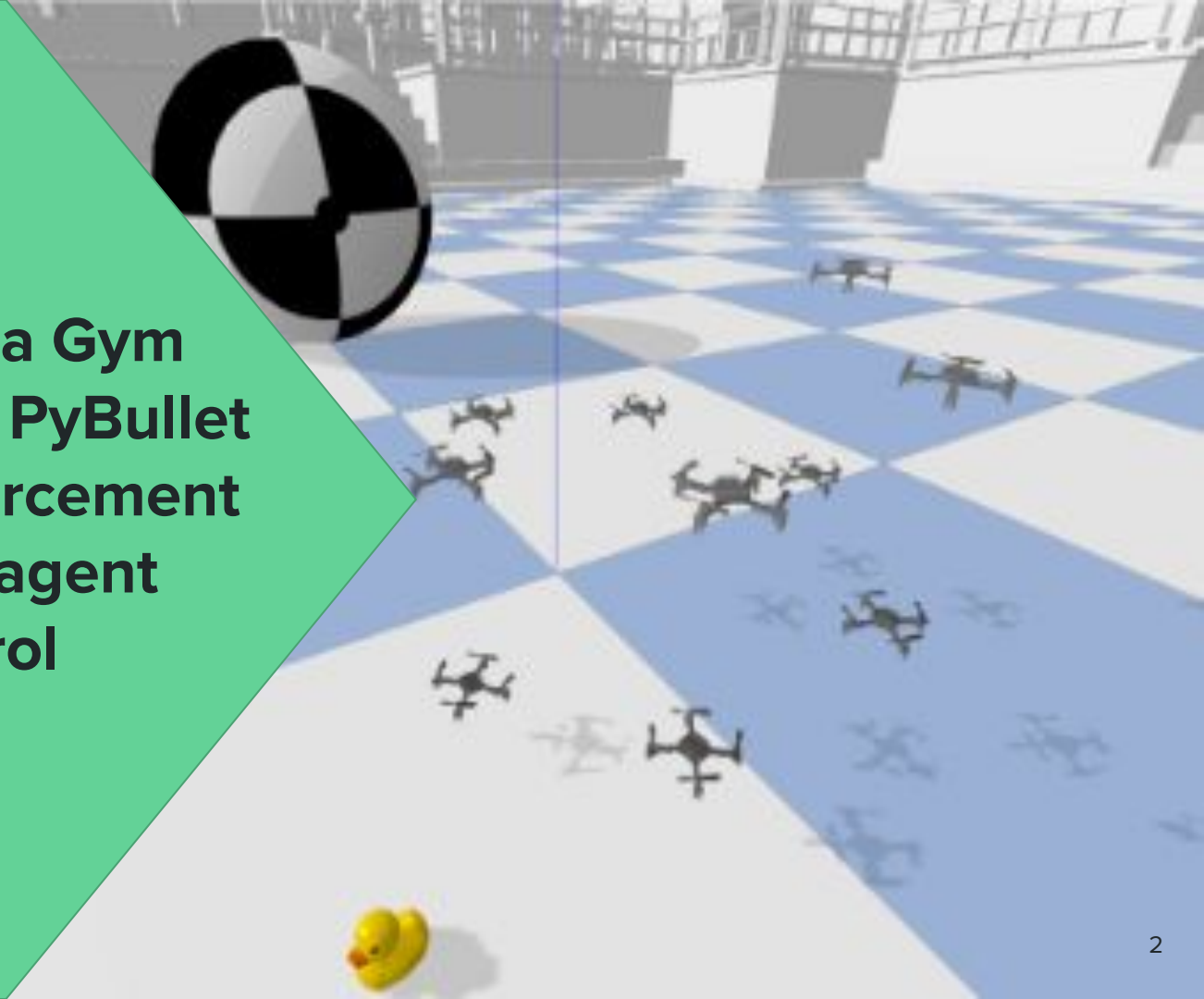
Topics in Intelligent Systems

Group E - Daniel Carneiro, Félix Martins, Gonçalo Costa

GitHub Repository: https://github.com/Minipoloalex/gym-pybullet-drones/tree/main

Paper

Learning to Fly – a Gym Environment with PyBullet Physics for Reinforcement Learning of Multi-agent Quadcopter Control



Environment Overview and Objectives

“Open-source OpenAI Gym-like environment for multiple quadcopters based on the Bullet physics engine”. Includes multi-agent and vision-based reinforcement learning interfaces.

The paper focuses on:

- realistic collisions and aerodynamic effects
- reinforcement learning flexibility
- examples with control theory methods and workflows using state-of-the-art reinforcement learning libraries

Reinforcement learning interfaces

Observation spaces

- Kinematics
- Adjacency Matrix of Multi-Robot Systems (neighbouring robots)
- Video frames from each drone’s perspective

Action spaces

- Propeller’s RPMs
- Desired velocity input (abstracting the stabilization task)

Other control modes require customizing observation and action spaces.

Paper Libraries

- **Stable Baselines3 Workflow**
  - It performs Quadcopter control tasks, such as takeoff and flight stabilization. It is known for providing efficient implementations of popular RL algorithms, such as:
    - **A2C** (Advantage Actor-Critic) which uses a critic to evaluate the actions of the agent and an actor to generate the actions.
    - **SAC** (Soft Actor-Critic) which maximizes the reward while promoting action exploration.
- **RLlib Workflow**
  - This facilitates the training and coordination of multiple drones.
  - It has the capability to:
    - configure multi-agent scenarios
    - associate specific policies with different drones
    - use a centralized critic for joint learning.

Examples and results

Control theory examples:

- **Trajectory tracking with PID control**: receive kinematics observations, return motors RPMs.
- **Desired velocity input**: receive kinematics, return velocity inputs.
- **Ground effect**: Compares takeoff of a quadcopter with and without ground effect. Ground effect refers to the aerodynamics interaction with the ground when taking off.
- **Downwash**: Tests how the downwash model affects drones on top of each other. Comparison is performed with real world tests. Downwash refers to the aerodynamics interaction with other drones when passing over them.

Reinforcement Learning examples:

- **Single Agent Takeoff and Hover**: Agent trained to hover at a given height.
- **Multi-Agent Leader-follower**: Leader agent trained as above and follower rewarded by tracking leader's altitude.

Project Tasks

1. **Work results reproduction**
   a. Find correct library versions
   b. Test the examples and verify the results
   c. Create and adapt plots to display the results

2. **Modifications**
   a. New multiagent systems tasks/environments (with libraries from paper)
      i. All drones meeting at a coordinated height
      ii. Two drones chase each other
      iii. Drones hover at arbitrary positions (unsuccessful)
   b. New library workflow implementation - Tianshou (unsuccessful)

3. **Comparison**
   a. Compare and interpret results obtained from each of the tasks and libraries

Work results reproduction

The paper is from 2021 and the library versions were not specified. The plots generated were also not automatically available.

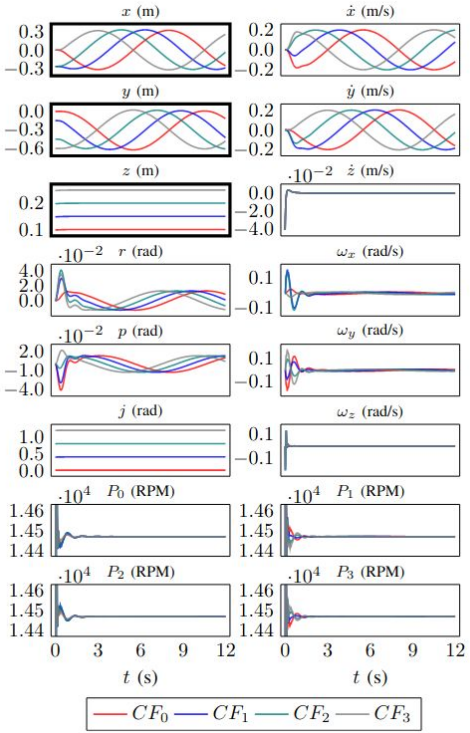
Requirements for work results reproduction:

- Identify correct **library versions** by performing multiple tests
- Adapting and **creating plots** shown on the paper

