Prof. Matteo Matteucci, Giacomo Boracchi

DEEP REINFORCEMENT LEARNING BASED ROBUST QUADCOPTER STABILIZATION SYSTEM

Advanced Deep Learning Models and Methods Research Project, 2022

Lorenzo Poretti, Alessandro Restifo

I. INTRODUCTION

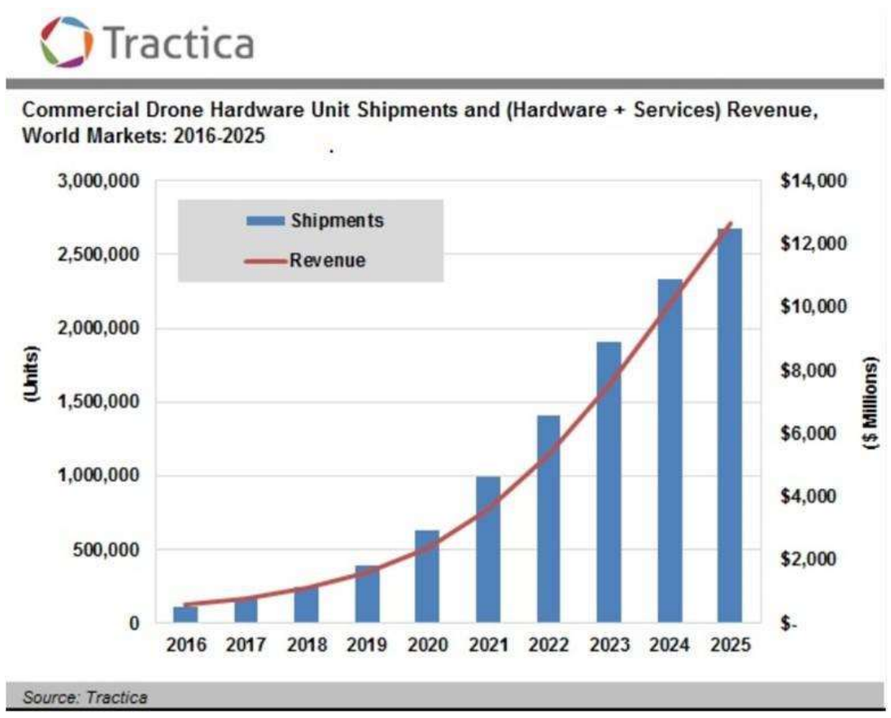
Goals, quadcopters, traditional PID, RL controllers

GOALS

- Explore and test applications of deep RL in the autonomous control systems field
- Develop a robust deep RL based controller for quadcopters

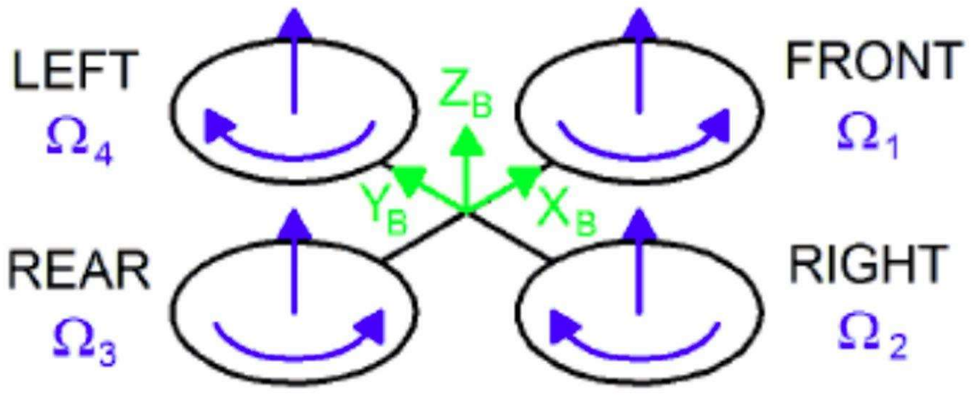
QUADCOPTERS

- Exponential gain in interest all over the world



QUADCOPTERS

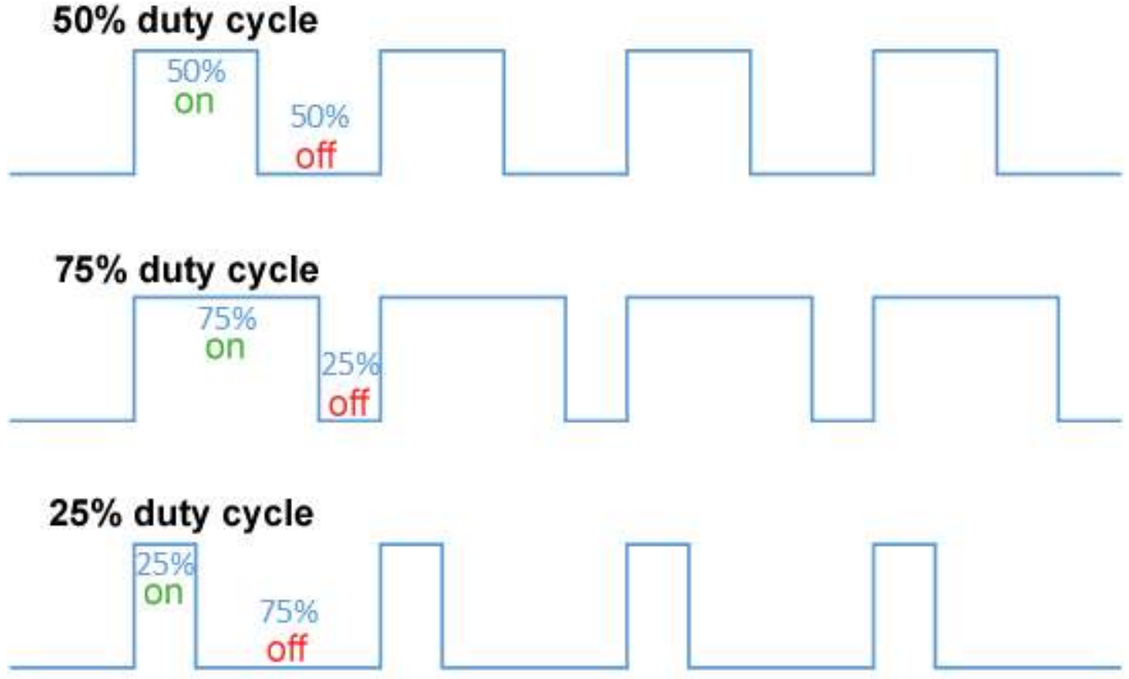
- Four propellers, spinning in opposite directions > overall moment and propellers torque in the system is null



QUADCOPTERS

- Power delivered to propellers via *PWM*
- Reduces average power delivered by discretizing it
- Average value of voltage/current controlled by switching on/off

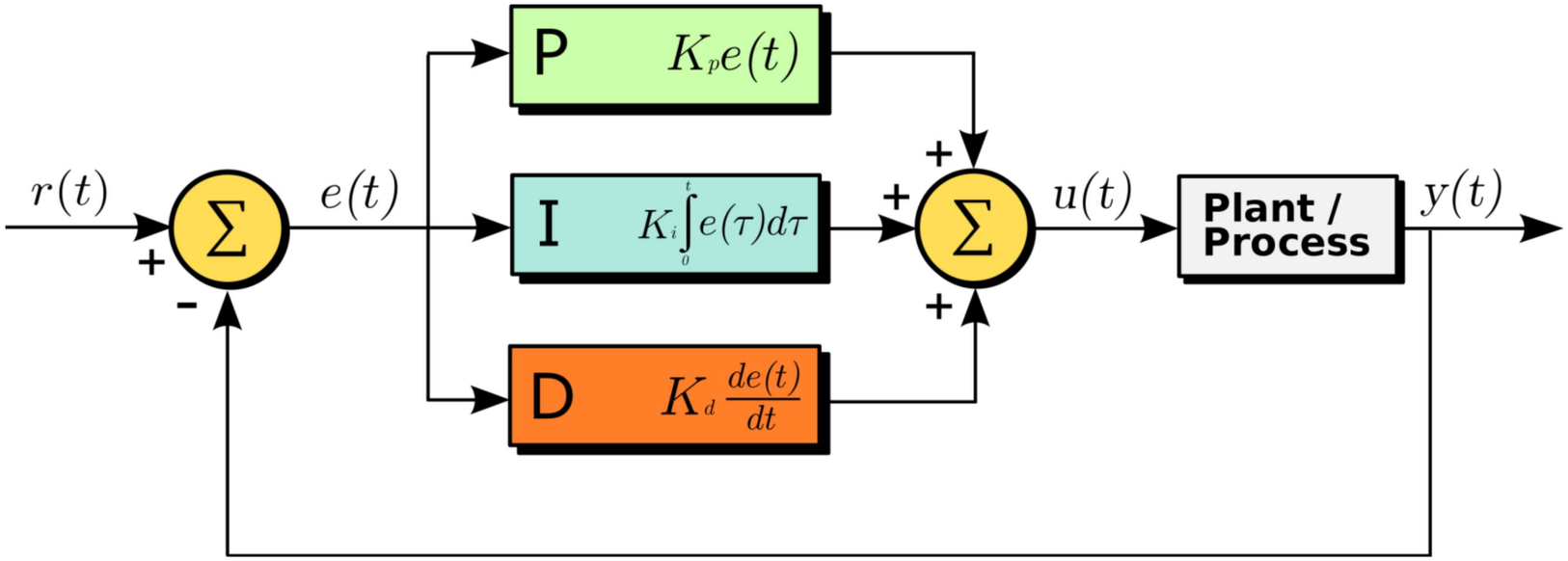
QUADCOPTERS



TRADITIONAL PID CONTROLLER

- Compute error *e(t)* between process set value *r(t)* and measured process value *y(t)*
- Contributions from error derivation and integration summed to proportional contribution and used to set system input

TRADITIONAL PID CONTROLLER



TRADITIONAL PID CONTROLLER

Issues:

- If system encounters challenging changes in the environment, the PID controller may deliver poor performance
- Mostly needs explicit kinematic model

REINFORCEMENT LEARNING CONTROLLER

Advantages:

- No kinematic model needs to be specified: the agent is able to directly model it through data
- Non-linear dynamics and kinematics can be modeled
- In theory, should learn the optimal policy in every challenging environment with feasible control solution

REINFORCEMENT LEARNING CONTROLLER

Environment as MDP:

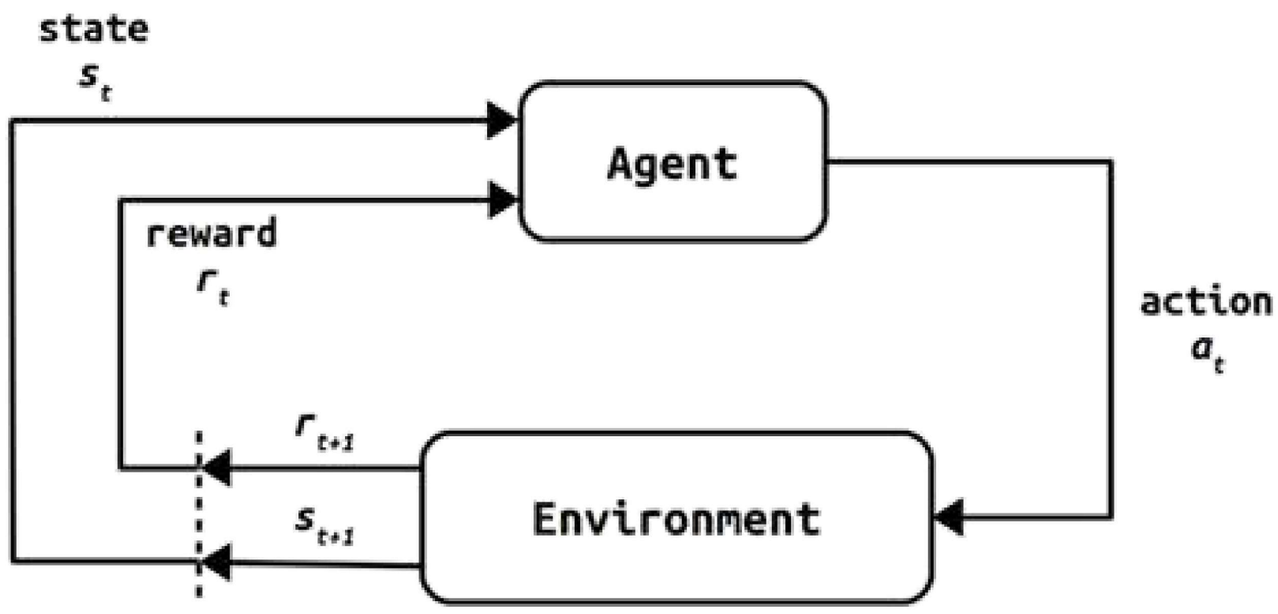
- State set, S
- Action set, A
- Transition probability

$$P_a(s,\ s') = Pr(s_{t+1} = s' \mid s_t = s,\ a_t = a)$$

- Reward after transiting

$$R_a(s,\ s')$$

REINFORCEMENT LEARNING CONTROLLER



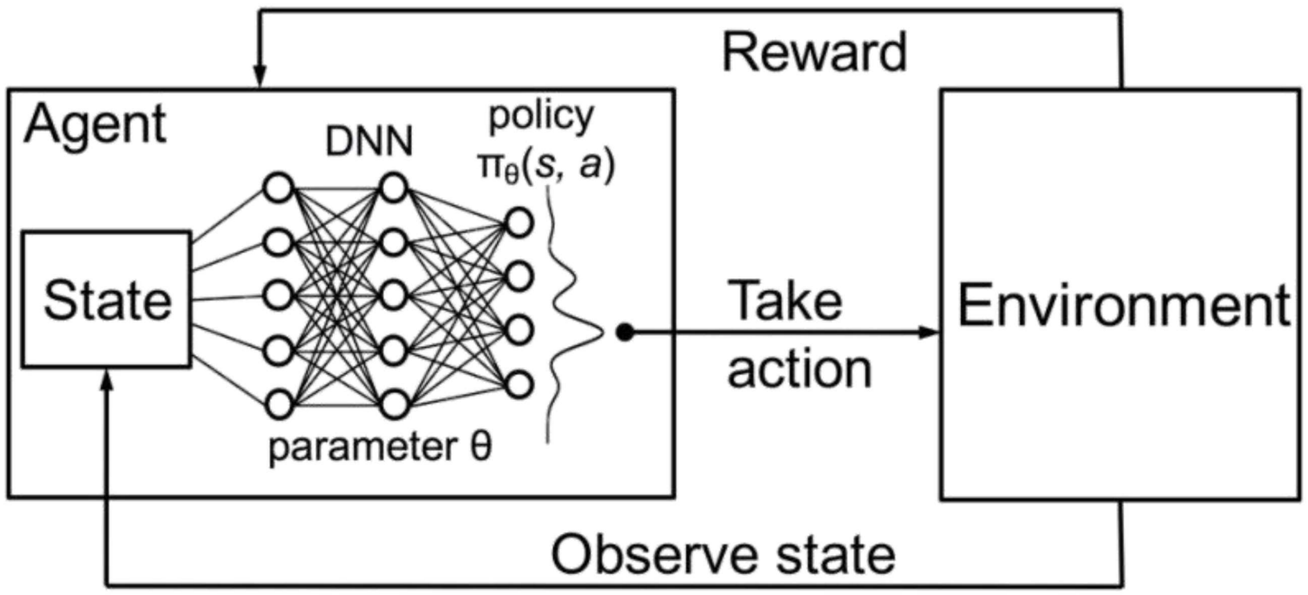
REINFORCEMENT LEARNING CONTROLLER

Deep RL:

- Approximate the Q-value function via neural networks
- In actor-critic frameworks: approximate also the best action given a state
- The state-action mapping is called policy:

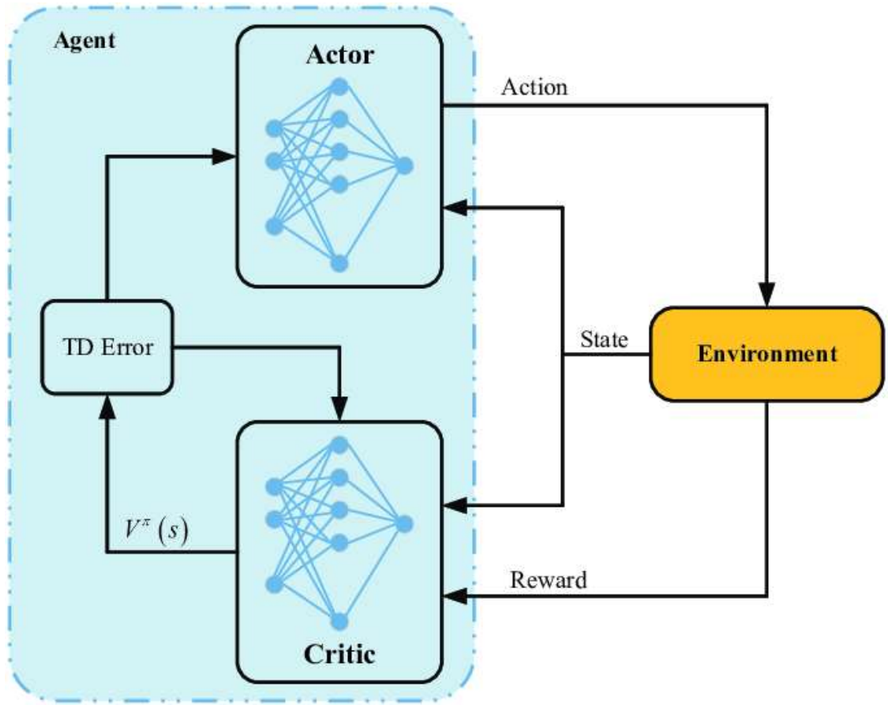
$$\pi(a, s) = Pr(a_t = a \mid s_t = s)$$

REINFORCEMENT LEARNING CONTROLLER



DQN

REINFORCEMENT LEARNING CONTROLLER



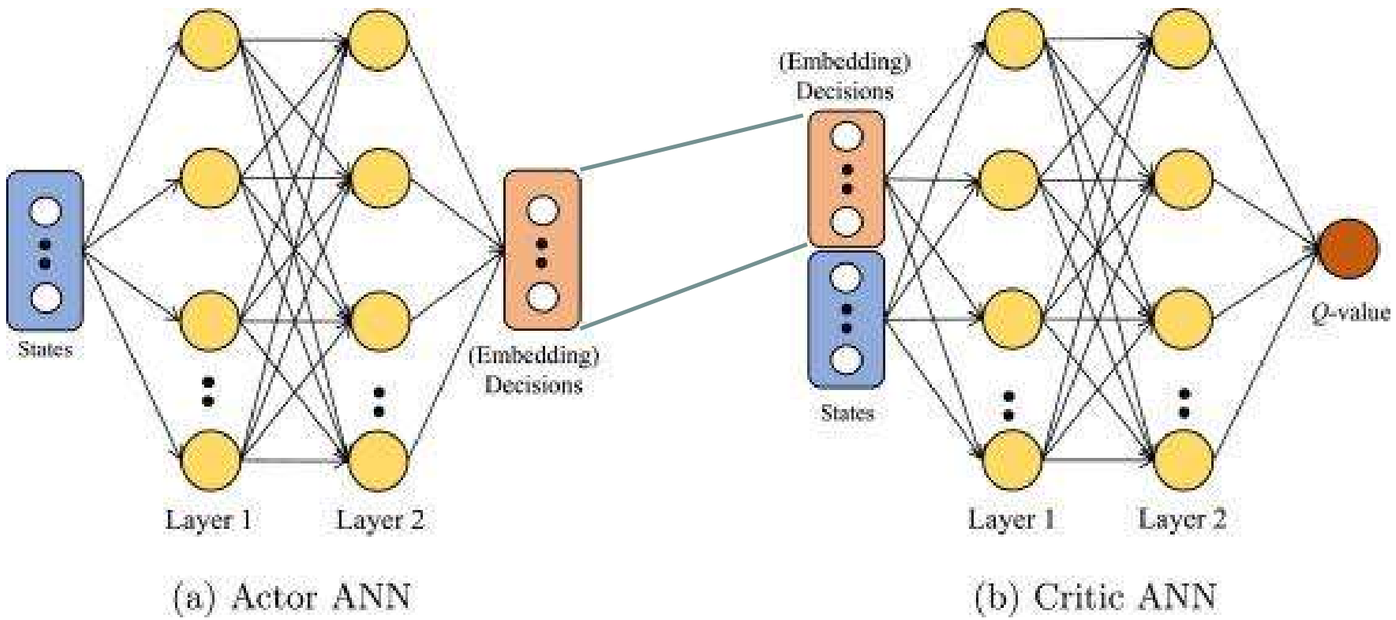
Actor-critic

REINFORCEMENT LEARNING CONTROLLER

Learning methods:

- On-policy algorithms, directly improve the optimal policy learned used to make decisions
- Off-policy algorithms, use different policy for exploration, called behavioral policy

REINFORCEMENT LEARNING CONTROLLER



(a) Actor ANN

(b) Critic ANN

Alternate update of actor/critic (off-policy)

Replay buffer: <*state, action, reward, state'*>

Actor training:

plug actor in critic, freeze critic

maximize Q = critic(actor(state), action)

Critic training:

Q = critic(action, state)

Q' = target_critic(actor(state'), state')

Optimize via Q-learning update equation:

*Q := reward + discount * Q'*

REINFORCEMENT LEARNING CONTROLLER

Previous works:

- [Chiang] focuses on a simple on-policy PPO controller
- [Lynch] extends [Chiang]. Focuses on Proximal Policy Optimization, extends the work evaluating different possible reward functions

REINFORCEMENT LEARNING CONTROLLER

Previous works:

- [Huffer] explores off-policy learning, with DDPG and custom RL algorithms. Their networks have 2x64 tanh activated units both for actor and critic
- [Huan] presents a robust controller made of RL actor and compensator. Their actor uses 2x32 relu-activated units. They test their system with wind disturbance up to 3.6 m/s

II. APPROACH

Agents, networks, thrust management, simulation environment

GENERAL APPROACH

- Deep reinforcement learning controller
- Off-policy learning (much higher sample efficiency)
- Agent chosen is Twin Delayed Deep Deterministic Policy Gradient (TD3), one of the latest and most effective off-policy algorithms

AGENTS EVALUATION

Two main techniques available:

- Discretization of action space, usage of DQN (deep Q-network)
- Usage of **actor-critic** methods

AGENTS EVALUATION

Many actor-critic paradigm available. On-policy:

- TRPO: uses trust regions where local approximations of policy function (differentiated for the policy gradients) are accurate
- PPO: substitutes trust regions by imposing a limit on the ratio between the new policy and old policy:

$$r_t = \pi(a_t|s_t) \,/\, \pi_{old}(a_t|s_t),$$

$$r_t \in (1-\epsilon,\ 1+\epsilon)$$

result: objective function is clipped, updates via gradient kept small enough. Nearly monotonic reward increase.

