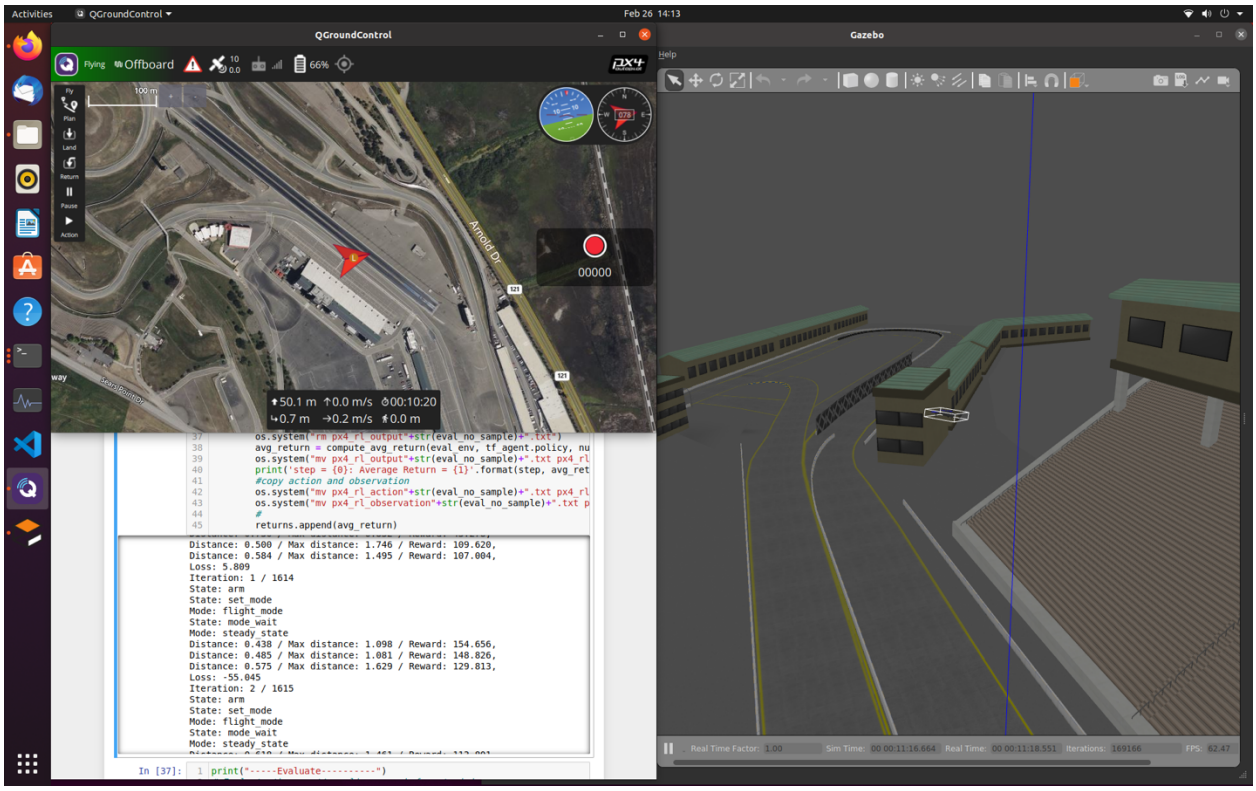
Deep Reinforcement Learning of PID Control for Rotational Wing Aerial Vehicles



Angelina Kim
angelina.kim.25@bishops.com

Dr. Marcus Jaiclin
marcus.jaiclin@bishops.com

The Bishop's School, La Jolla, CA

Table of Contents

Abstract

Deep Reinforcement Learning (DRL) in a mission controller has potential to improve the performance of a Rotational Wing Aerial Vehicle (RWAV) with static Proportional, Integral, and Derivative (PID) controllers by updating the PID set-up dynamically. This project examined the effect of an implemented DRL network on a RWAV in Software-In-The-Loop (SITL). It started by examining a baseline of a static PID RWAV's position-holding response under random wind direction and velocity. Then the DRL network was trained by flight telemetry data including a gyroscope at 10Hz, and it dynamically updated the PID controllers with their coefficients to hold a target position at 50m above the coordinate origin under random wind. Two DRL engagement cases were experimented: 1) the rate PID controller only and 2) all position, velocity, attitude, and rate PID controllers. A 5-layer DRL network with the REINFORCE TensorFlow agent was used. Under random wind under the mean of 5m/s velocity, the DRL-engaged RWAV's training and evaluation processes were collected and compared. The DRL network with all-dynamic PID controllers' 19 coefficient updates improved mean position consistency by 45% at 0.27m from a static PID controller's 0.5m under 5m/s random wind. The rate PID controller update with 8 coefficients maintained a similar distance at 0.5m with slightly lowered standard deviation by 1.2%. Thus, the all-dynamic PID controller was more effective in training the DRL to stabilize the RWAV position holding.

