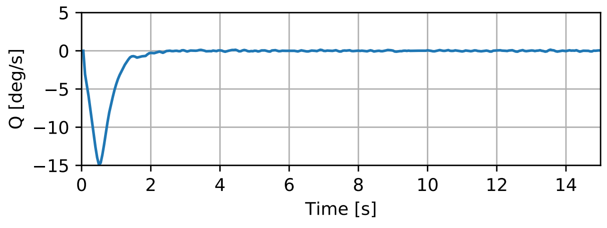
(c) after 1000 epochs



(d) after 5000 epochs



(e) after 10000 epochs



(f) after 15000 epochs

Figure 8 – Aircraft Model Simulation of Vertical Speed (w_{BODY}) and Pitch Angle (θ) control, with agents of different epochs of Figure 6d

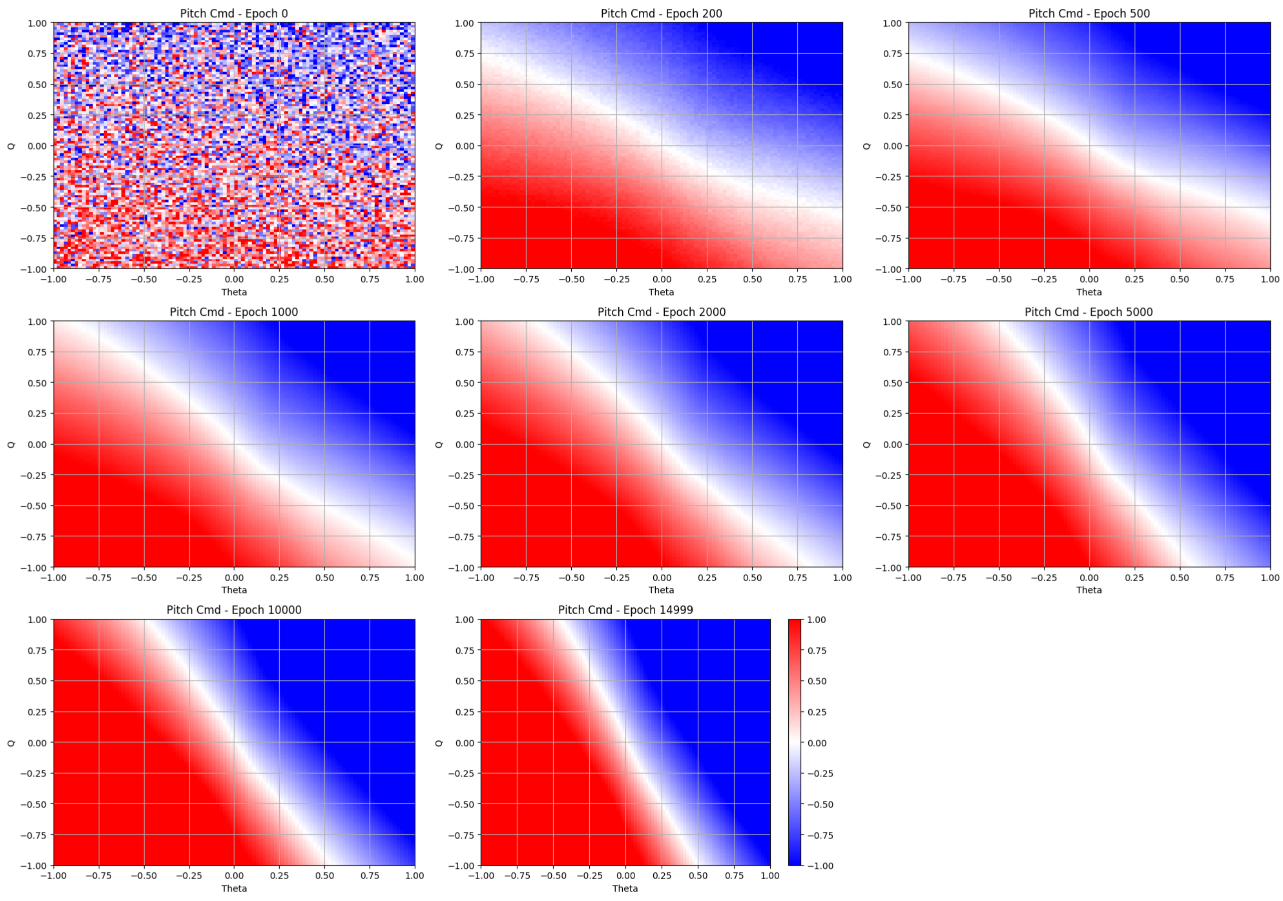


Figure 9 – Controller Pitch Command in function of θ and q for different epochs of training of Figure 6d

4.4 DRL Method Comparison

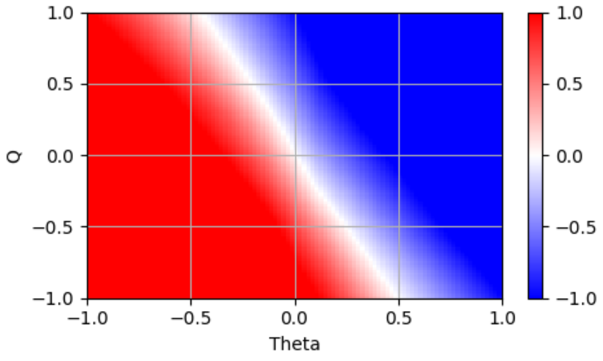
After studying different parameters for optimizing the PPO results, the next step was to test different SpinningUp algorithms and compare them. In order to do this, the DDPG and SAC methods were tested and their hyperparameters optimized in order to result in the best average return. The Table 5 