AGENTS EVALUATION

Many actor-critic paradigm available. Off-policy:

- DDPG: same update equation as Q-learning, uses replay buffers and target networks
- TD3: DDPG with Clipped Double-Q Learning, “Delayed” Policy Updates, Target Policy Smoothing
- SAC: similar to TD3, adds entropy regularization, lacks the target smoothing

AGENTS EVALUATION

Test executed with same settings:

- Reward function chosen:

$$R_a(s,s') = max(0,\ 1-||x'||) - 0.1\ ||\theta'|| - 0.1\ ||w'||,$$

x' new position of drone, θ' new orientation, w' new angular velocity, after action

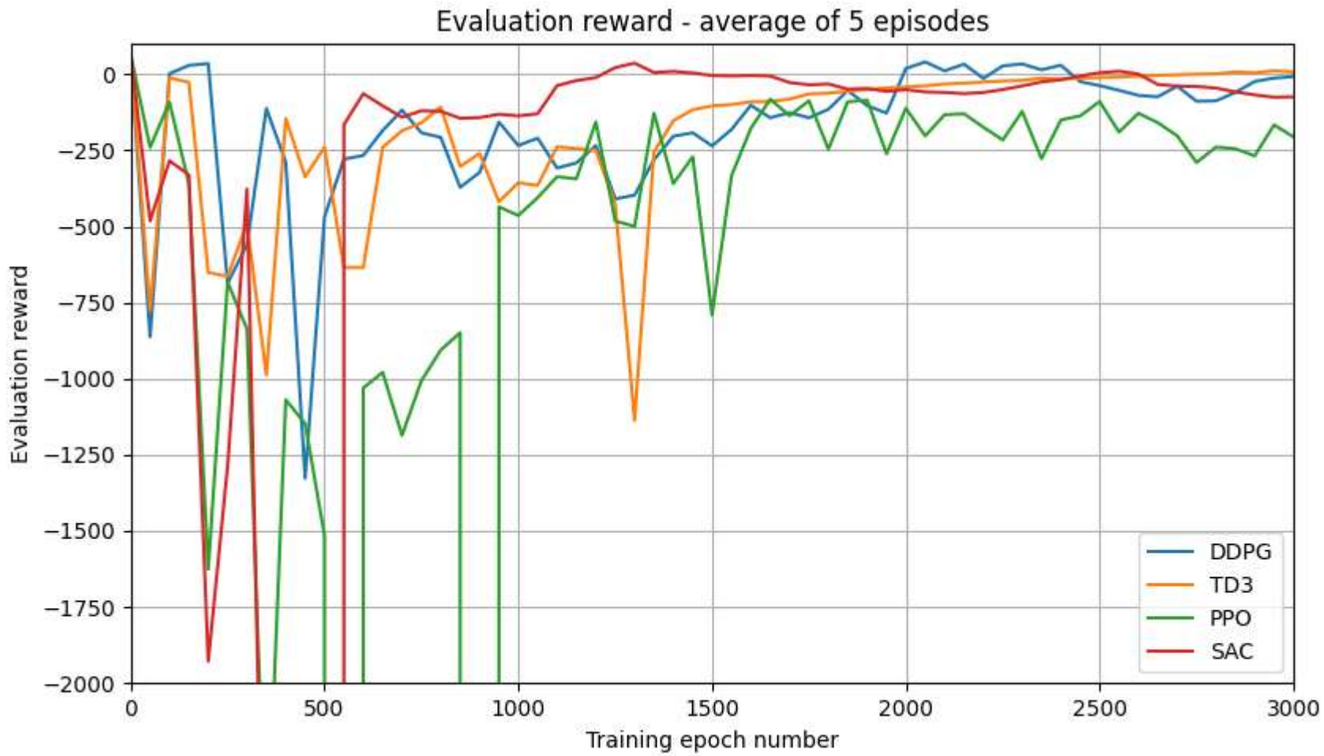
- Special attention on orientation and angular velocity, main causes of instability

AGENTS EVALUATION

Test executed with same settings:

- Replay buffer dimension: 1M timesteps
- Training episodes length: 200 timesteps
- Evaluation episodes length: 400 timesteps
- Batch size: 512
- Learning rate: 3e-4 (1e-3 for PPO, to speed up learning process)

AGENTS EVALUATION



Test results

AGENTS EVALUATION

In our testing, TD3 performed the best overall, with faster learning and better visual results.

NEURAL NETWORK STRUCTURE

Structure of actor and critic is different, as their function:

- Actor estimates best action given a state
- Critic estimates Q-value of a state-action pair

NEURAL NETWORK STRUCTURE

Input state:

- 19 fp32 values
- 3 values for Δx, difference between current position and set point
- 4 values for $\Delta\theta$, difference between current orientation and set point orientation
- 3 values for $<dw_x/dt, dw_y/dt, dw_z/dt>$ angular accelerations
- 3 values for $<w_x, w_y, w_z>$ angular velocities
- 3 values for $<dv_x/dt, dv_y/dt, dv_z/dt>$ linear accelerations
- 3 values for $<v_x, v_y, v_z>$ linear velocities

NEURAL NETWORK STRUCTURE

Actions:

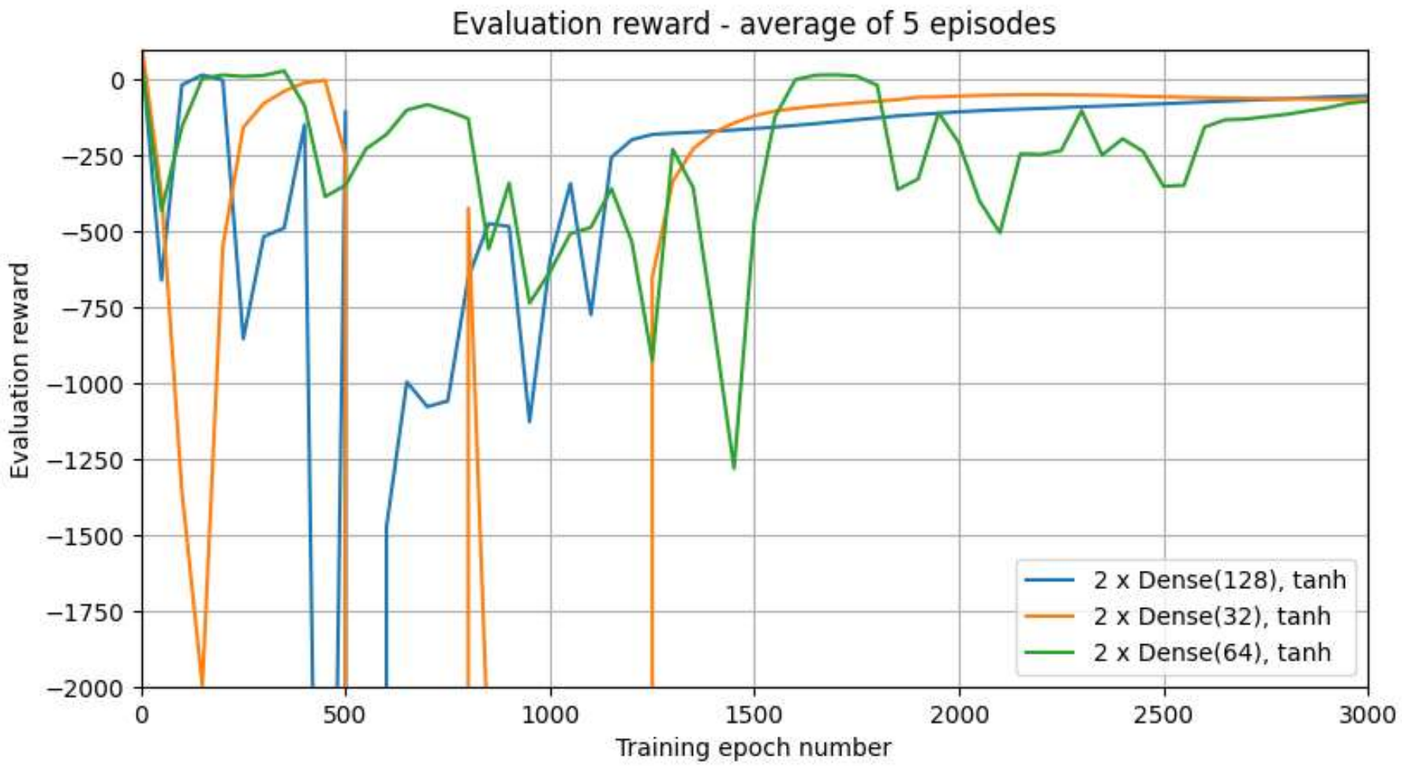
- 4 fp32 values
  - Front right thrust
  - Rear left thrust
  - Front left thrust
  - Rear right thrust

NEURAL NETWORK STRUCTURE

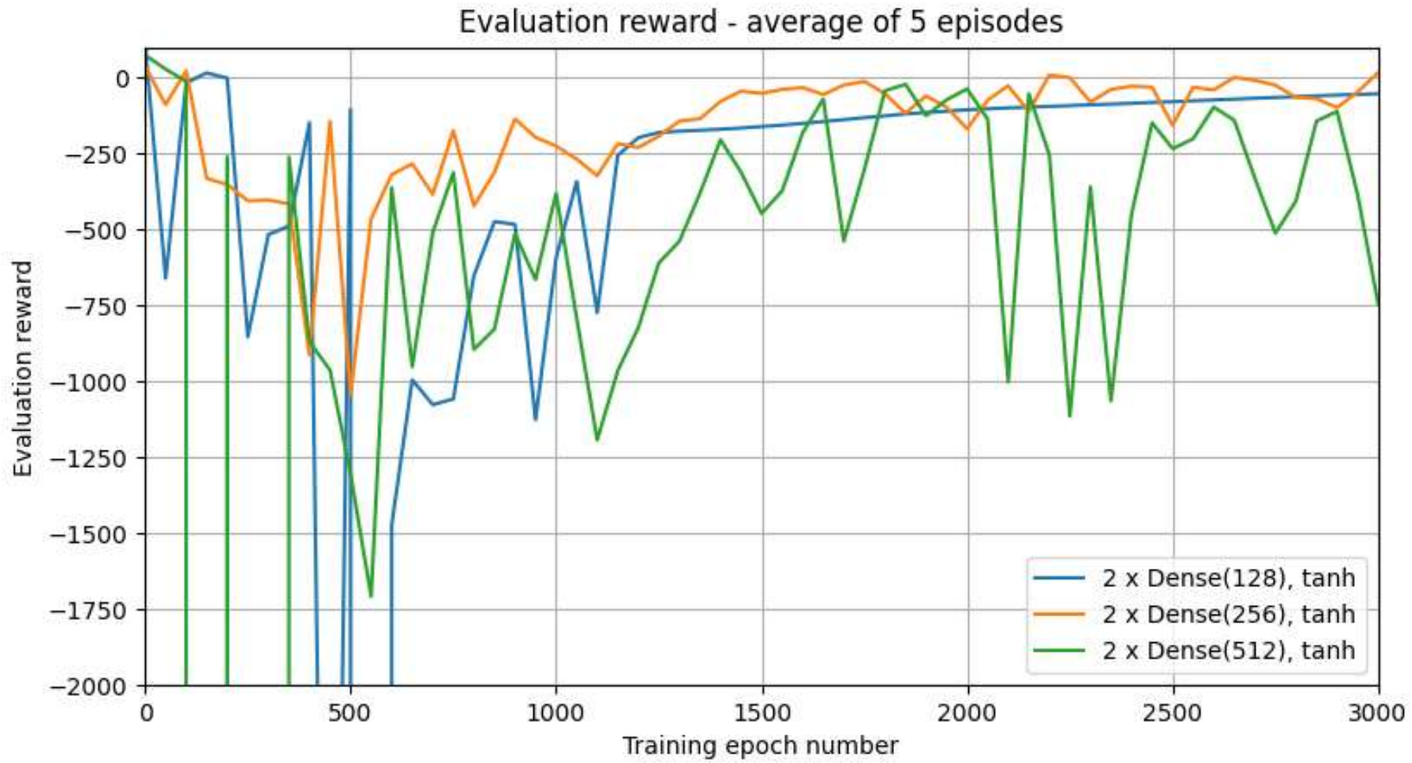
Various networks tests, executed with same settings:

- Agent: TD3
- Replay buffer dimension: 1M timesteps
- Batch size: 512
- Learning rate: 3e-4

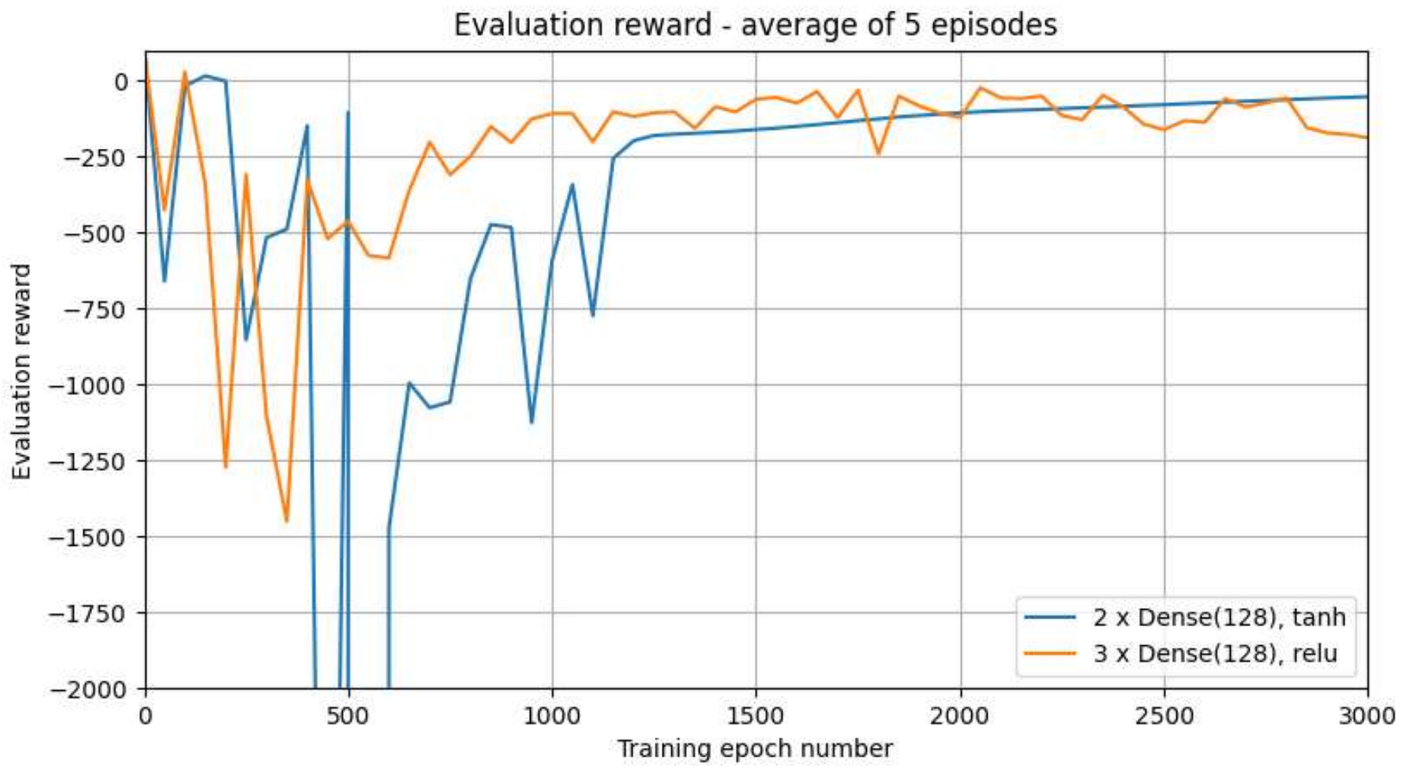
NEURAL NETWORK STRUCTURE



NEURAL NETWORK STRUCTURE



NEURAL NETWORK STRUCTURE



NEURAL NETWORK STRUCTURE

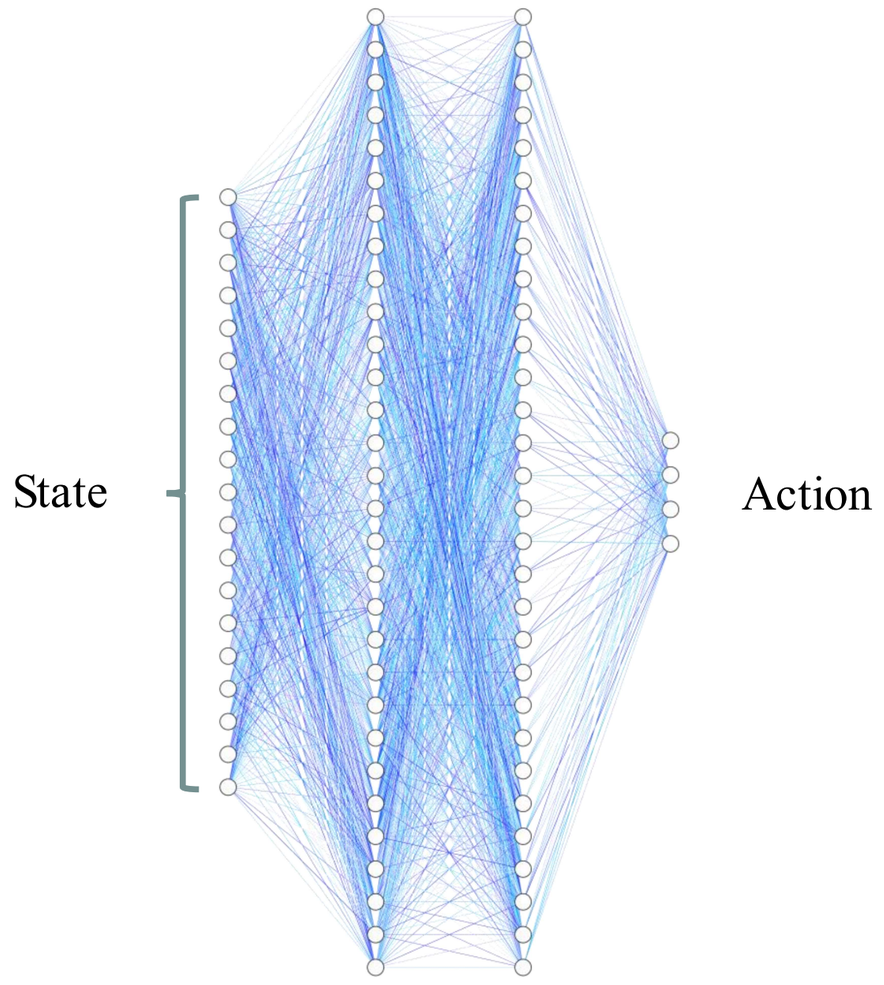
Final actor network:

- Input layer
- 2 hidden layers, 128 units each, tanh activation
- Output layer, tanh activation

Tanh activation seems to perform better than sigmoid or ReLU.

Might be due to the symmetry of the function and the “pseudo-regularization” of the layers output, byproduct of the squashing of the output in the (-1, 1) range.

NEURAL NETWORK STRUCTURE



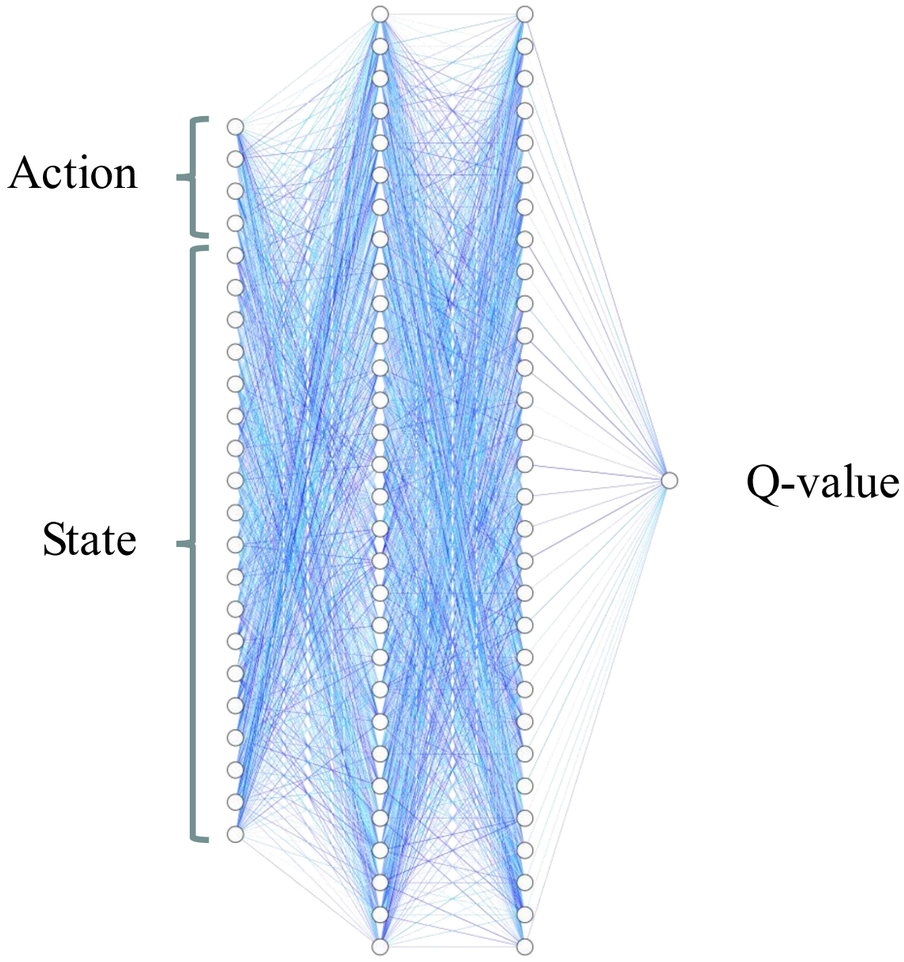
Input Layer $\in R^{19}$ | Hidden Layer $\in R^{128}$ | Hidden Layer $\in R^{128}$ | Output Layer $\in R^{4}$

NEURAL NETWORK STRUCTURE

Final critic network:

- Input layer
- 2 hidden layers, 128 units each, ReLU activation
- Output layer, linear activation

NEURAL NETWORK STRUCTURE



Input Layer $\in R^{23}$ | Hidden Layer $\in R^{128}$ | Hidden Layer $\in R^{128}$ | Output Layer $\in R^{1}$

THRUST MANAGEMENT

- Previous works directly estimate the *PWM* value to supply to the drone through the actor network
- Instead, we interpret the network output as a scaled delta thrust: the network output is constrained in a smaller interval, to allow for more fine-grained control

THRUST MANAGEMENT

The output *PWM*, supplied to each rotor *i*, is:

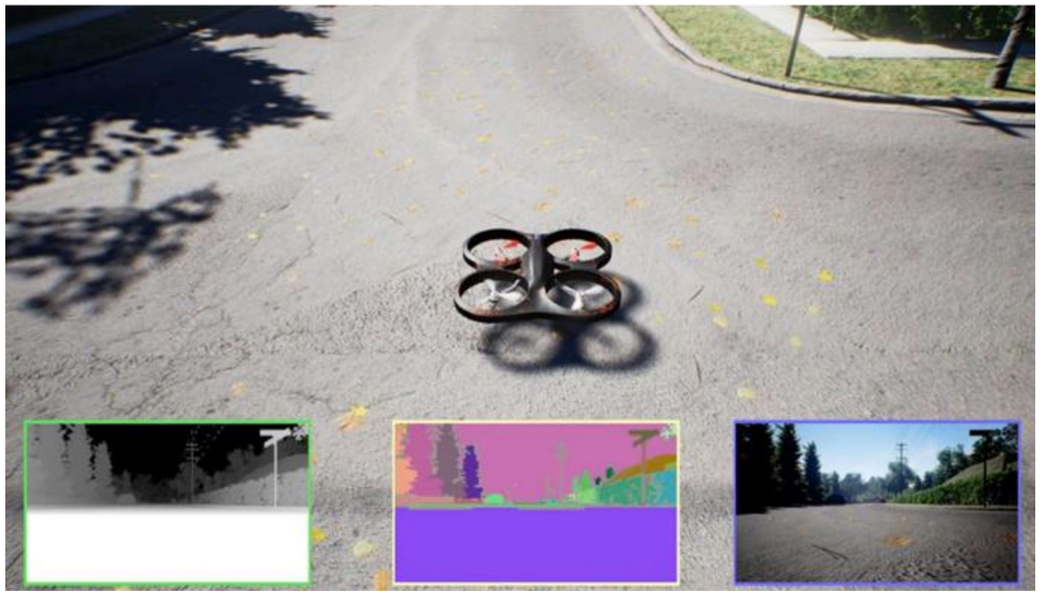
$$\phi[i] = clip(\beta + \gamma(\sigma[i] - 0.5), 0, 1) \ \forall i \in [0, 3], \phi[i] \in (0, 1)$$

- β *[= 0.59]*: thrust bias, which stabilize the drone in ideal conditions (given by the specific quadcopter physical structure)
- $\sigma \in (0, 1)$: tanh output of the neural network, squashed in the range (0, 1)
- γ *[= 0.4]*: scaling factor which constrains the neural network output

SIMULATION ENVIRONMENT

AirSim:

- Opensource simulator
- Built upon Unreal Engine
- Developed by Microsoft
- Focus on simulations of quadcopters and ground vehicles, for RL development



SIMULATION ENVIRONMENT

Exposes API as interface between python code and PhysX simulator.

Usual pipeline:

- Modify environment, e.g. setting wind
- Receive data from virtual sensors
- Process data with custom script
- Provide commands to actuators
- Deploy on multiple flight controllers, e.g. PX4 or ArduPilot

SIMULATION ENVIRONMENT

- AirSim adopts a *z*-down reference system
- Position: center of quadcopter in 3D space, $\langle x, y, z \rangle$ coordinates [m]
- Orientation: quaternion $\langle x, y, z, w \rangle$ (roll, pitch, yaw angles are derived)
- Initial configuration of the quadcopter (reset at each episode):

$$position := \langle 0, 0, -100 \rangle$$

$$orientation := \langle 0, 0, 0, 1 \rangle$$

III. RESULTS

Experimental setup, analysis

EXPERIMENTAL SETUP

Final TD3 training settings:

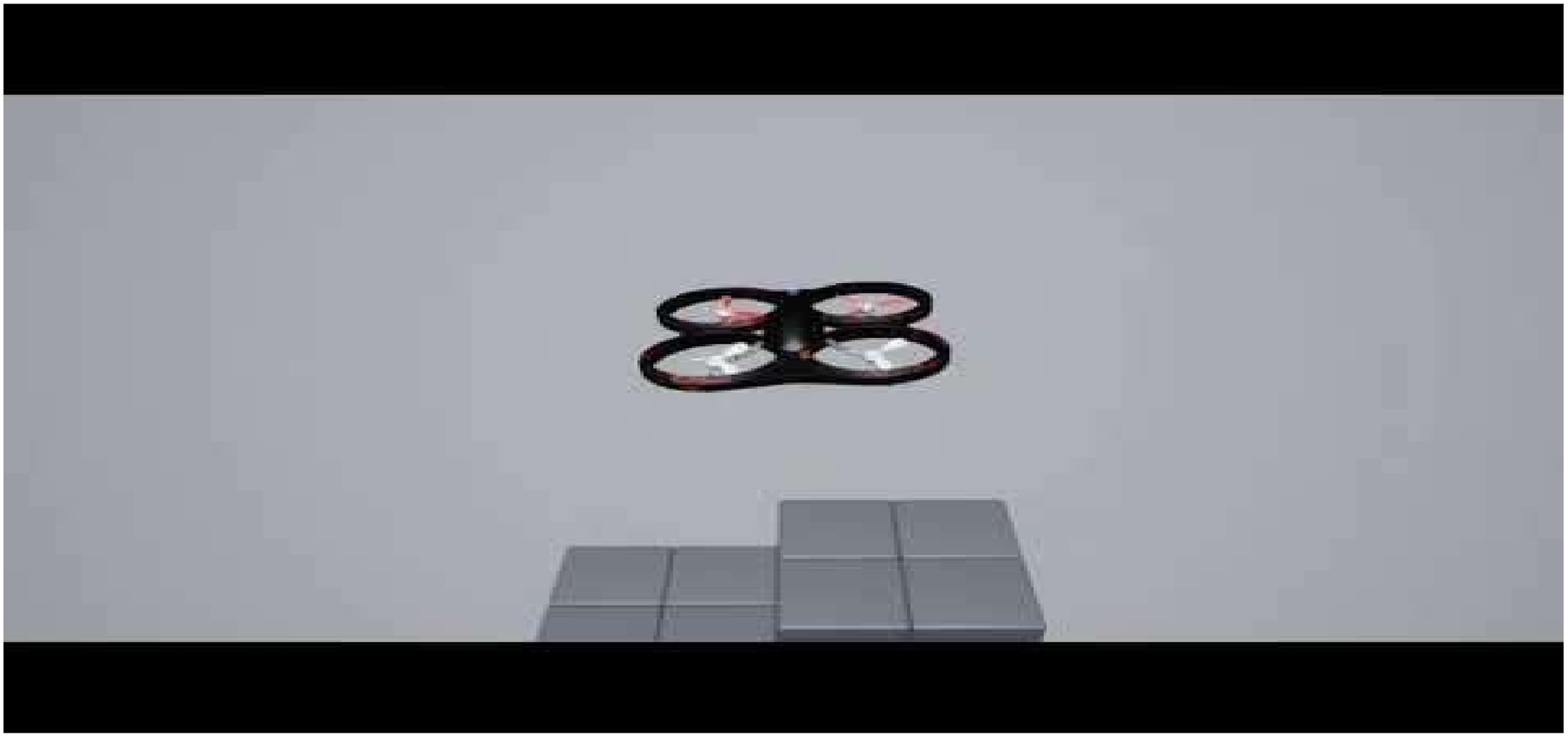
- Gaussian random wind ~N(0 m/s, 2.5 m/s)
- Batch-size: 512
- Learning rate: 1e-3

EXPERIMENTAL SETUP

- Replay buffer dimension: 1M timesteps
- Initial collection (random policy): 5000 timesteps
- Training episodes length: 200 timesteps
- Evaluation episodes length: 400 timesteps
- Evaluation frequency: 1 every 50 training epochs
- Number of episodes for each evaluation: 5, rewards averaged

ANALYSIS

- TD3 Reinforcement Learning algorithm obtained the best policy to stabilize the quadcopter
- The best model has been obtained after training for 5500 epochs



TD3 training process, epochs 0 to 5500

ANALYSIS

Three sensitivity tests are performed until system failure

- Wind stress test
- Initial orientation test
- Initial position test

ANALYSIS

Wind stress test results. Robustness to:

Axis	Maximum wind (neg)	Maximum wind (pos)
x	-20 m/s	16 m/s
y	-22 m/s	20 m/s
z	-6 m/s	6 m/s

Most commercial drones cannot sustain wind > 10 m/s

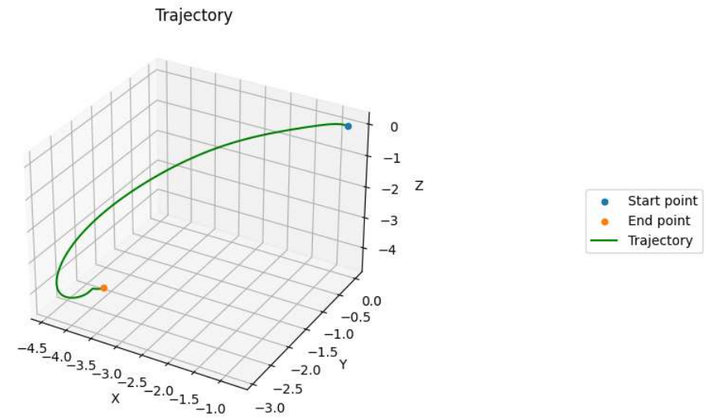
ANALYSIS

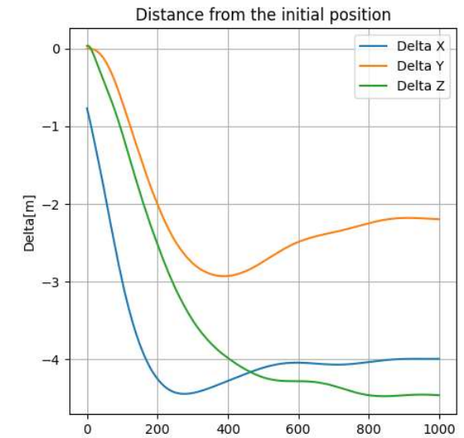
Initial parameters

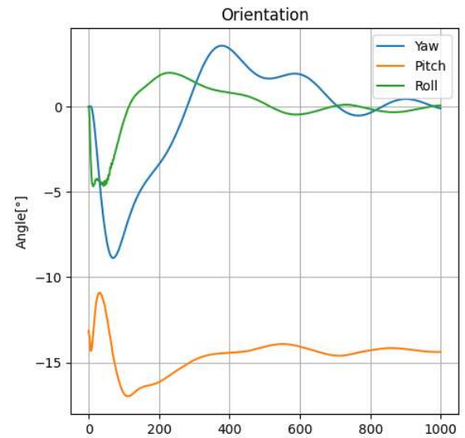
Δx = 0.0 m | Δy = 0.0 m | Δz = 0.0 m

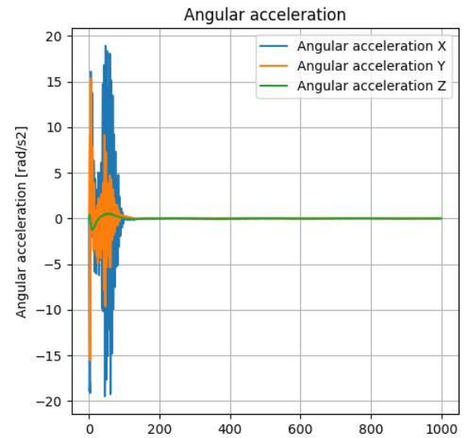
roll = 0 deg | pitch = 0 deg | yaw = 0 deg