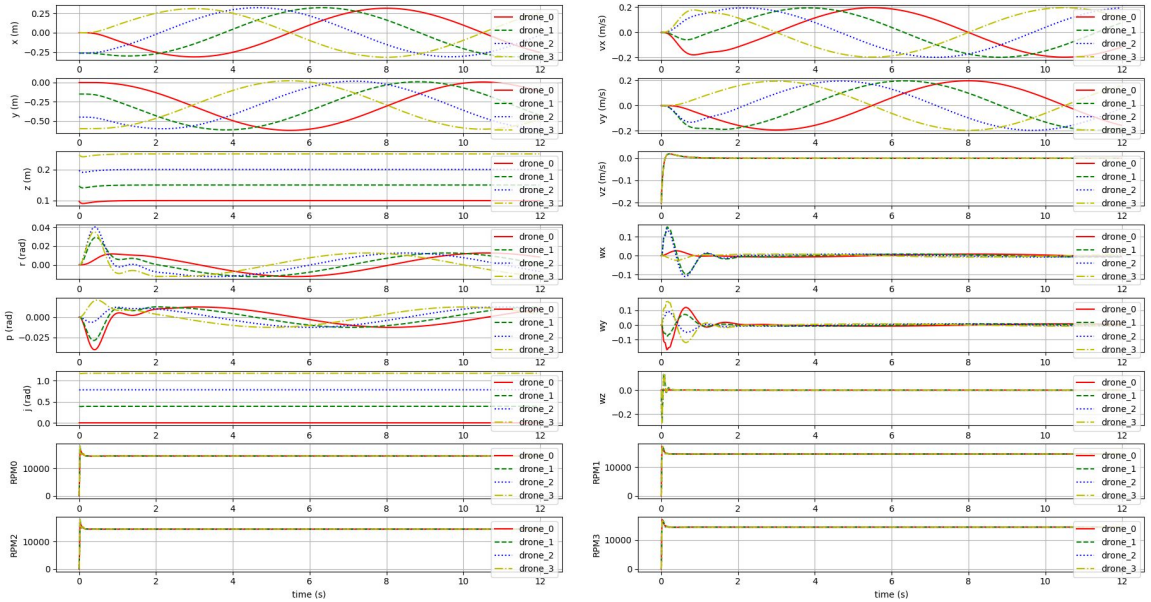
Some examples from the paper are not multiagent. These were also reproduced and mentioned, but the focus is on multiagent systems.