describe the tested configurations and, in bold, the best configuration of each hyperparameter. Other parameters were used as the SpinningUp default and the reward used was the same as described in section 4.1

Table 5 – Parameters tested for DDPG and SAC optimization

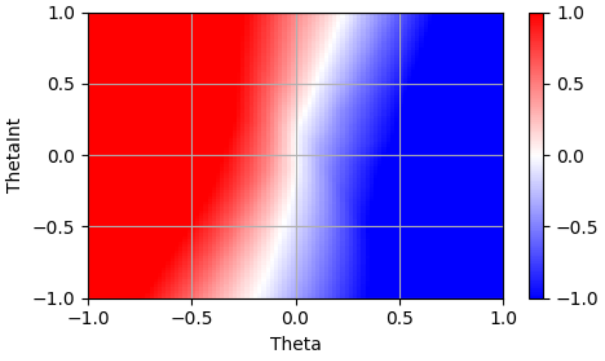
DDPG		**SAC**	
Neural Network Architecture	8 ; 16 ; 32 ; 8-8 ; 16-16 ; 32-32 ; **64-64**	Neural Network Architecture	8 ; 16 ; 32 ; 8-8 ; 16-16 ; 32-32 ; **64-64**
Activation Function	tanh ; **selu**	Activation Function	tanh ; **selu**
Gamma	0.85 ; **0.90** ; 0.95 ; 0.99	Gamma	0.85 ; **0.90** ; 0.95 ; 0.99
Steps per Epoch	**3600**	Steps per Epoch	**3600**
Polyak	0.990 ; 0.995 ; **0.999**	Polyak	0.990 ; **0.995** ; 0.999
Action Noise	0.05 ; **0.10** ; 0.20	Alpha	0.1 ; 0.2 ; **0.3**

