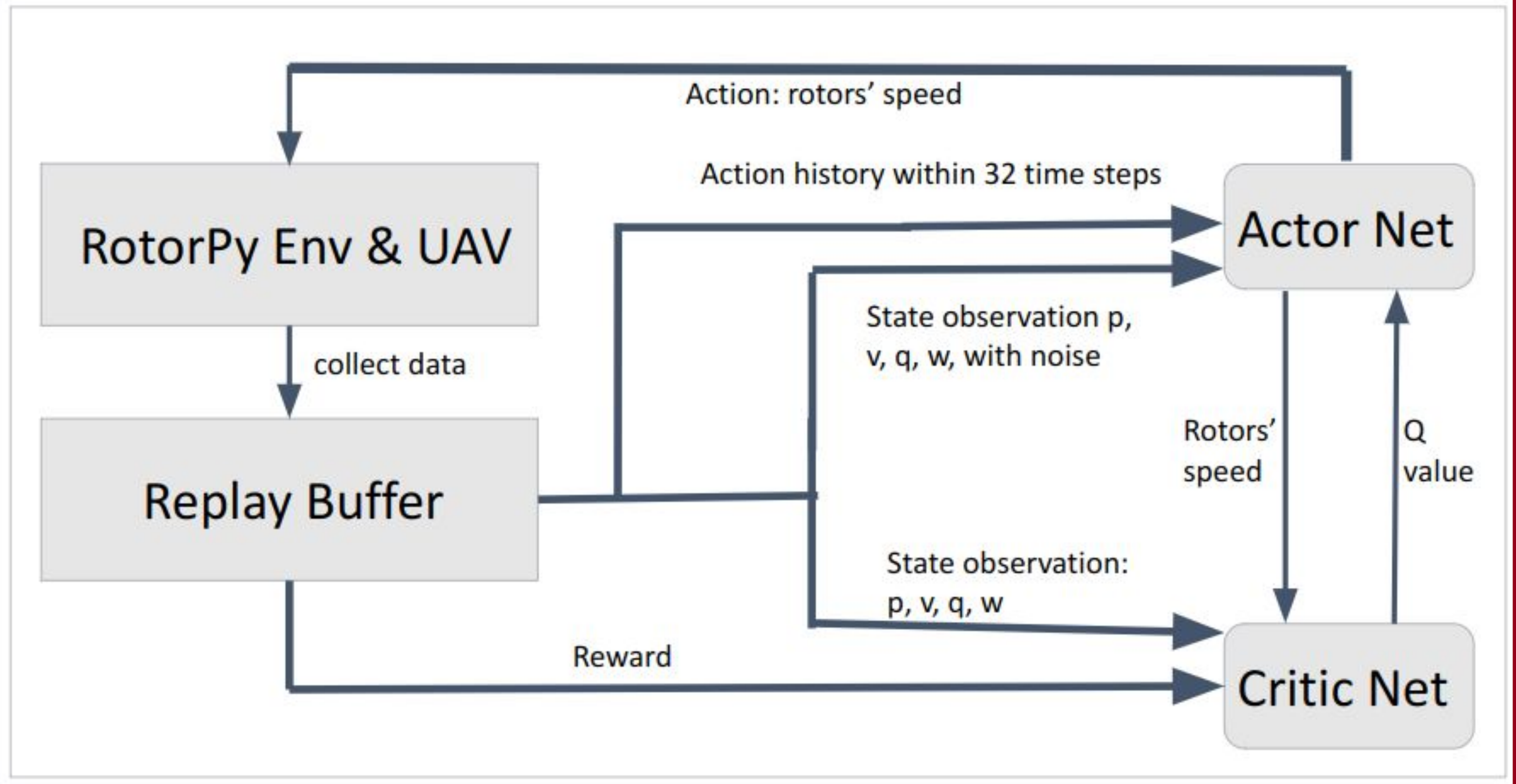
Learning Robust UAV Hovering in Wind-Disturbed Environments

Songhao Huang, Bowen Yang

INTRODUCTION/MOTIVATION

Hovering is essential for UAV applications like delivery and disaster response, where stability in unpredictable conditions is crucial. This challenge involves the complexity of end-to-end learning, adapting to varying winds, and stability across diverse initial conditions. We used a reinforcement learning approach to train an end-to-end policy for UAV hovering under wind disturbances. Compared to traditional methods, our policy achieves robust performance across varying winds and initial conditions.



METHODS

1. Simulation Environment

We used RotorPy [1], a Python-based multirotor simulation with aerodynamic wrenches. The state **observation** includes position, velocity, quaternion, and angular velocity, while the control **action** is defined as four rotor speeds.

We defined the **reward** function as the weighted sum of: current position error, current velocity error, the error of constant in quaternion, survival

METHODS Continued

reward, and goal reaching reward.

2. Architecture

Inspired by [2][3], we utilized an off-policy RL algorithm (TD3) with Actor and Critic networks, each comprising a 3-layer structure with 64 neurons per hidden layer.

3. Training Details

3.1 **Guidance**: We randomly initialize all the UAV state within a reasonable range, occasionally starting it in a hover position to enhance training.

3.2 **Random noise** is added to observations to improve policy robustness against wind variability.

3.3 **Action history** is fed into the policy to accelerate motor response, modeled as a first-order low-pass filter.

3.4 **Curriculum learning** adjusts reward weights and guidance probabilities to optimize training progression.