Acknowledgements

I would like to thank Dr. Marcus Jaiclin for his continuous guidance and patience. Another big thank-you to my community and family for their support and care.

Introduction

Recently, Rotational Wing Aerial Vehicles (RWAV) have increased in number. GAO estimated U.S. will have a 1.5 million recreational drone fleet by 2024 (Drone operations, 2023). FAA reports 0.87 million drones have registered (Unmanned Aircraft Systems, 2023). Manned RWAV will be next step due to its versatility. The automatic control of RWAV is essential for safety, and the flight integrity is managed by nested Proportional, Integral, and Derivative (PID) controllers (PX4 Drone Autopilot, 2023).

RWAV has multiple levels of nested position, velocity, acceleration, attitude, and rate PID controllers. They are fixed to a conservative set of values with margin for stable flight. Under extreme conditions, such as strong wind, RWAV might not be able to maintain its flight and mission. If PID controllers are tuned against such conditions, it might have the capability to overcome the conditions.

Deep Reinforced Learning (DRL) networks have been recently introduced as an attractive machine-learning algorithm for complex problem solving (AlphaGo, 2023). It operates from observations to generate actions, then learn from rewards. Through iterative trainings, it has demonstrated performance beyond human intelligence.

Using a Software-In-The-Loop (SITL) physics simulation of RWAV and DRL network, flight and mission controls can be integrated together. There is a potential for dynamic performance and safety improvement in flight control.

Review of Literature

PX4 autopilot has been adopted as an open-source platform for ground, marine, and aerial vehicle development (PX4 Drone Autopilot, 2023). PX4 supports both SITL and Hardware In The Loop (HITL) simulations (PX4 autopilot user guide, 2023).

Gazebo is an open-source 3D robotics simulator with a high-performance physics engine adopted by NASA and DARPA robotics challenges (Gazebo, 2023). Gazebo's wind plugin introduces random wind with average and variation values for velocity and direction. Only horizontal wind at random direction and velocity is used for the DRL network training and evaluation. Currently wind plugin treats wind as a force to the vehicle.

QGroundControl assists the test process of RWAV in SITL and HITL by providing visual assistance through real-time vehicle status display (QGroundControl, 2023).

As shown in Figure 1, the Micro Aerial Vehicle Link (MAVLink) connects PX4, Gazebo simulator, QGroundControl, and mission control program through data networks. All components are implemented in a single computer and the MAVLink is established through a TCP and UDP loop backs (MAVLink Developer Guide, 2023).