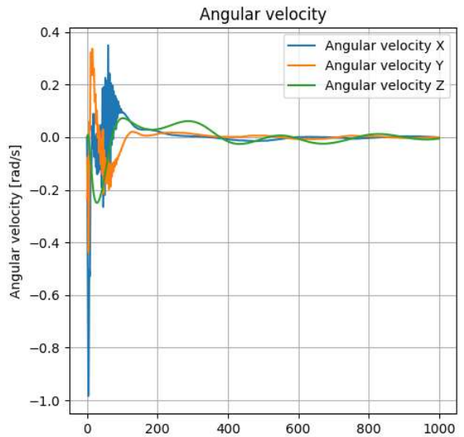
wind_x = -20.0 m/s | wind_y = 0.0 m/s | wind_z = 0.0 m/s

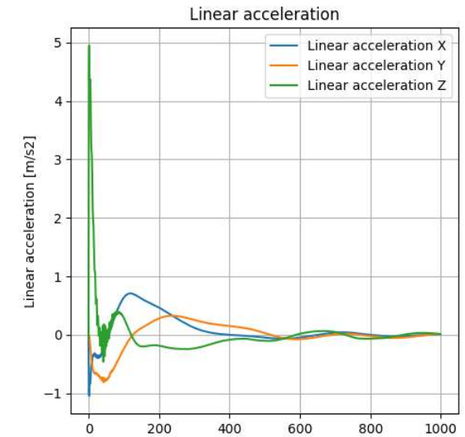


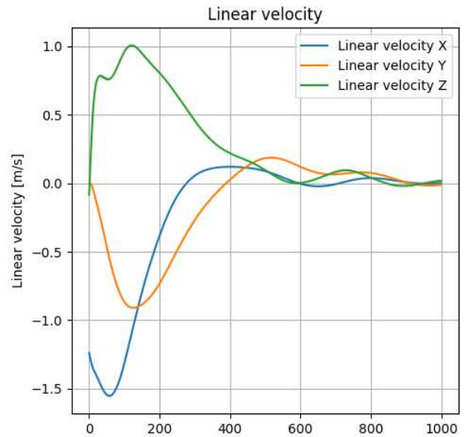


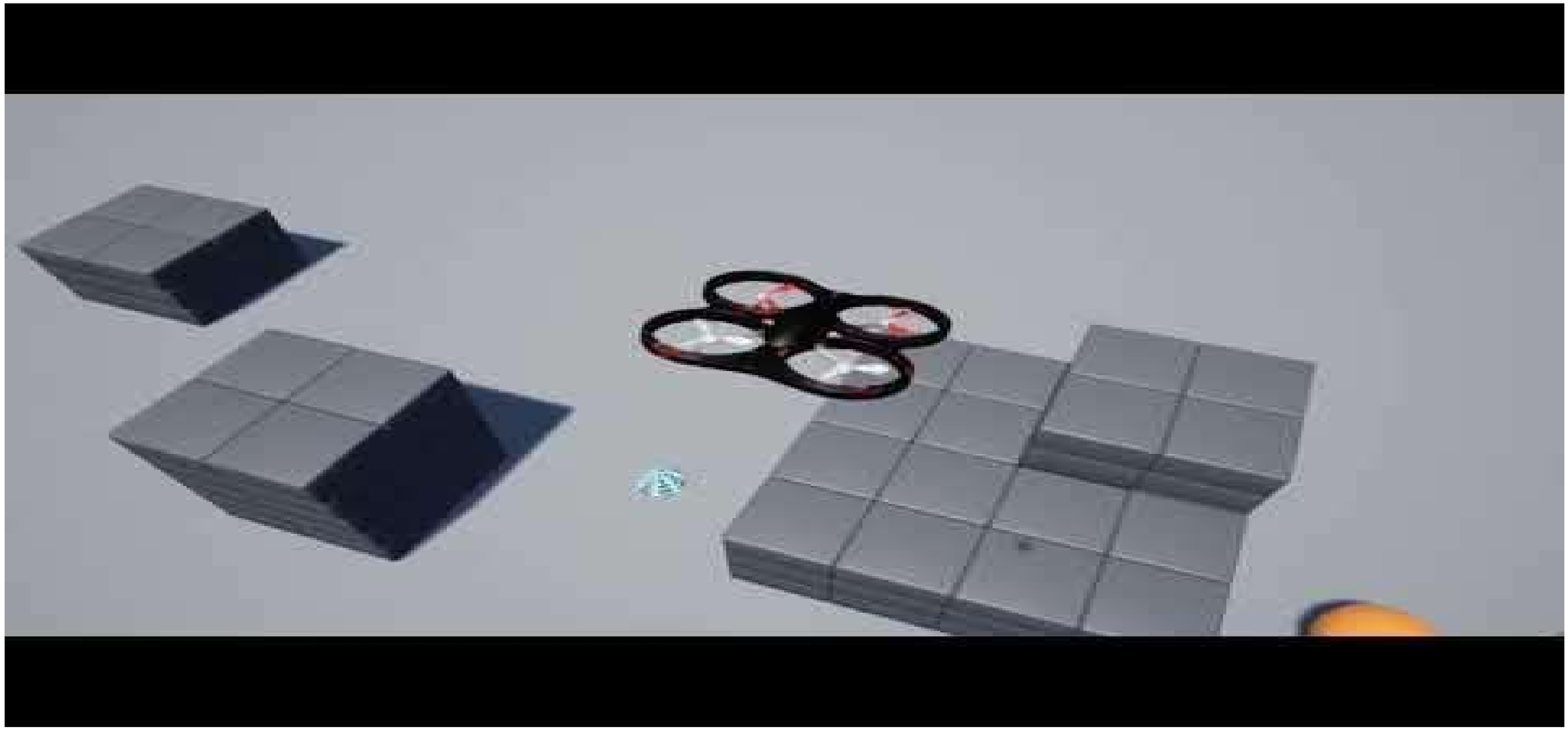












TD3 evaluation, wind_x = -20 m/s

Notice the 15 deg pitch to counteract the wind

ANALYSIS

Orientation test results (steps of 30 deg). Robustness to:

Rotation axis	Maximum angle
Roll	180 deg
Pitch	90 deg
Yaw	120 deg

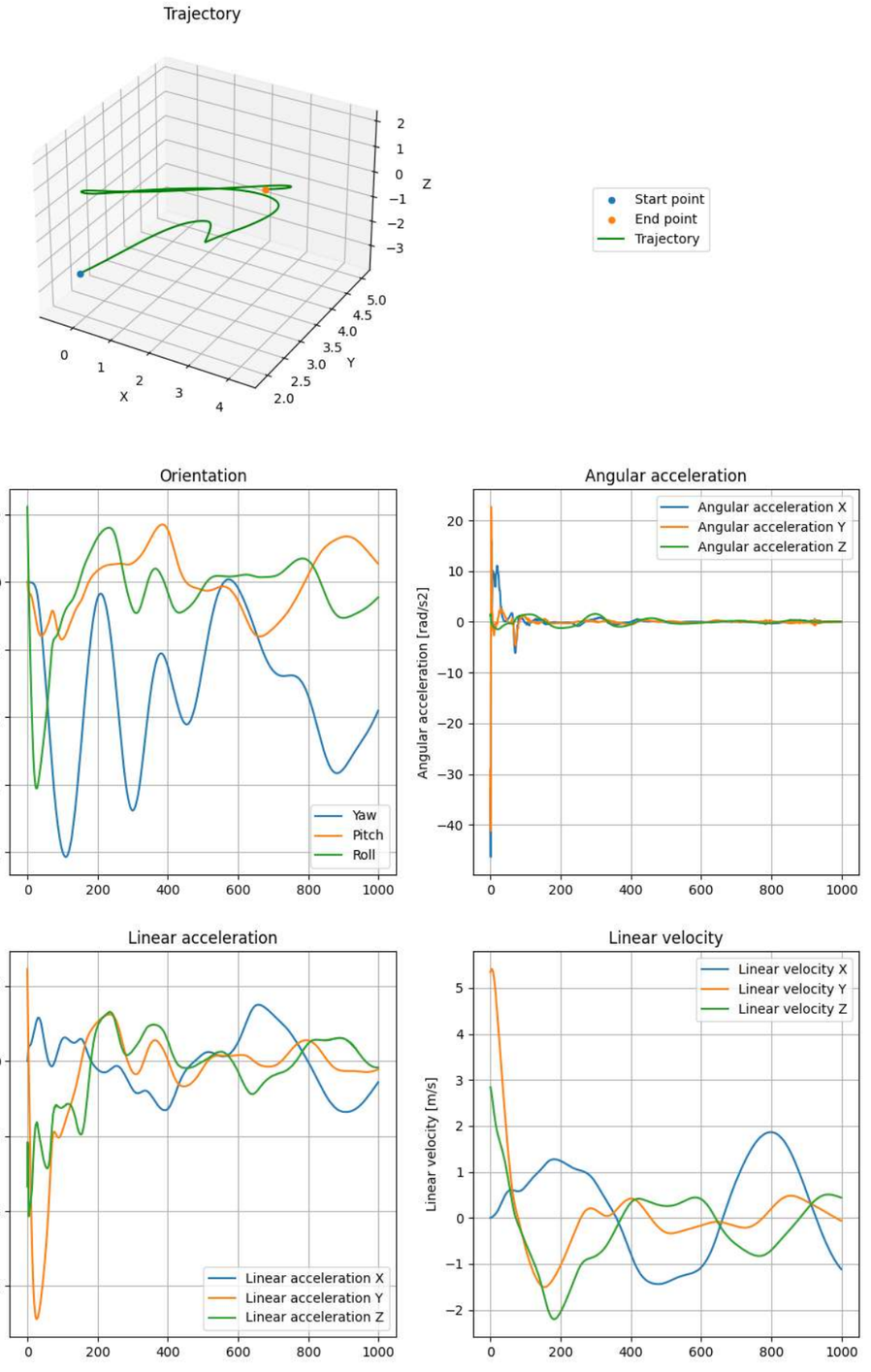
ANALYSIS

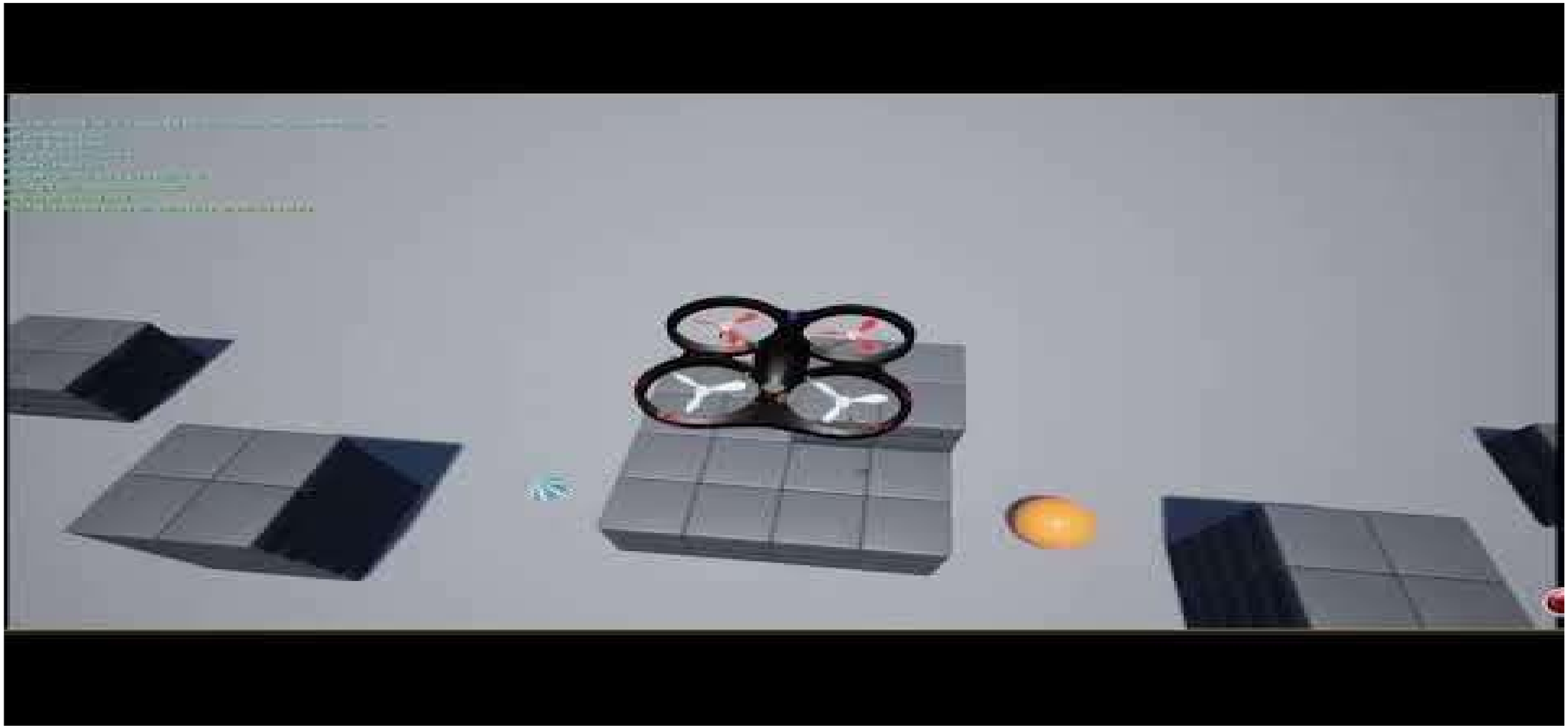
Initial parameters

$\Delta x = 0.0$ m | $\Delta y = 0.0$ m | $\Delta z = 0.0$ m

roll = 120 deg | pitch = 0 deg | yaw = 0 deg

wind_x = 0.0 m/s | wind_y = 0.0 m/s | wind_z = 0.0 m/s





TD3 evaluation, roll = 120 deg

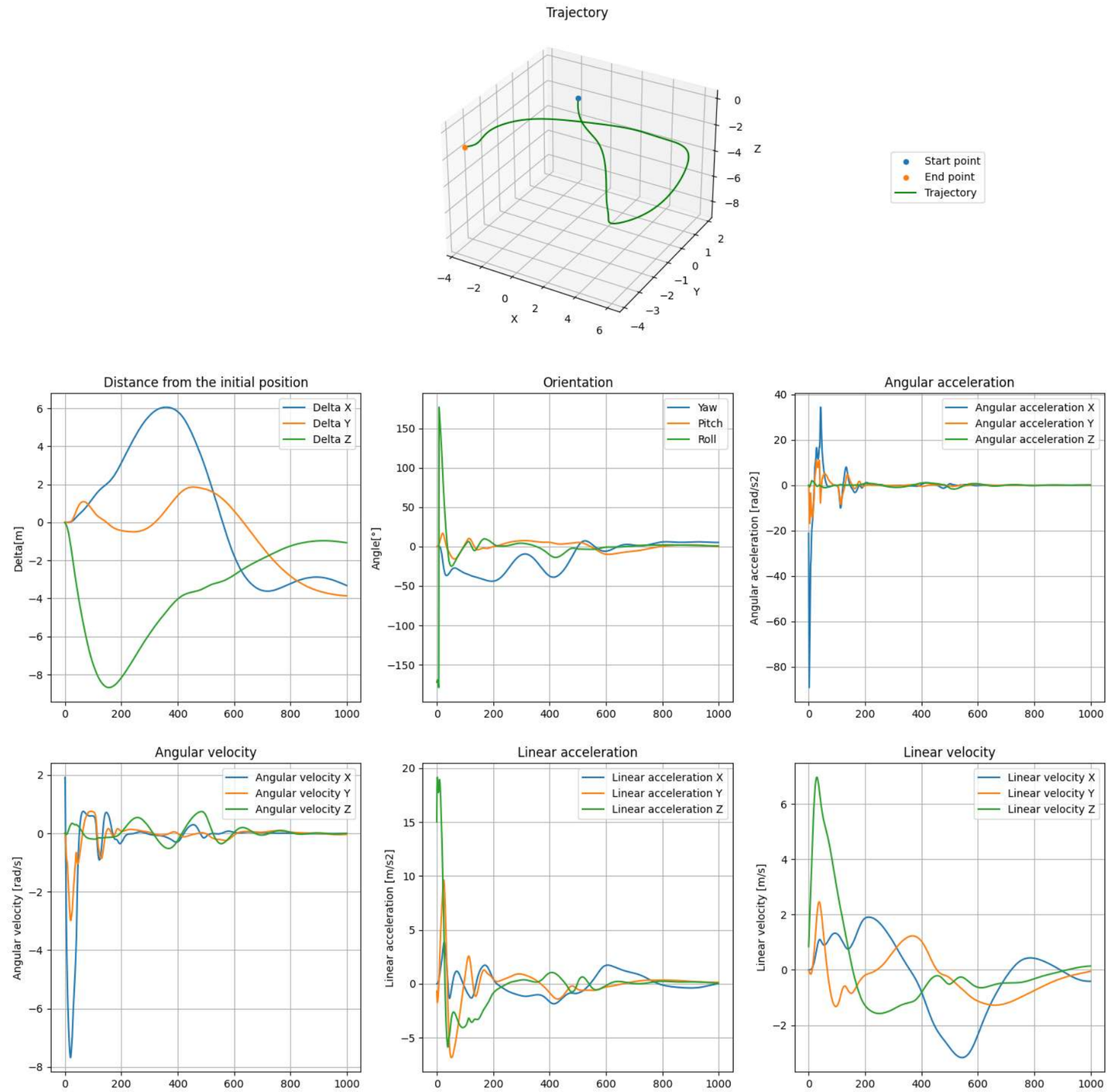
ANALYSIS

Initial parameters

$\Delta x = 0.0$ m | $\Delta y = 0.0$ m | $\Delta z = 0.0$ m

roll = 180 deg | pitch = 0 deg | yaw = 0 deg

wind_x = 0.0 m/s | wind_y = 0.0 m/s | wind_z = 0.0 m/s



TD3 evaluation, roll = 180 deg

ANALYSIS

Position test results (steps of 5 m). Robustness to:

Axis	Maximum distance (neg)	Maximum distance (pos)
x	-20 m	10 m
y	-10 m	10 m
z	-25 m	15 m

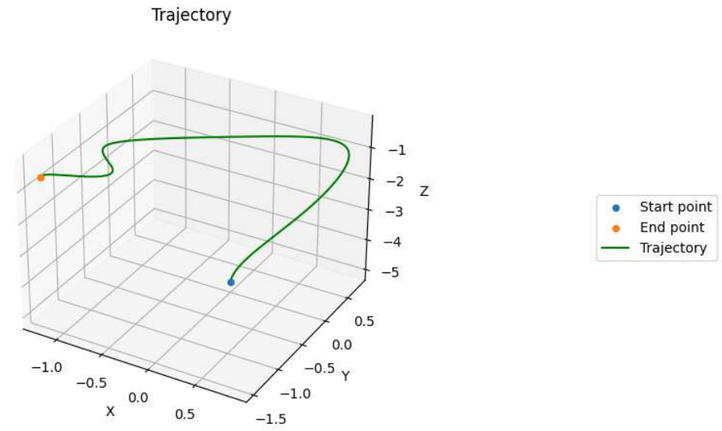
ANALYSIS

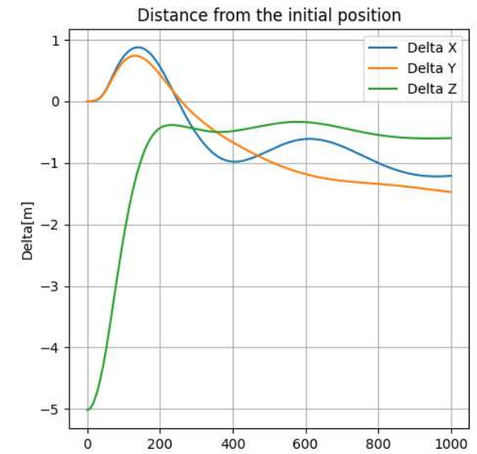
Initial parameters

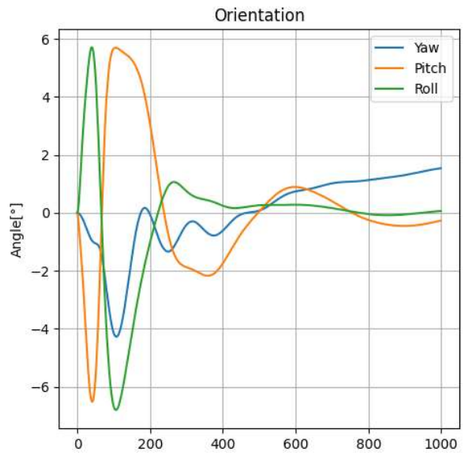
Δx = 0.0 m | Δy = 0.0 m | **Δz = 5.0 m**

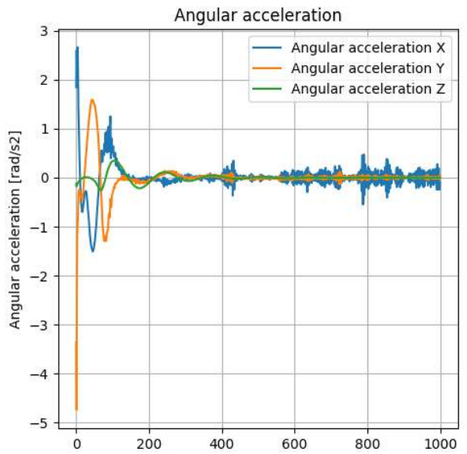
roll = 0 deg | pitch = 0 deg | yaw = 0 deg

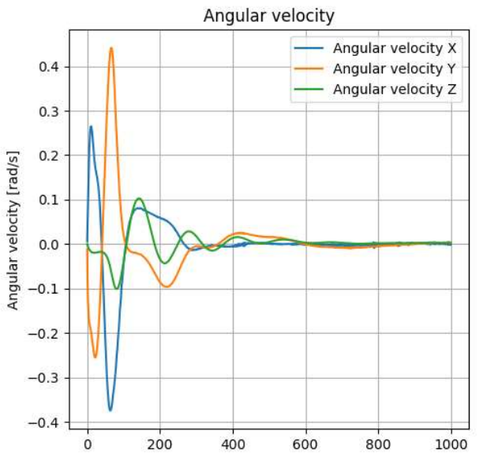
wind_x = 0.0 m/s | wind_y = 0.0 m/s | wind_z = 0.0 m/s