RESULTS

1. **Training Time**

   Our policy was trained on a laptop with a RTX4060 GPU for **40 mins** (~700000 iteration)

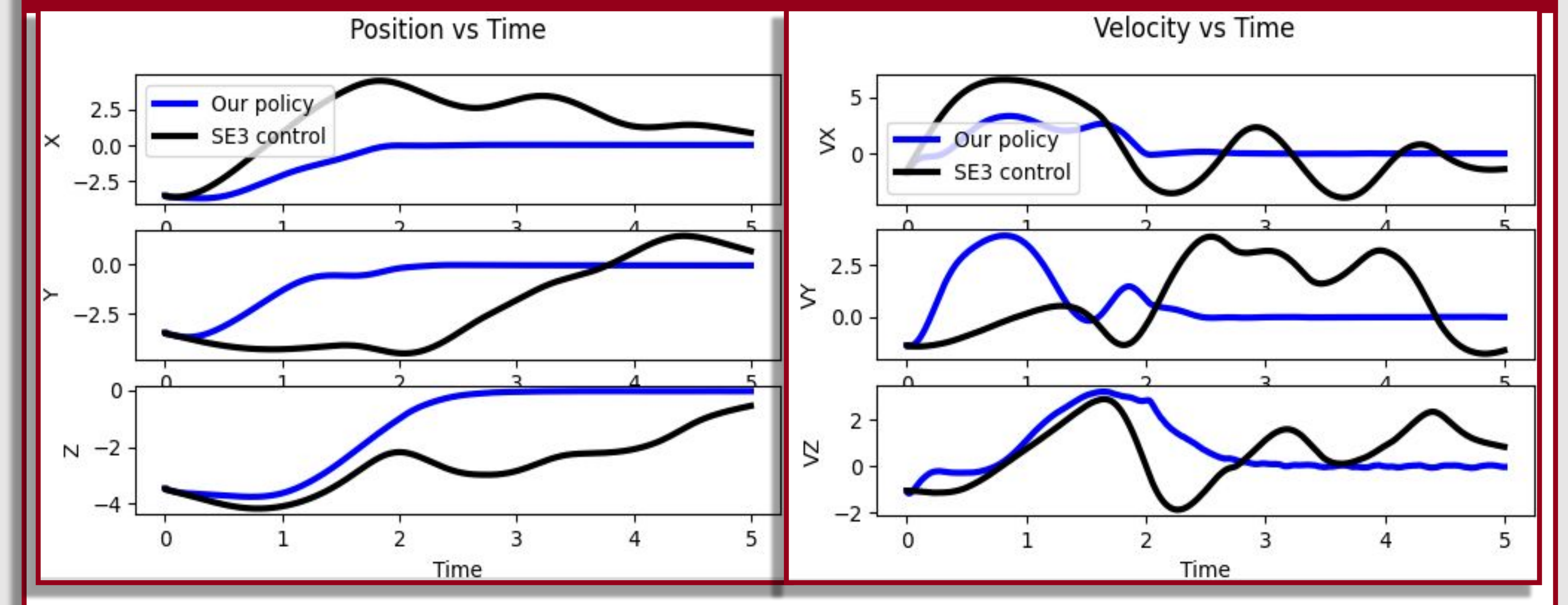
2. **Simulation Benchmark**

   We compare the performance with SE3 controller[4]. The following are corresponding position/velocity vs. time plots in **one test**, with the same adverse random initial conditions and constant wind disturbance.

3. **More Experiments**

   The table shows the performance for two controllers testing for **100 times** in each case.

RESULTS Continued



Constant wind	Mild			Medium			Fierce		
	Success rate	Reaching time	Control cost	success rate	Reaching time	control cost	success rate	Reaching time	Control cost
Ours	100 %	**2.57 s**	**222.99**	**100 %**	3.01 s	**268.02**	**57 %**	**2.99 s**	**249.66**
SE3 ctrl	100 %	3.29 s	307.92	21 %	3.38 s	305.01	2 %	3.9 s	313.34
Sine wind	Mild			Medium			Fierce		
	Success rate	Reaching time	Control cost	success rate	Reaching time	control cost	success rate	Reaching time	Control cost
Ours	100 %	**2.59 s**	**223.37**	**98 %**	**2.99 s**	**256.85**	**73 %**	**3.65 s**	**310.62**
SE3 ctrl	100 %	3.25 s	302.49	79 %	3.66 s	341.29	21 %	4.1 s	382
Initial Condition	Ideal			Challenge			Adverse		
	Success rate	Reaching time	Control cost	success rate	Reaching time	control cost	success rate	Reaching time	Control cost
Ours	100 %	1.67 s	133.58	**100 %**	**2.52 s**	**199.61**	**100 %**	2.77 s	**230.32**
SE3 ctrl	100 %	2.16 s	163.19	74 %	3.27 s	297.59	50 %	3.18 s	285.64

4. Flaw and Future works

We also evaluated our policy on a path-tracking task by feeding offset position and velocity into the hover policy. It struggled with high-speed or steep curves. We tried to train a path-tracking policy using the RL framework and proved difficult due to poor convergence and generalization. Future works could involve fine-tuning the hover policy for path tracking or learning a residual policy to compensate for disturbances on a nominal controller's output.

5. Video, Code and References:

github: https://github.com/KANZEZ/rl_uav_ctrl

Youtube

References links:

[1] https://arxiv.org/abs/2306.04485

[2] https://arxiv.org/pdf/2311.13081

[3] https://arxiv.org/pdf/1802.09477

[4] https://ieeexplore.ieee.org/document/5717652