Note: in bold, the best (with highest average return) combination tested

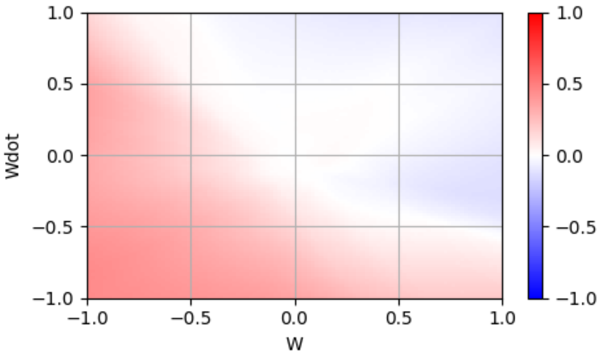
After determining the best hyperparameters, the best configuration for each algorithm was tested for 1000 epochs and compared with the previously optimized PPO. The Figure 12a shows the Return evolution in function of the epochs and the Figure 12b shows the Return in function of Execution Time. The PPO algorithm is the only one capable of reaching values close to the PID baseline and



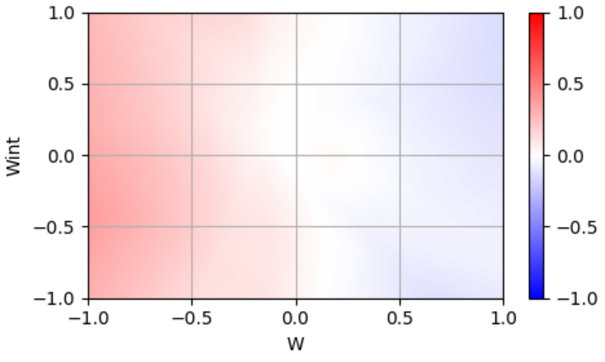
(a) Pitch Command x θ x q



(b) Pitch Command x θ x $\int \theta$

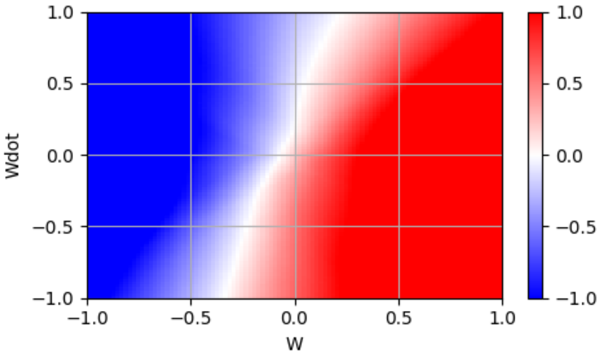


(c) Pitch Command x w_{body} x $\dot{w}_{body}$

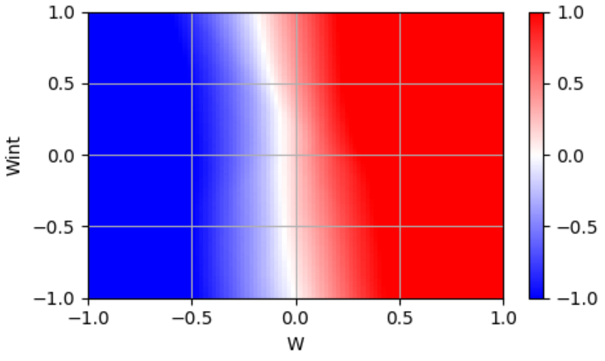


(d) Pitch Command x w_{body} x $\int w_{body}$

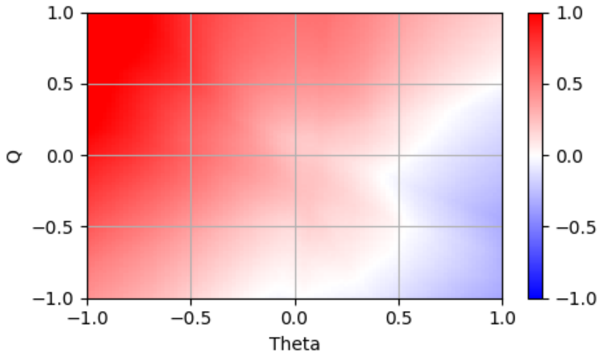
Figure 10 – Controller Pitch Command in function of different observations. Best agent of Figure 6d