Work results reproduction - fly

Paper results

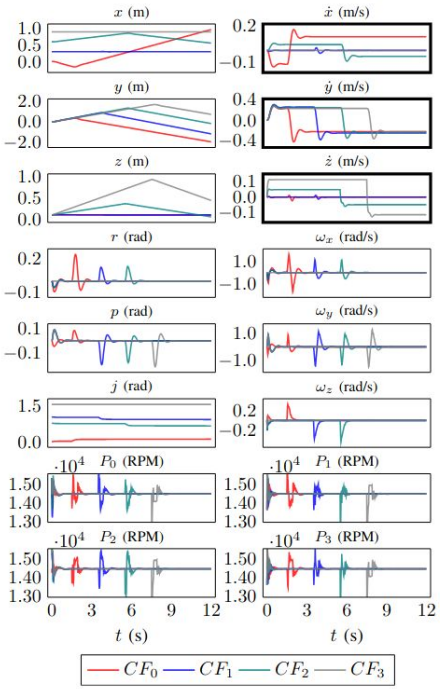


Our results

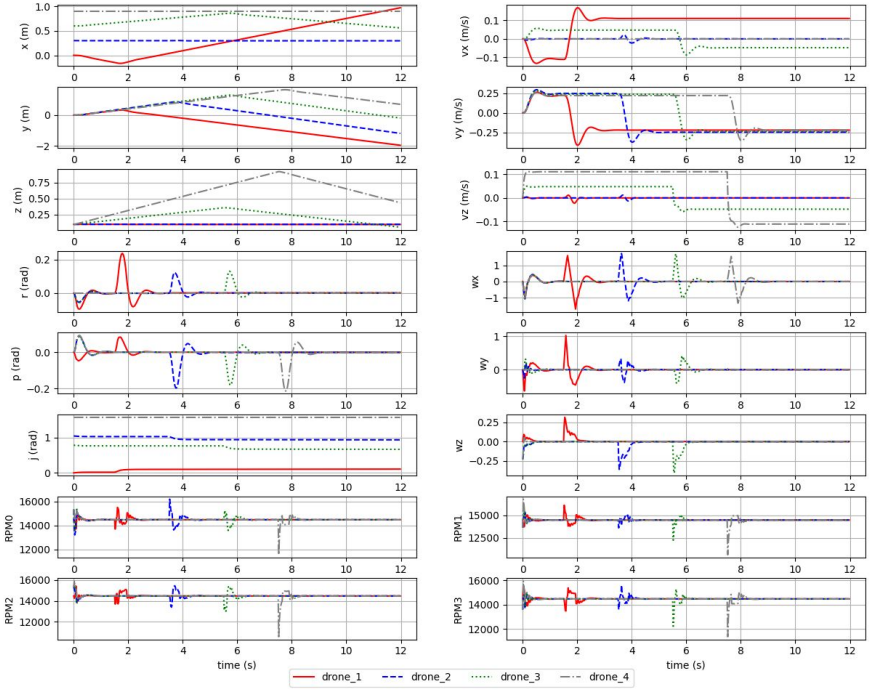


Work results reproduction - velocity

Paper results

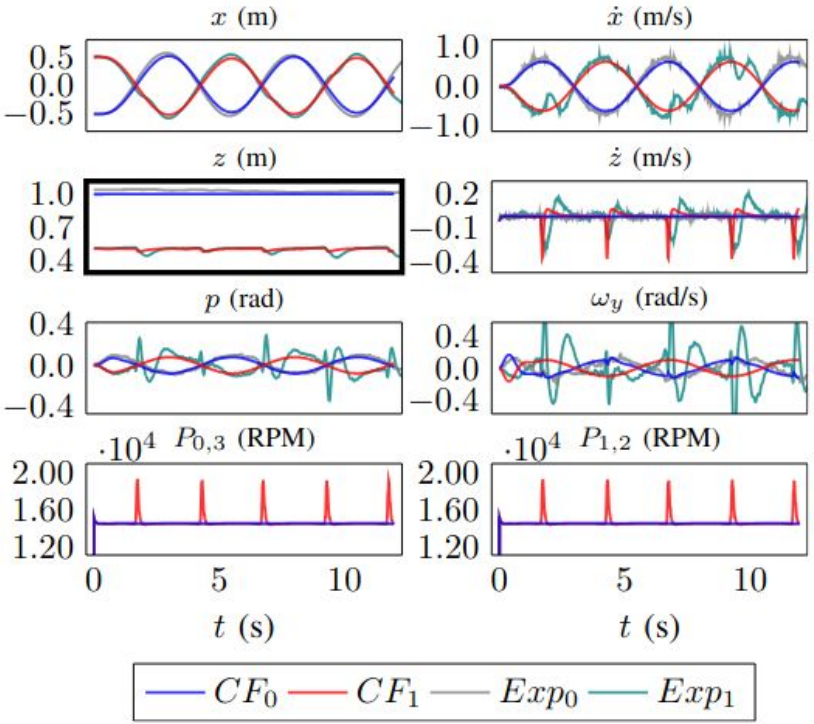


Our results

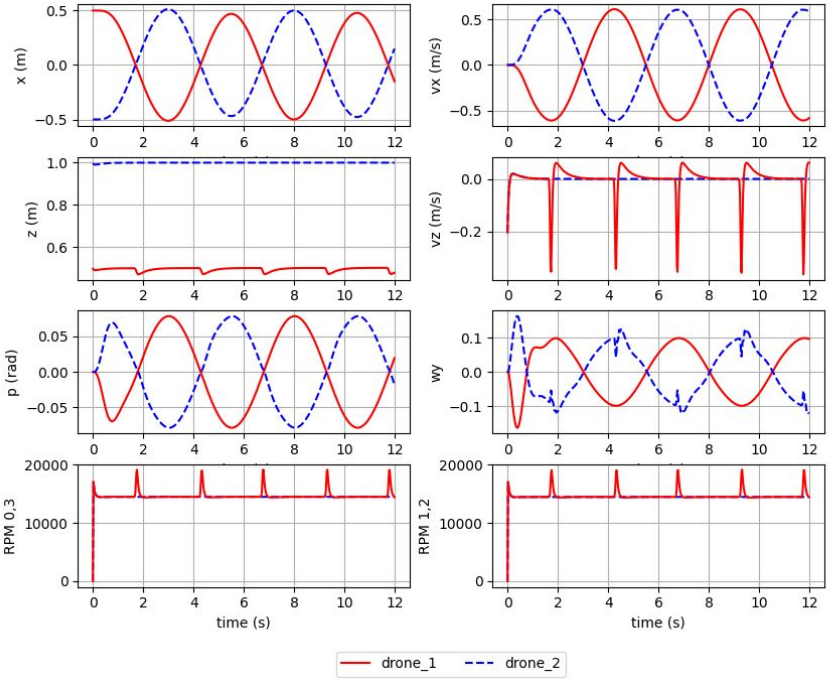


Work results reproduction - downwash

Paper results

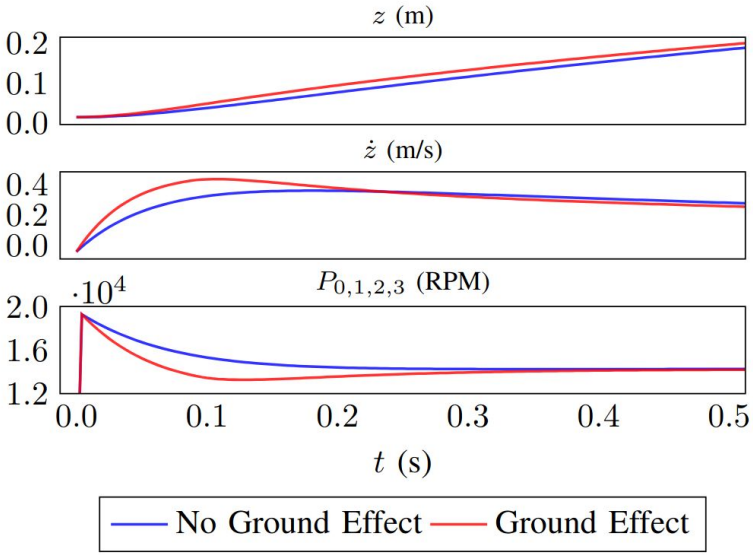


Our results

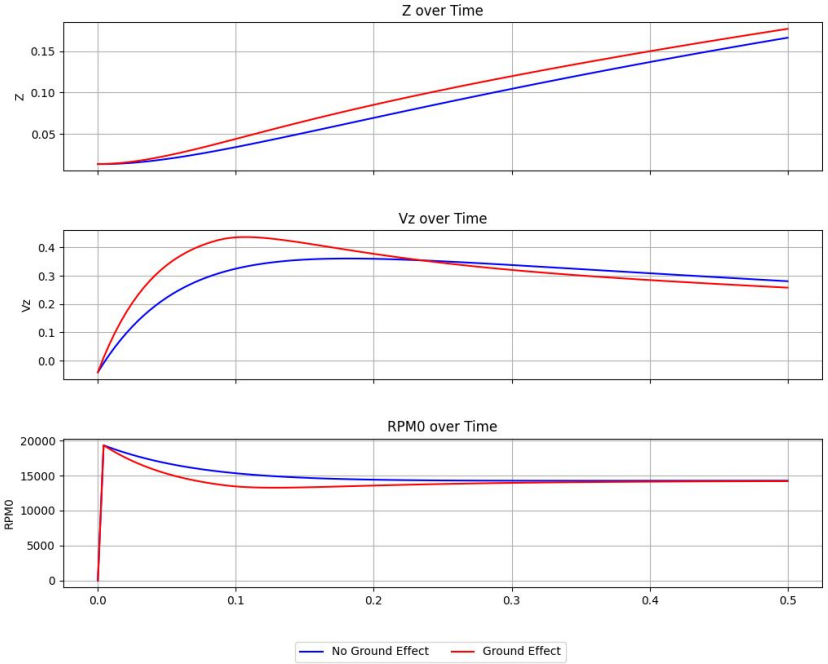


Work results reproduction - groundeffect

Paper results

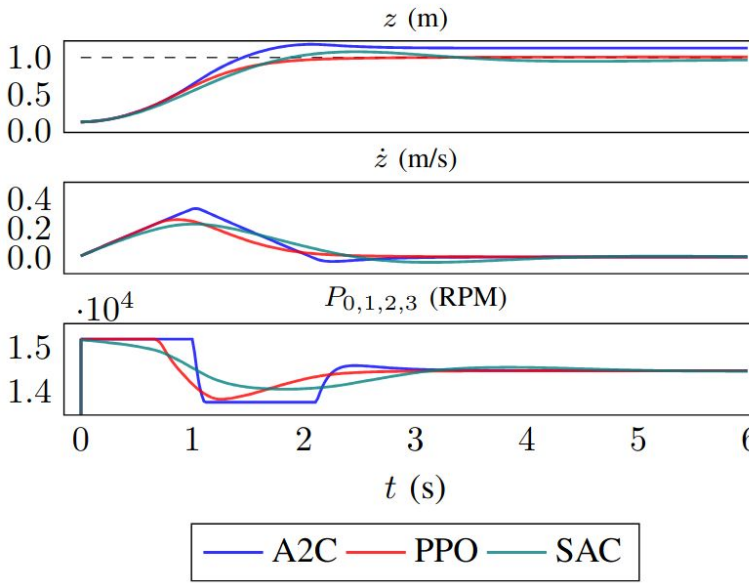


Our results

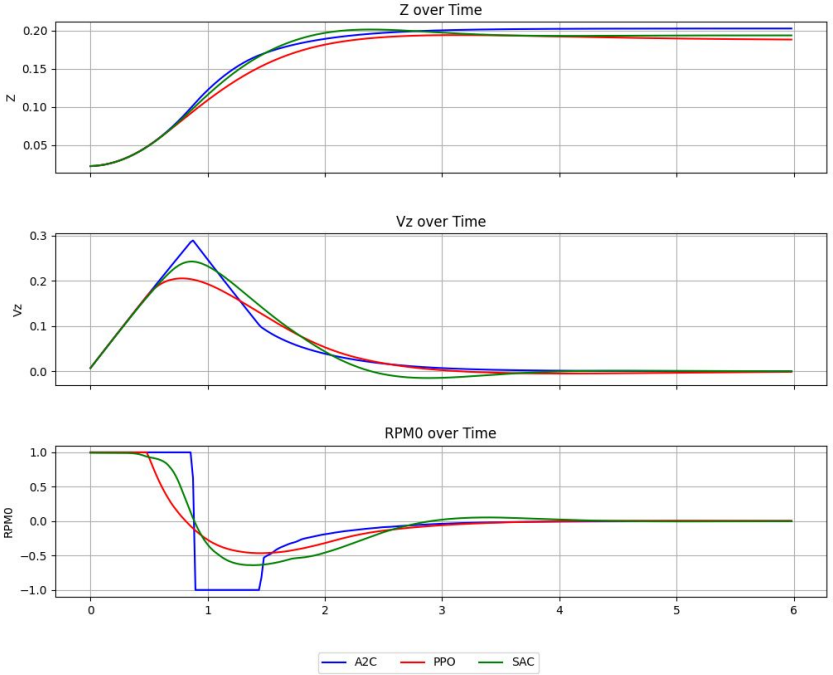


Work results reproduction - Single Agent RL

Paper results



Our results



Work results reproduction - Multiagent Leader Follower

Observation Space

- Kinematics (position, linear velocity, angle, angular velocity, quaternions)

Action Space

- 1D RPM (one motor speed for all motors)

Reward Functions

- **Leader**: hover at a given height ➔ point (0, 0, 0.5)

$$R_0(t) = -1 \cdot (X_0^2 + Y_0^2 + (Z_0 - 0.5)^2)$$

- **Follower**: track the leader's height ➔ follow leader

$$\forall j \in \text{drones} \setminus \{0\}, \quad R_j(t) = -\frac{1}{|drones|} \cdot ((X_j - X_0)^2 + (Y_j - Y_0)^2 + (Z_j - Z_0)^2)$$