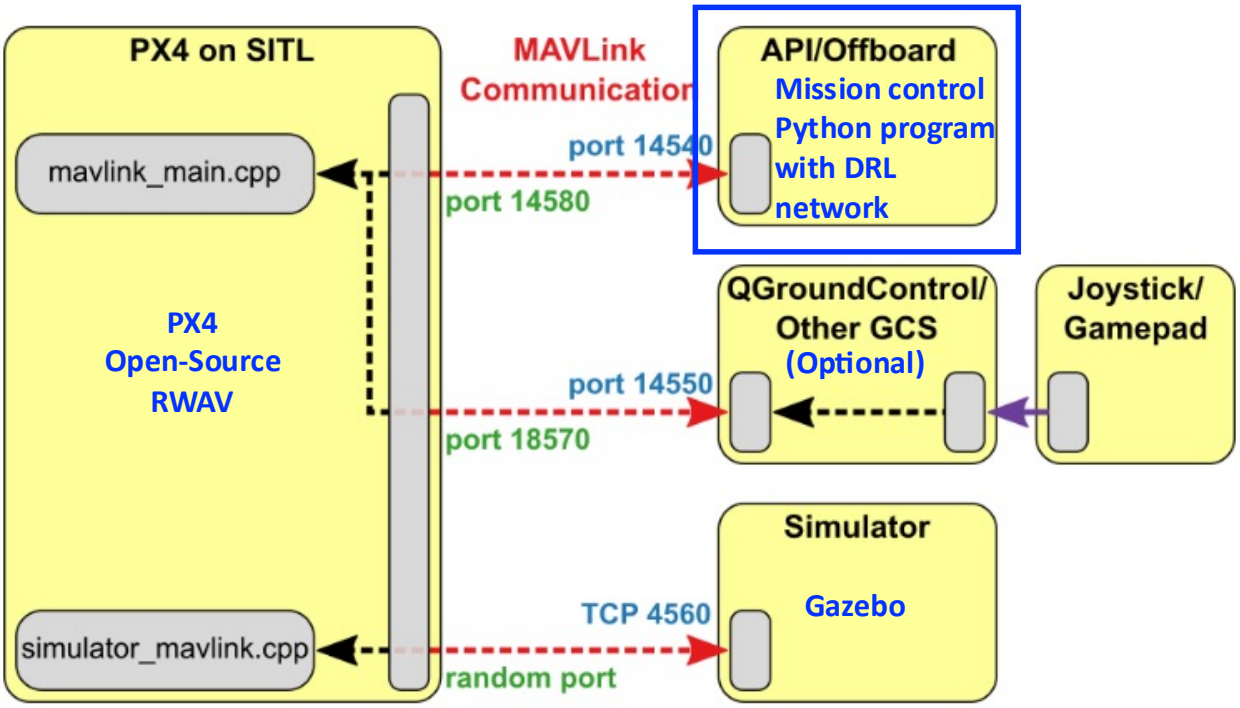


Figure 1. Block diagram of RWAV simulation and communication (Simulation, 2023). PX4 RWAV flight controller firmware runs SITL. Gazebo physics simulator interfaces with PX4 to provide flight physics including random wind. QGroundControl is an optional tool to analyze flight status. Mission controller is a custom Python code developed for this project with DRL TF-agent to update PID controller coefficients dynamically based on flight telemetry data. System blocks are connected through loop-back TCP and UDP ports within a Linux development workstation.

The mission control program is a custom Python code programmed for this project to communicate with PX4 RWAV and to bring the vehicle through the test procedure and to implement the DRL network agent for training and evaluation.

The DRL network uses REINFORCE TF agent to update PID controller coefficients (REINFORCE agent, 2023). REINFORCE agent supports continuous action space, rather than discrete action space such as Deep Q Network (Introduction to RL and Deep Q Networks, 2023).

Figure 2 depicts the mission controller's interaction with PX4 RWAV and PID controllers. The mission controller program requests flight telemetry data at 10Hz rate to PX4 RWAV through MAVLink. The telemetry parameters are received as MAVLink messages, then provided to the DRL as input or observation. The DRL generates actions as PID controller

coefficients, and they are updated to each controller through MAVLink set command at 10Hz rate.