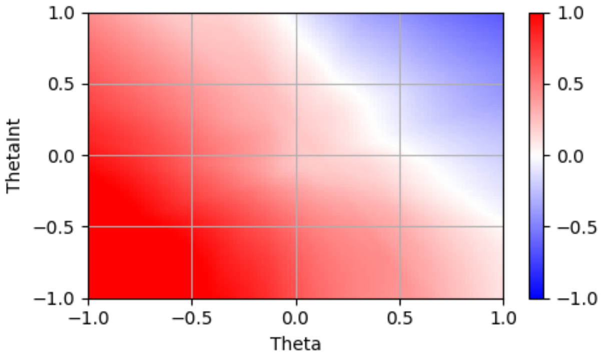
(a) Vertical Command x w_{body} x $\dot{w}_{body}$



(b) Vertical Command x w_{body} x $\int w_{body}$



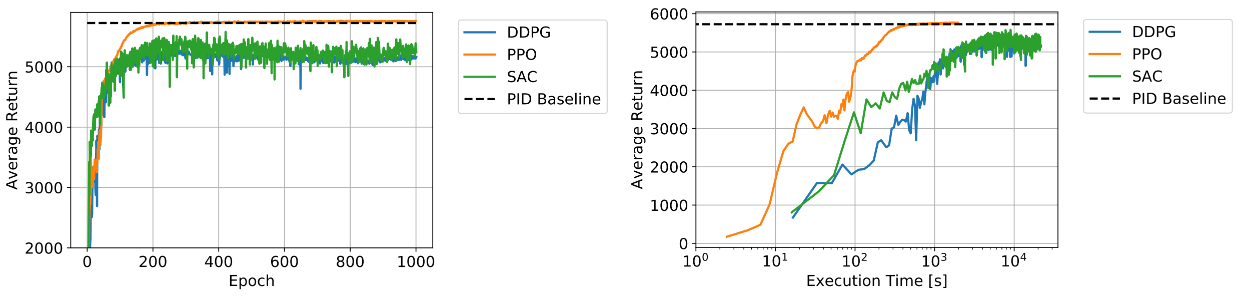
(c) Vertical Command x θ x q



(d) Vertical Command x θ x $\int \theta$

Figure 11 – Controller Vertical Command in function of different observations. Best agent of Figure 6d

other methods (SAC and DDPG) were not capable of surpass apparent local minimum values. This conclusion corroborates with the review made by Azar et al. [6].



(a) Return x Epoch (b) Return x Execution Time

Figure 12 – DRL method comparison

The Table 6 shows the compiled results for all test cases. Besides the better maximum return achieved by the PPO algorithm, the average time / epoch is almost 10 times lower than the other algorithms, indicating that, among the tested methods, the PPO is the best for this control case.

Table 6 – DRL method comparison

METHOD	MAXIMUM RETURN	AVERAGE TIME [s] / EPOCH
PPO	5751	1.9
DDPG	5153	21.7
SAC	5274	21.3

5. Conclusion

This study showed that the Deep Reinforcement Learning has a promising application in eVTOL flight control. Here, it was demonstrated that the PPO algorithm is the best suited for hover control, as expected by previous references ([6], [8]), as it delivered the best results both in algorithm speed and final agent performance (average return).

Moreover, for this simple hover case, the Neural Network architecture and activation function choice have a major impact in the final agent performance. After optimizing these parameters by trial and error, other parameters had a minor effect: for the PPO the default SpinningUp parameters already delivered results better than the PID baseline, and for DDPG and SAC the hyperparameters variation did not helped to improve the performance over the 5000 threshold

The next steps is to expand this study for the vertical to horizontal flight transition, tilting the wings (and motors with it) in order to accelerate the aircraft minimizing altitude loss and also minimizing energy consumption.