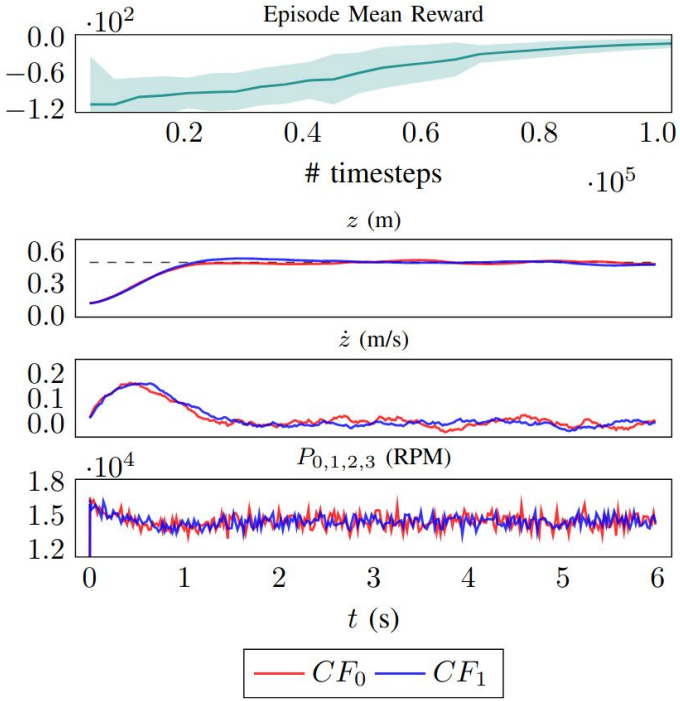
Work results reproduction - Multiagent Leader Follower

Notes on the follower's learning

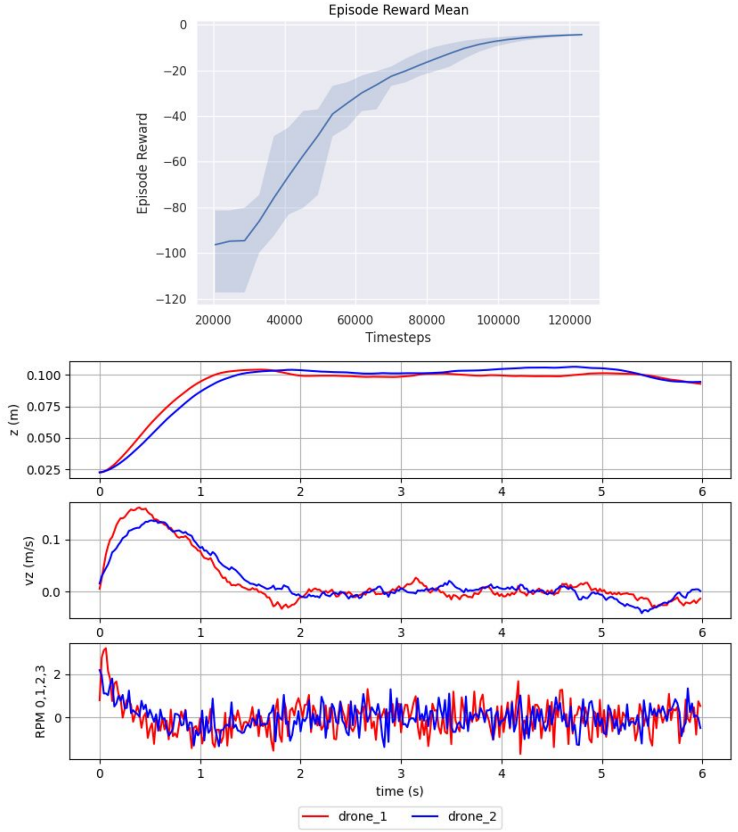
- Since the action space is 1D RPM, the X and Y cannot be changed, making that part of the reward constant. Therefore, the drones learn to follow the given Z coordinate.
- Since the follower's observation space does not integrate any information from the leader, the follower will learn the leader's trajectory (which is always the same, as it is always following the same point), instead of following the leader. To follow an arbitrary drone, the specific position of that drone should be included in the follower's observation space.

Work results reproduction Leader Follower

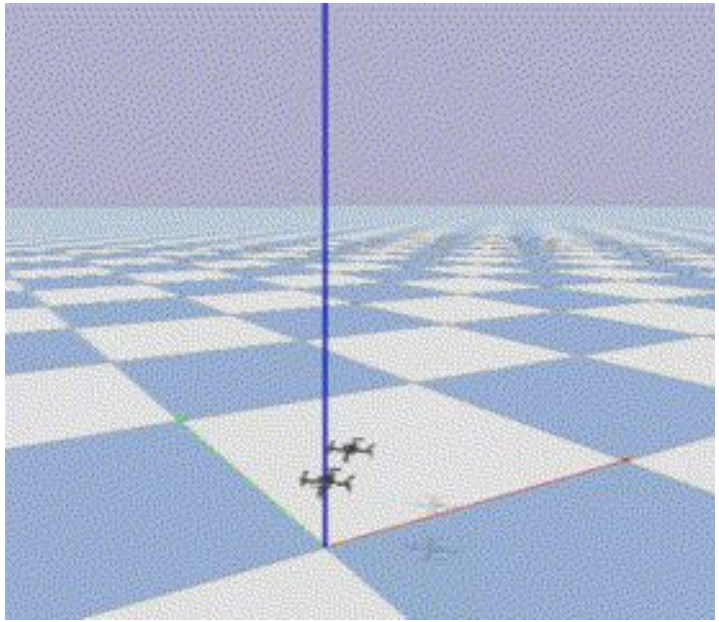
Paper results



Our results



Work results reproduction - Leader Follower



https://github.com/Minipoloalex/gym-pybullet-drones/blob/main/files/videos/leaderfollower.gif

Modifications - New Environments and Tasks

Meet at a height (Cooperative Env.)

- Several drones meet at a coordinated height

Chase (Competitive Env.)

- One drone tries to distance itself
- The other tries to close the distance

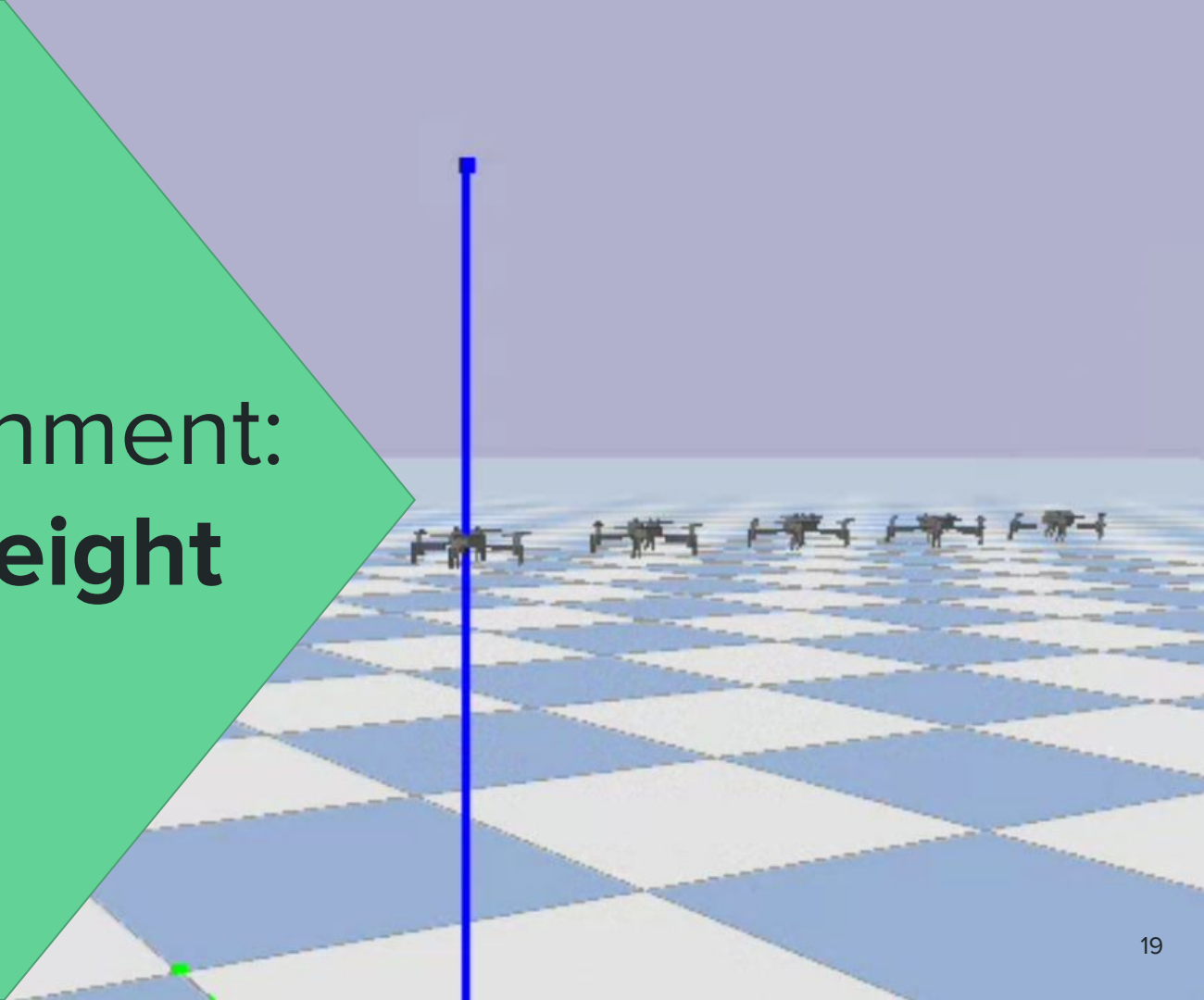
Figure (Cooperative Env.)