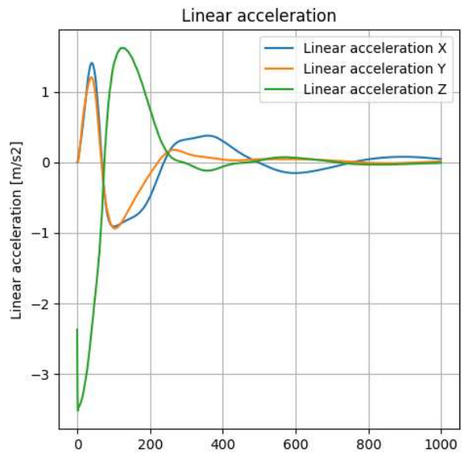


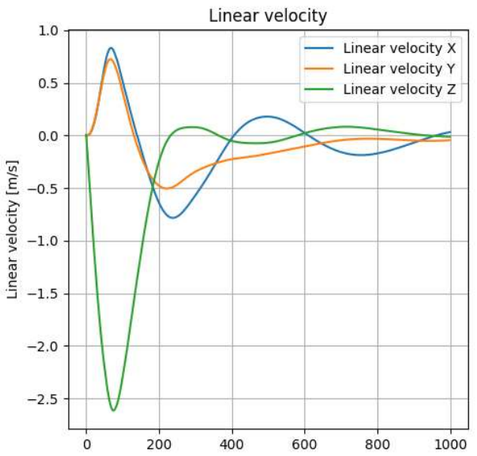


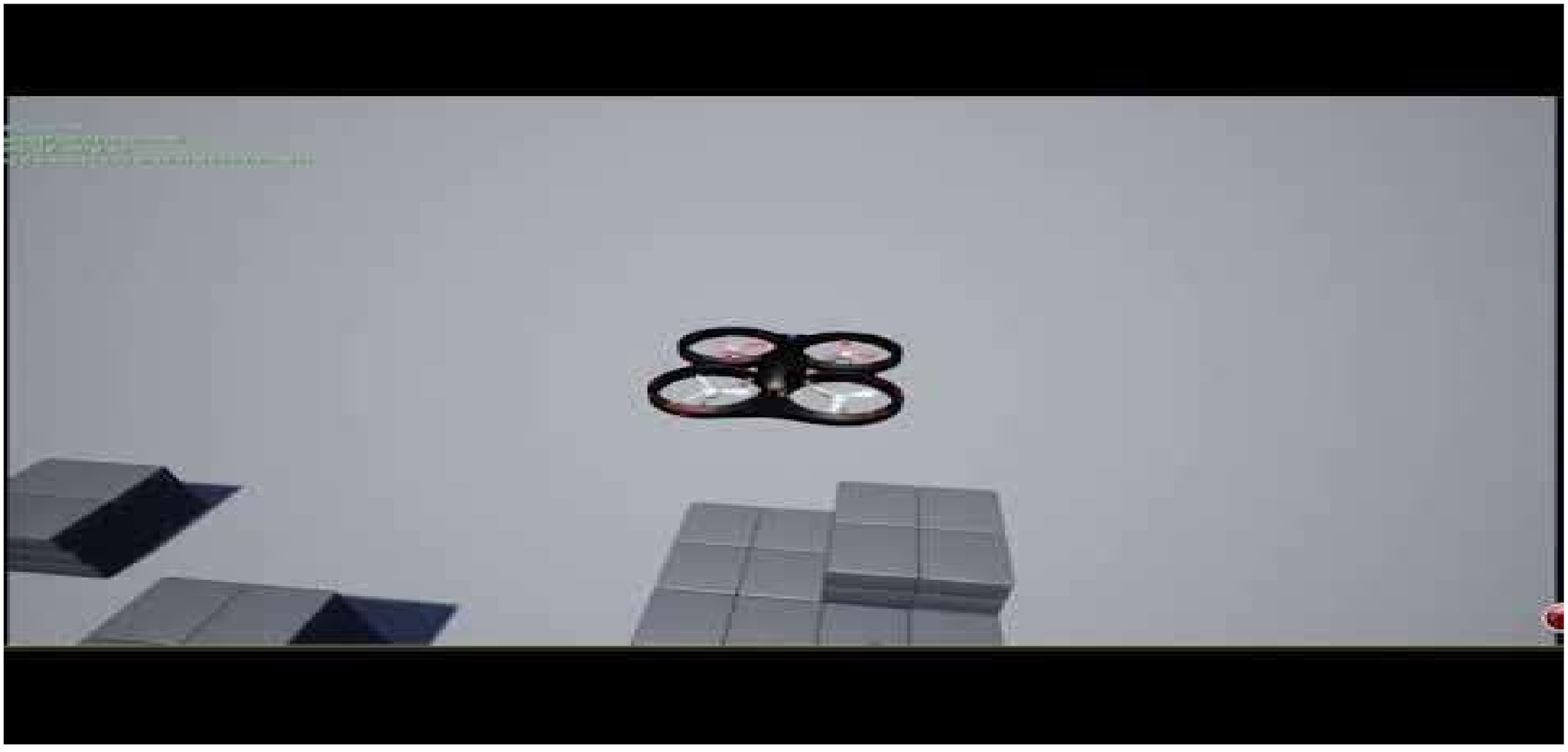












TD3 evaluation, $\Delta z = 5$ m

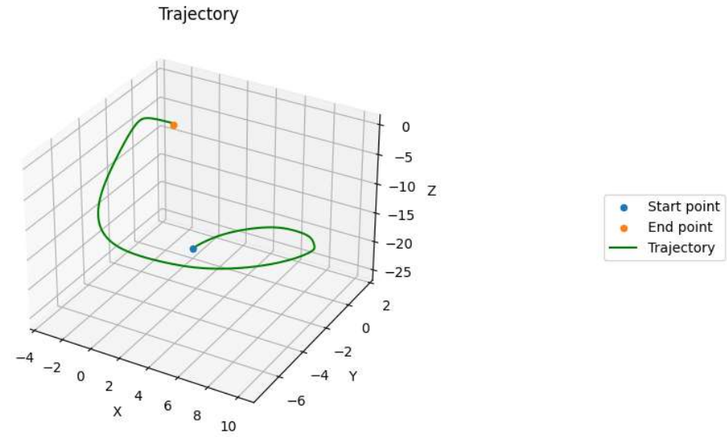
ANALYSIS

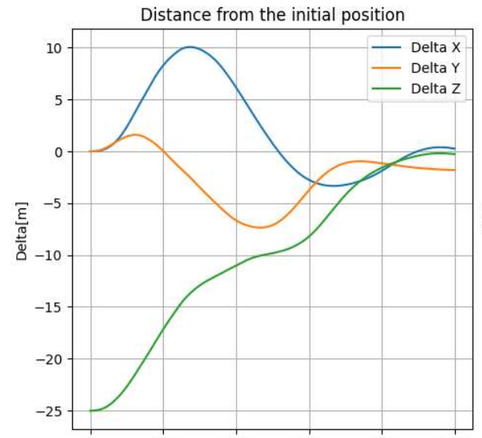
Initial parameters

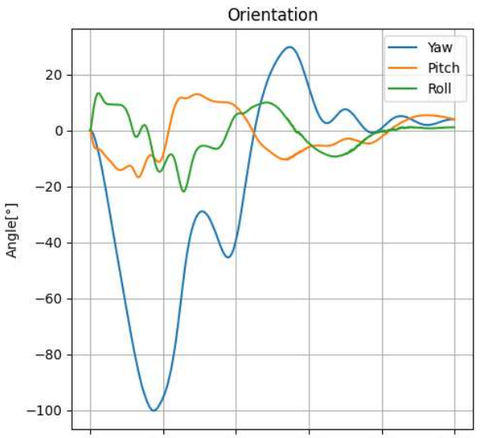
Δx = 0.0 m | Δy = 0.0 m | **Δz = 25.0 m**

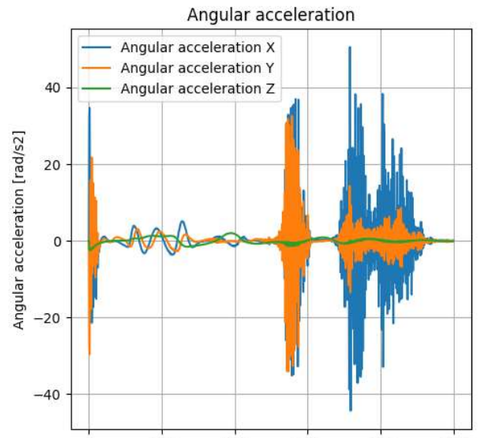
roll = 0 deg | pitch = 0 deg | yaw = 0 deg

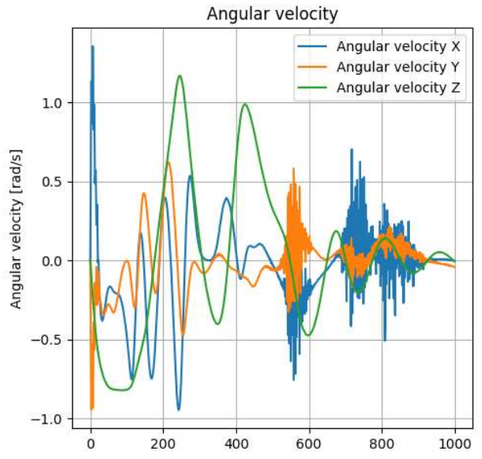
wind_x = 0.0 m/s | wind_y = 0.0 m/s | wind_z = 0.0 m/s

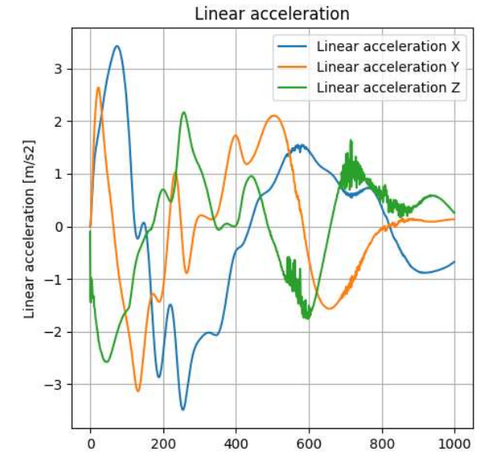


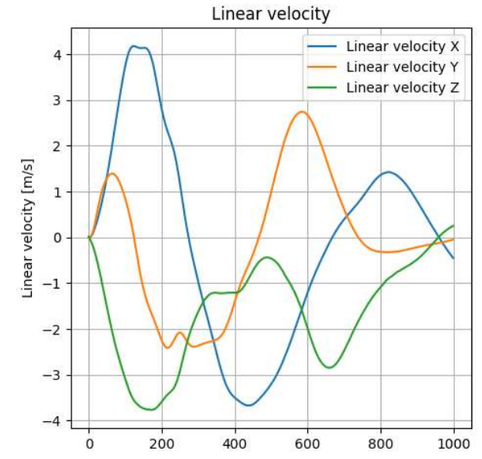












TD3 evaluation, $\Delta z = 25$ m

IV. LIMITATIONS

LIMITATIONS

High frequency components in the velocities and accelerations might hinder the stabilization. The issue might be solved:

- Implementing a LSTM architecture (able to better account for past information)
- Adding the previous action executed to the observation state or the reward function, to minimize high-frequency adjustments in the propellers thrust

LIMITATIONS

Steady position of the drone is not always accurate:

- This is partially expected, the reward focuses on orientation and angular velocity of the drone (that most influence stability)
- Might be attenuated via longer training or reward tuning

LIMITATIONS

Controller limited to certain distance from target set point:

- Limitation easily solvable by continuously updating the end position based on the current one, approaching the end-point in sequential steps

V. CONCLUSION

CONCLUSION

- The approach developed is more flexible than traditional PID controllers
- The new action encoding (scaled delta thrust) allows for finer-grained control, and wider compatibility with other drones
- The new networks architecture strikes a balance between learning speed and representational capability

CONCLUSION

- Completely off-policy training: no particular exploration policies are needed to converge
- Resistant to up to 5x stronger wind compared to previous works, and to 2x stronger wind compared to most commercial solutions
- Good performance with high deltas between initial pose and set pose
- Robust to extreme perturbations in initial pitch, roll and yaw angles: can recover from upside down states

The presented controller - with adequate real world testing - could play an important role in the stability recovery of a new generation of commercial solutions, where the safety of the individuals is a primary concern.

Solid solutions are at the heart of the development in the field, as technology progress must also constantly improve the reliability of its application.

THANK YOU FOR THE ATTENTION