6. Contact Author Email Address

Danilo Sartori Alarcon - mail to: danilo.alarcon@usp.br
Jorge Henrique Bidinotto - mail to: jhbidi@sc.usp.br

7. Copyright Statement

References

[1] ARCHER. Archer and atlas crest announce strategic reset of transaction terms to further align with world-class investors and achieve long-term partnership. Archer, 07 2021. Available at: <https://archer.com/news/archer-and-atlas-crest-announce-strategic-reset-of-transaction-terms-to-further-align-with-world-class-investors-and-achieve-long-term-partnership-oscar-munoz-former-united-airlines-chairman-and-ceo-joins-the-archer-board>

[2] EVE AIR MOBILITY. *Fusão com americana Zanite aproxima Eve da Bolsa de NY*. May-2022. Available at: <https://br.investing.com/news/stock-market-news/fusao-com-americana-zanite-aproxima-eve-da-bolsa-de-ny-997892>.

[3] GARRETT-GLASER, B. *Joby aviation raises $1.6 billion in SPAC merger at $6.6 billion valuation*. Electric VTOL News, feb-2021. Available at: <https: //evtol.com/features/joby-aviation-evtol-spac-merger-reinvent-technology/>.

[4] HEAD, E. *Lilium reveals seven-seat eVTOL and confirms SPAC merger with Qell*. eVTOL Magazine, mar-2021. Available at: <https://evtol.com/news/ lilium-seven-seat-evtol-spac-merger-qell/>.

[5] ALCOCK, C. *Vertical makes wall street eVTOL takeoff with $300 million flotation*. Future Flight, dec-2021. Available at: <https://www.futureflight.aero/news-article/2021-12-16/ vertical-makes-wall-street-evtol-takeoff-300-million-flotation>.

[6] AZAR, A. T. et al. *Drone deep reinforcement learning: A review*. MDPI Electronics, 4 2021.

[7] BøHN, E. COATES, E. M. MOE, S. JOHANSEN, T. A. *Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization*. ICUAS, jun-2019.

[8] KOCH, W. MANCUSO, R. WEST, R. BESTAVROS, A. *Reinforcement learning for UAV attitude control*. ACM Trans. Cyber Phys, 4 2019.

[9] XU, J. DU, T. FOSHEY, M. LI, B. ZHU, B. SCHULZ, A. MATUSIK, W. *Learning to fly: Computational controller design for hybrid uavs with reinforcement learning*. ACM Transactions on Graphics. 2019.

[10] SUTTON, R. BARTO, A. *Reinforcement Learning, an Introduction*. 2nd ed. The MIT Press, 2018.

[11] SCHULMAN, J. WOLSKI, F. DHARIWAL, P. RADFORD, A. KLIMOV, O. *Proximal Policy Optimization algorithms*. CoRR, 2017

[12] LILLICRAP, T. P. HUNT, J. J. PRITZEL, A. HEESS, N. EREZ, T. TASSA, Y. SILVER, D. WIERSTRA, D. *Continuous control with deep reinforcement learning*. 2019.

[13] HAARNOJA, T. ZHOU, A. ABBEEL, P. LEVINE, S. *Soft Actor-Critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor*. 2018.

[14] ABBEEL, P. *Spinning Up in Deep RL*. OpenAI. 2018. Available at <https://spinningup.openai.com>

[15] AIRBUS. Vahana. our single-seat evtol demonstrator. 2019? Available at: <https://www.airbus.com/innovation/zero-emission/urban-air-mobility/vahana.html>.

[16] OPEN AI, *Gym*. 2021. Available at <https://gym.openai.com>.

[17] NASA. *U.S. Standard Atmosphere*. National Oceanic and Atmospheric Administration and National Aeronautics and Space Administration, 1976.

[18] SCHULMAN, J. MORITZ, P. LEVINE, S. JORDAN, M. ABBEEL, P. *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. 2016. arXiv:1506.02438.