- Several drones move and attempt to hover at different positions
- Very complex due to 4D action space
- Unsuccessful environment

Use of new libraries (Leader Follower)

- Tianshou
- Sample-Factory

New Environment:
Meet at a height



New Environment - Meet at a height

Objective: drones coordinate and move to the same height

Observation Space (only requires 3 values)

- Z coordinate
- Z velocity
- average Z coordinate for all drones

Action Space

- 1D RPM: provide one value of motor speed to all motors (abstracts from drone stabilization)

Reward Function

$$\forall i \in drones, R_i(t) = -1 * \left(\frac{\sum_{j \in drones} Z_j(t)}{|drones|} - Z_i(t) \right)^2$$

New Environment - Meet at a height

Problem: Reward function only makes drones follow average height. The easiest way to get all drones with the same height is to have all drones land and have `Z = 0`.

Improvement: define a minimum height to make the drones fly and “work for it”.

New Reward Function

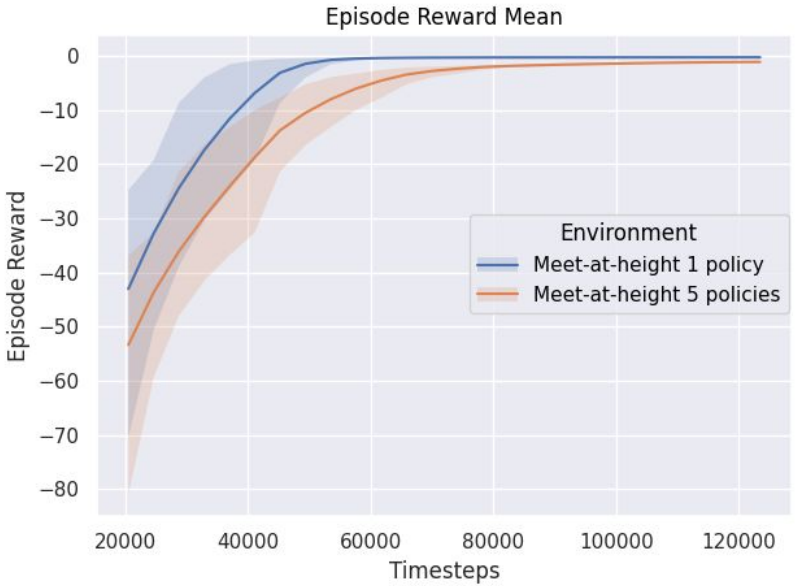
$$\forall i \in drones, R_i(t) = -1 * \left(max\left(\frac{\sum_{j \in drones} Z_j(t)}{|drones|}, MIN\_HEIGHT \right) - Z_i(t) \right)^2$$

Observation: Given the same observation space (input), all drones should perform the same way (output). This means we can use the same policy for all drones.

Improvement: learning can be performed by multiple drones on the same policy.

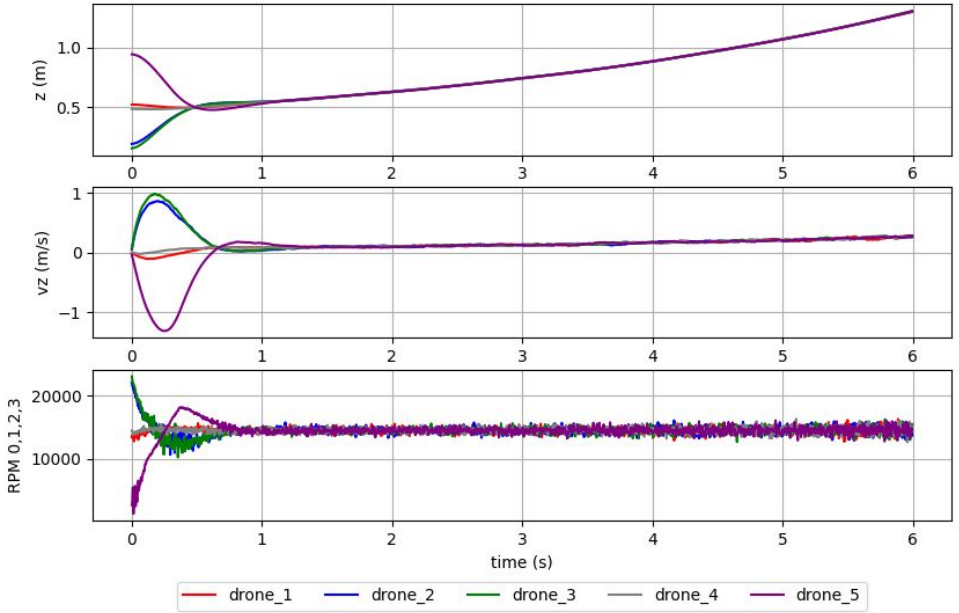
Learning Improvement - Meet at a height

- Rapid reward convergence for both cases.
- Improvement on how fast the agents learn when learning just one policy.
- For more complex tasks, this may be very useful.

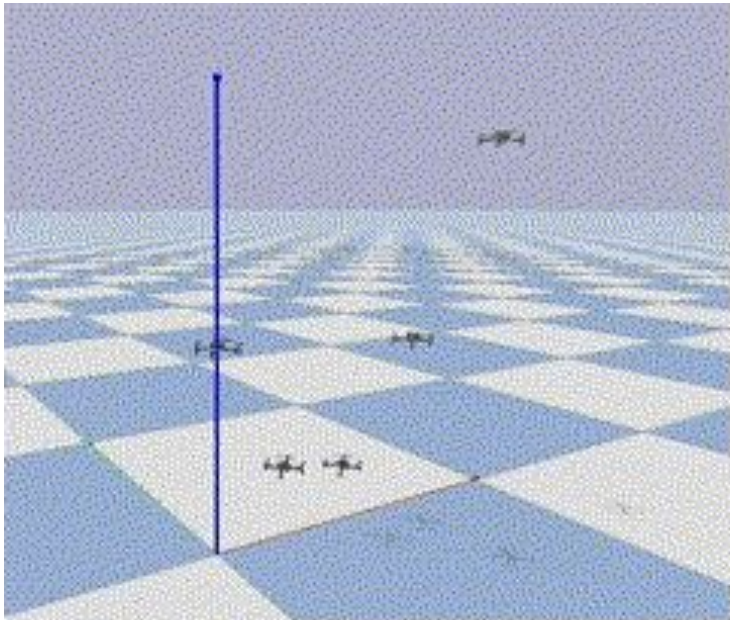


Results - Meet at a height

- Drones respect minimum height defined.
- No attempt to reduce velocity and stay on the same position: drones may move as a group.
- Drones rapidly get to the same height.