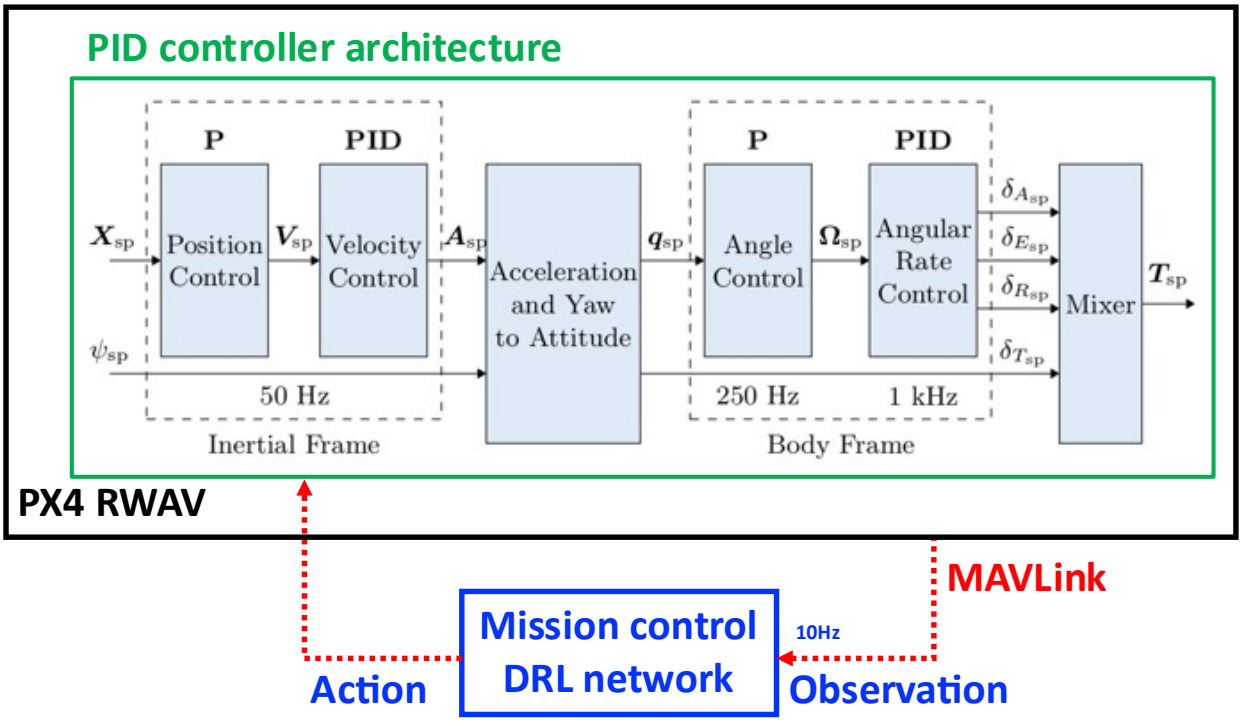


Figure 2. RWAV PID controller architecture (Controller Diagrams, 2023) and mission control DRL network interface diagram. Mission controller with DRL network receives flight controller's 50-point telemetry data from MAVLink message at 10Hz. The DRL agent provides actions to update PID controller coefficients at 10Hz.

Table 1 shows flight telemetry data collected from RWAV (MAVLINK Common Message Set, 2023). Each message name and number are unique, and they are received as MAVLink message at 10Hz. MAVLink has extensive list of parameters, and PX4 RWAV has another set of extensive custom parameters (Parameter Reference, 2023). This project utilized MAVLink standard parameters only for training. The parameters include gyroscope, intermediate control parameter attitude, position, actuator output, and estimator. Total 50 floating point data set is normalized between -1 and 1 to simplify observation input to the DRL. For

MAVLink's Python integration, pymavlink (pymavlink, 2023) was used as recommended by MAVLink development guide (MAVLink Developer Guide, 2023).

MAVLink message #	MAVLink message name	Description	Number of data
26	SCALED_IMU	X, Y, and Z angular acceleration and speed	6
30	ATTITUDE	Roll, pitch, and yaw angle and angular speed	6
31	ATTITUDE_QUATERNION	Quaternion component 1 - 4, roll, pitch, and yaw angular speed	7
32	LOCAL_POSITION_NED	Local position and speed in X, Y, and Z with respect to North, East, and Down (NED)	6
36	SERVO_OUTPUT_RAW	Servo 1-4 output values	4
74	VFR_HUD	Air and ground speed, heading, throttle, altitude, and climb rate	4
83	ATTITUDE_TARGET	Current vehicle attitude target in attitude quaternion, body roll, pitch, and yaw rate, thrust	8
230	ESTIMATOR_STATUS	Output of EKF estimator. Velocity, horizontal, vertical innovation test ratio	4
340	UTM_GLOBAL_POSITION	X, Y, and Z velocity	3
12901	OPEN_DRONE_ID_LOCATION	Horizontal and vertical speed	2
		Total parameters	50

Table 1. Selected MAVLink telemetry data list and number of data points (MAVLINK Common Message Set, 2023). Data is refreshed at 10Hz rate, and fed into the DRL network observation input during training and evaluation. Total 50 data points are used. MAVLink provides extensive data set, and there are more potential parameters which might be useful for DRL. Also, MAVLink command can extract PX4 flight controller's internal parameters.

The DRL generates actions as output when the observation of telemetry data set is given. Each value of action corresponds to a coefficient of PID controllers. Figure 3 describes K, I, and D coefficient updates at roll and pitch rate controller. Yaw rate controller uses K and I only, and the DRL updates them accordingly out of the action values.