Results - Meet at a height

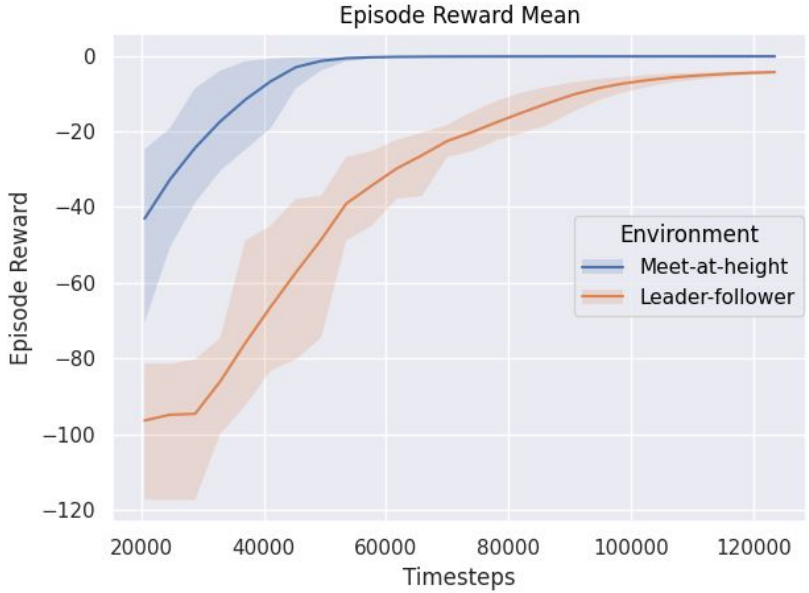


https://github.com/Minipoloalex/gym-pybullet-drones/blob/main/files/videos/meet_at_height.gif

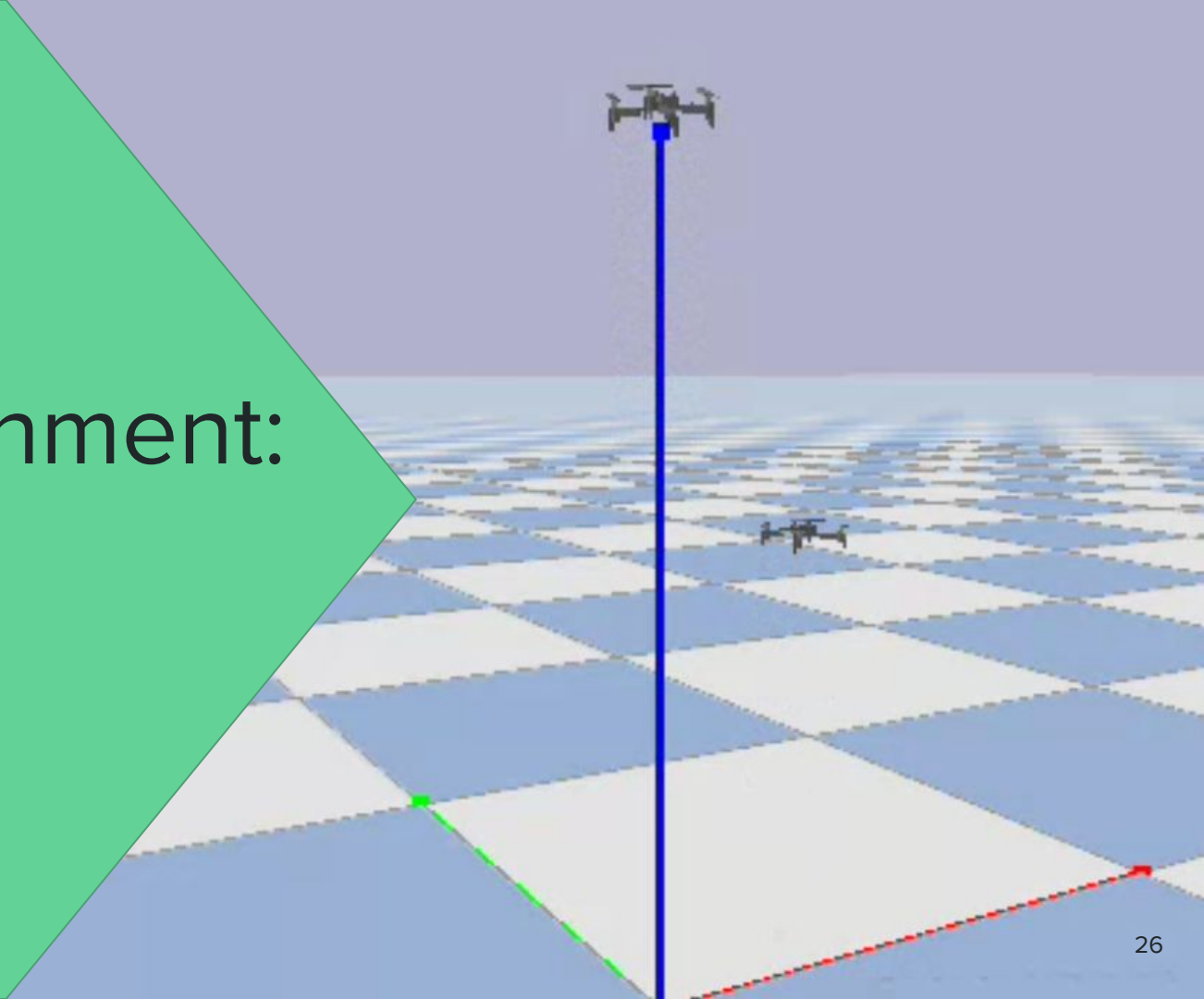
Comparisons - Leader-follower vs Meet at a height

Leader follower is slower to learn:

- The leader must first learn to hover at the given point, only after the follower will learn properly.
- In meet at a height, all drones immediately start to learn correctly.
- Using one policy only also improves the speed of convergence.



New Environment: **Chase**



New Environment - Chase

Objective: one drone distances itself while the other closes the distance

Note: only allows 2 drones!

Observation Space (only requires 3 values)

- Z coordinate
- Z velocity
- opponent's Z coordinate

The opponent's Z velocity is not given, since it made the drones much faster to converge in one position.

Action Space

- 1D RPM: 4D is too complex for the drones to learn it well

New Environment - Chase

Reward Functions

- Runner: Maximize distance to Chaser, while inside the boundaries: Z ∈ [0.1, 1]

$$R_0(t) = \min((Z_0 - Z_1)^2, MAX\_DIST\_SQ) + penalty(Z_0)$$

- Chaser: Minimize distance to Runner, while inside the boundaries: Z ∈ [0.1, 1]

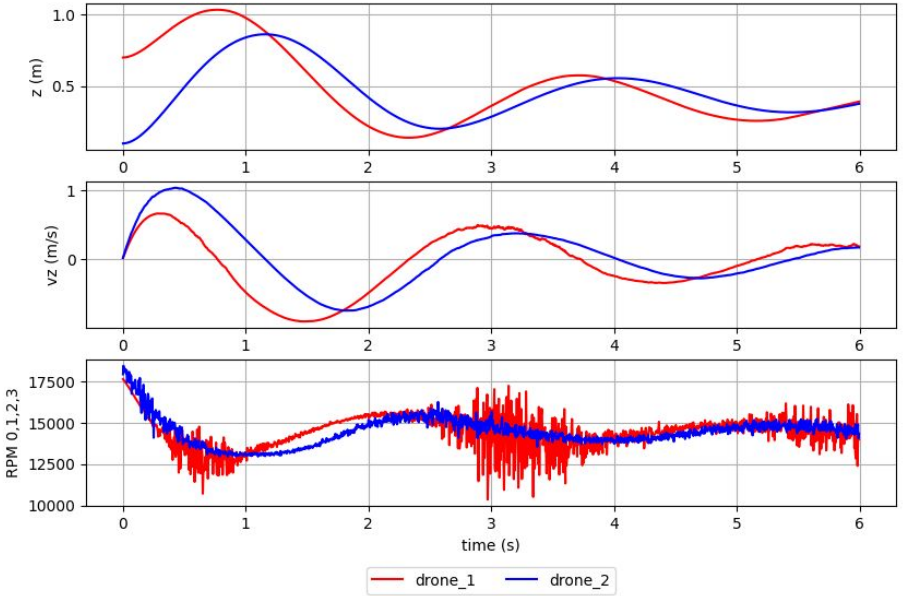
$$R_1(t) = -\min((Z_0 - Z_1)^2, MAX\_DIST\_SQ) + penalty(Z_1)$$

where: $penalty(Z) = \begin{cases} 0, & \text{if } 0.1 \leq Z \leq 1 \\ -10, & \text{otherwise} \end{cases}$

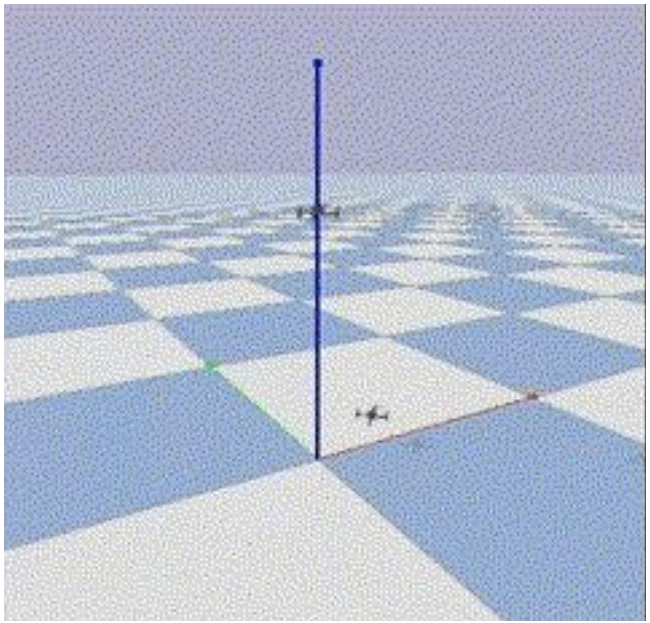
Note: The minimum for the Z distance must be used, otherwise the first drone (the runner) would “get out of the screen” and leave the defined boundaries to maximize that distance.

Results - Chase

- Eventually, the chaser will “catch” the runner and the height will stabilize.
- The runner wants to maximize the distance, but it learns that, after some time, it cannot get much distance from the other.
- Without the explicit knowledge of the opponent’s velocity, the chaser initially “overshoots” the position of the runner and oscillates before converging to exact position.



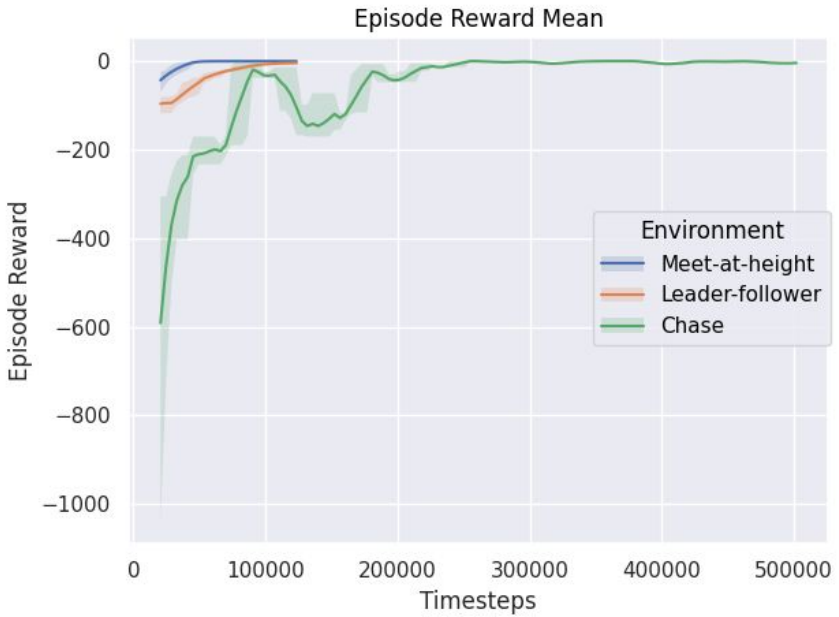
Chase Environment - Results



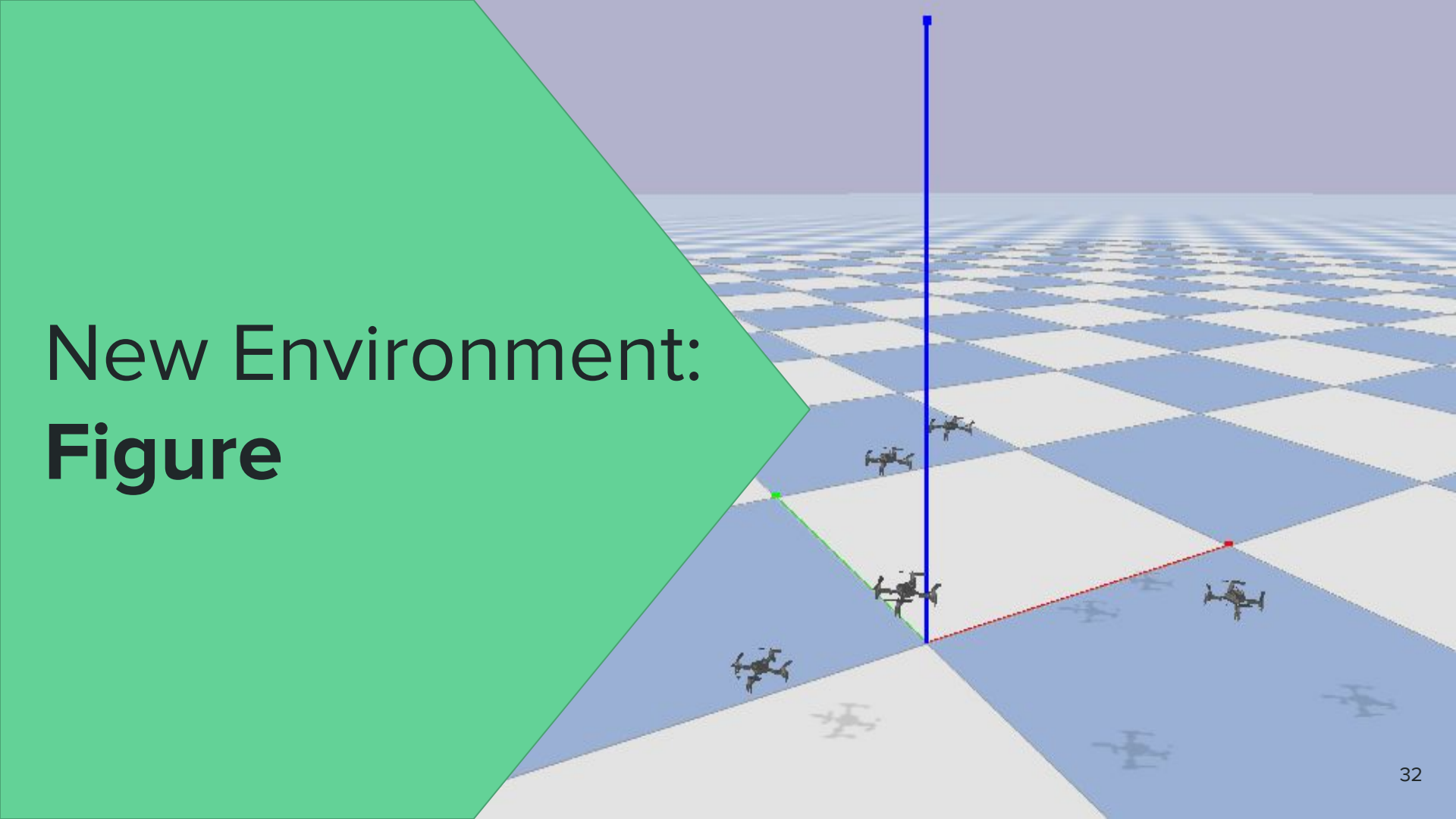
https://github.com/Minipoloalex/gym-pybullet-drones/blob/main/files/videos/chase.gif

Comparisons - Chase vs Other Environments

- Initial reward values are exacerbated due to the punishment of drones outside the desired Z range.
- Takes longer to converge since there are 2 agents learning and adapting competing strategies at the same time.
- Suffers from decreasing rewards, which does not happen in other environments. Here, as the second agent learns to follow the first, the first likely attempts to go outside Z range before understanding what it should actually do.



New Environment: **Figure**



New Environment - Figure

Objective: Several drones move to designated positions and hover on these positions

Observation Space

- Kinematics of drone (position, linear velocity, angle, angular velocity, quaternions)
- Target's position
- Other drones' positions

Action Space

- 4D RPM: one RPM value per motor

Note: All drones can share the same policy, since given the same information, they should behave the same way. If we remove the position of the target in the observation space, then different policies would be required.

New Environment - Figure - Initial Reward Function

$$\forall i \in \text{drones},$$
$$R_i(t) = -1 * distance(pos[i], target[i]) + \sum_{j \in \text{drones} \setminus \{i\}} min(0, distance(pos[i], pos[j]) - MIN\_DISTANCE) * \frac{1}{MIN\_DISTANCE}$$

Focus on:

1. Minimizing the distance to the designated target,
2. While not going below the minimum distance to other drones.
   - If the drones are far enough away, their reward is 0.
   - If they are too close, they get punished (progressively) with negative rewards.

The reward function is naive, and the drones do not learn a correct policy.

New Environment - Figure - Attempts and Difficulties

Problem: the drones were not learning desired behaviour (hovering at the target position)

Possible Solutions (attempted and failed):

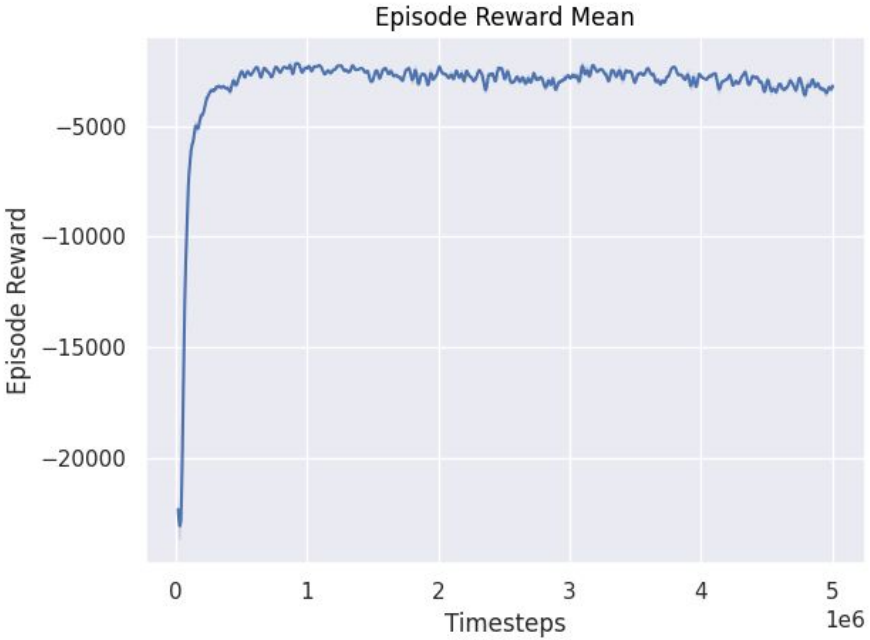
- Simplify the environment
  - initial position = target position
  - objective: hover at the initial position
- Change reward function
  - Simplification
    - objective: minimize the distance to the target
    - result: drones throw themselves to the ground as an easy way to get close to the target
  - Punishment for being too close to the ground
    - result: leads them to fly upwards as they are too “afraid” of the ground
  - Punishing not just low altitudes, but also high altitudes
    - results: still unable to learn the policy, the drones either dive to the ground or launch into the air, confused by the complexity of controlling a flying drone
- Train for longer (increase total time steps for training) ➔ confirm that the policy is not learnt even if given more samples
- Change the model size and configuration values

Conclusions: the combined observation and action space is too complex for the agents to learn correctly.

New Environment - Figure - Unsuccessful results

These are the results for the initial version of the Figure environment, with 5 drones:

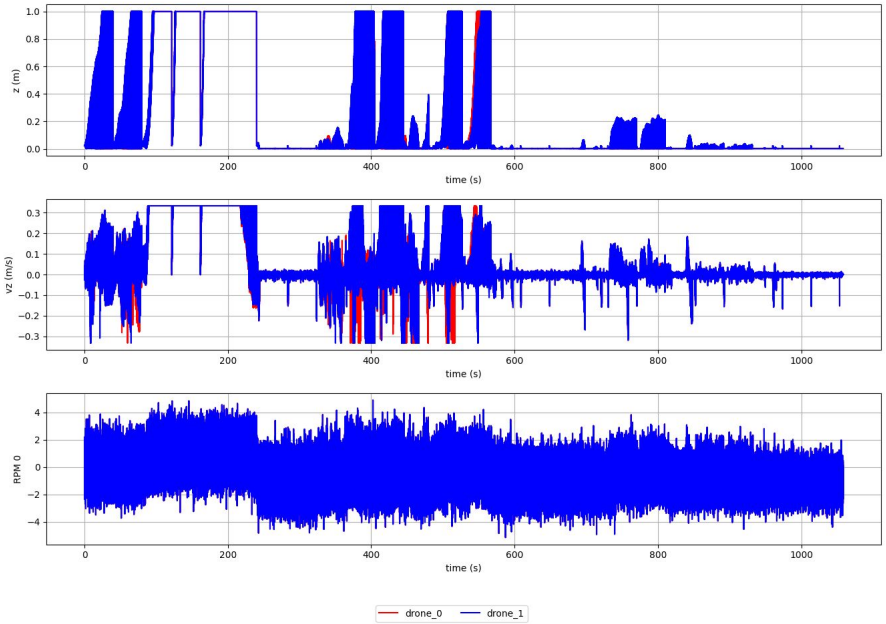
- Shows that controlling the drones using 4D RPM and MARL is very complex and difficult.
- Further reward shaping is required for the drones to even be able to hover at a position (with 4D action space).
- Getting a successful policy for this environment proved to be infeasible in our timeframe.

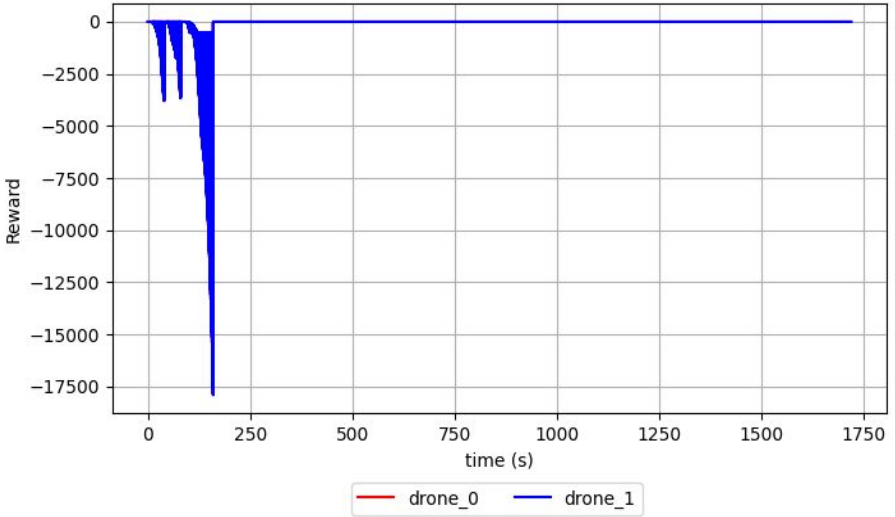


New Framework: **Tianshou**

Results - Tianshou

Results during training

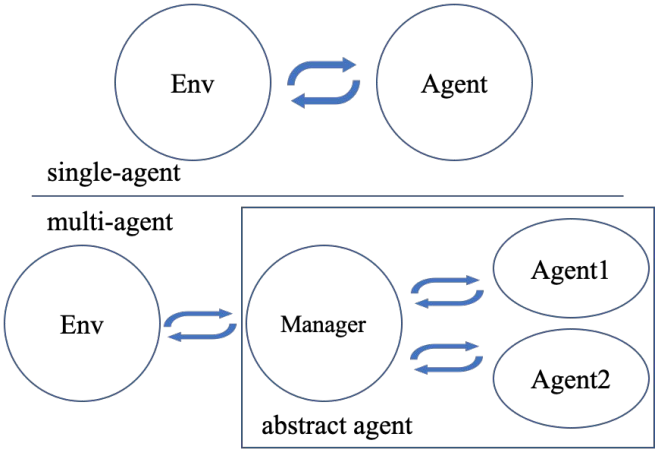




Problems - Tianshou

Note: the MultiAgentPolicyManager is shown in the example in a turn based MARL scenario with 1 agent in learning. We could not adapt it to work well in a simultaneous MARL scenario with 2 agents in the learning phase.

- To create multiagent environments, we used the `MultiAgentPolicyManager` class. However, the agents would be interacting alone in the environment (with each agent in its own environment).
- With both agents isolated, the actions taken by them are independent and did not influence the other's observations or rewards, with the best reward being around -140.
- Therefore, without interacting with the leader, the follower is unable to learn (as we previously explained in the comparison between the Meet at a height and Leader-follower environments).



References

- Panerati, Jacopo & Zheng, Hehui & Zhou, Siqi & Xu, James & Prorok, Amanda & Schoellig, Angela. (2021). Learning to Fly—a Gym Environment with PyBullet Physics for Reinforcement Learning of Multi-agent Quadcopter Control. 7512-7519. 10.1109/IROS51168.2021.9635857.
- MARLlib Documentation. (n.d.). Retrieved from https://marllib.readthedocs.io/en/latest/
- Replicable Multi-Agent RL (MARLlib). (n.d.). GitHub repository. Retrieved from https://github.com/Replicable-MARL/MARLlib
- Tianshou, A Reinforcement Learning Library for Researchers. (n.d.). GitHub repository. Retrieved from https://github.com/thu-ml/tianshou