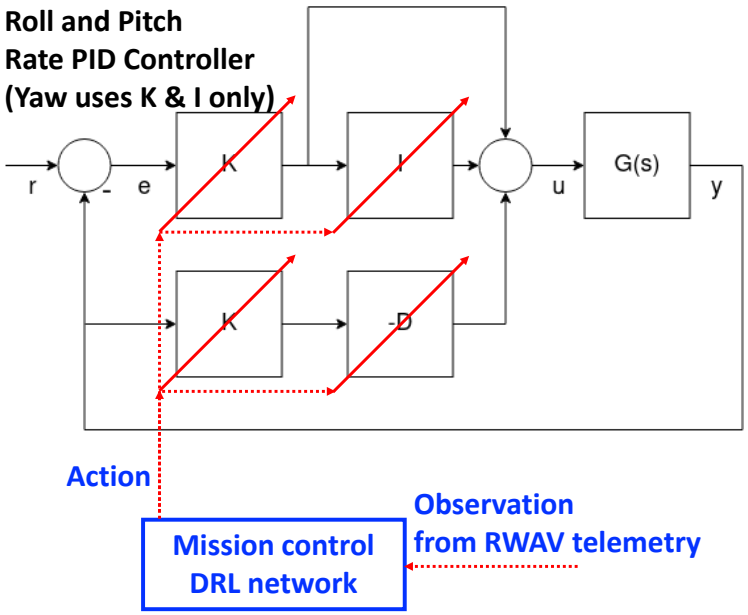


Figure 3. Roll and pitch rate PID controller coefficient K, I, and D update by the DRL network.

All PID controller's coefficients can be set through MAVLink set command. Table 2 shows PID coefficients updated by the DRL dynamically. The DRL is given with the minimum and maximum action range allowed to generate action space. While controllers have different operating rate, the DRL uses 10Hz update rate. There are two test cases: 1) rate controllers with 8 PID coefficients, and 2) all PID controllers with 19 PID coefficients updated by the DRL network. There were a few cases where RWAV crashes and flipped during training, as the action space generated by DRL explores randomness and extreme values where PID controllers cannot recover from the descent. After each experiment and training, PID controllers are set to default coefficients to return to original target position within 0.3m before next session.

#	Controller	PID parameter name	Minimum	Typical	Maximum
1	Rate @ 1000Hz	MC_PITCHRATE_K	0.3	1	3
2		MC_PITCHRATE_D	0.0004	0.003	0.01
3		MC_PITCHRATE_I	0.1	0.2	0.5
4		MC_ROLLRATE_K	0.3	1	3
5		MC_ROLLRATE_D	0.0004	0.003	0.01

6		MC_ROLLRATE_I	0.1	0.2	0.5
7		MC_YAWRATE_K	0.3	1	3
8		MC_YAWRATE_I	0.04	0.1	0.4
9	Attitude @ 250Hz	MC_ROLL_P	1	6.5	14
10		MC_PITCH_P	1	6.5	14
11		MC_YAW_P	1	2.8	5
12	Velocity @ 50Hz	MPC_XY_VEL_P_ACC	1.2	1.8	5
13		MPC_XY_VEL_I_ACC	0.2	0.2	10
14		MPC_XY_VEL_D_ACC	0.1	0.2	2
15		MPC_Z_VEL_P_ACC	2	4	15
16		MPC_Z_VEL_I_ACC	0.2	2	3
17		MPC_Z_VEL_D_ACC	0	0	2
18	Position @ 50Hz	MPC_XY_P	0	0.95	2
19		MPC_Z_P	0	1	2

Table 2. RWAV's PID coefficients at rate, attitude, velocity, and position controllers with operatin frequency (Parameter Reference, 2023). For this project, PID controller coefficients only were utilized to implement DRL mission controller. PX4 offers extensive set of parameters, and there are additional set of parameters might be useful for DRL. PX4 PID coefficients are updated to the flight controller through MAVLink's parameter setting command. DRL action space utilizes range of the coefficients during training. PID controllers are reset to typical values before training and evaluation.

The experiment uses the wind plugin in Gazebo simulator. Table 3 show a wind definition for the experiment. Mean velocity is at 5m/s and maximum wind is limited to 7m/s. Only horizontal wind is activated to emulate wind blowing in open field. The wind is implemented with C++ standard normal random distribution (PX4-SITL_gazebo-classic, 2023).

```
<plugin name='wind_plugin' filename='libgazebo_wind_plugin.so'>
   <frameId>base_link</frameId>
   <robotNamespace/>
   <windVelocityMean>5.0</windVelocityMean>
   <windVelocityMax>7.0</windVelocityMax>
   <windVelocityVariance>0.5</windVelocityVariance>
   <windDirectionMean>1 1 0</windDirectionMean>
   <windDirectionVariance>0.5</windDirectionVariance>
   <windGustStart>0</windGustStart>
   <windGustDuration>0</windGustDuration>
   <windGustVelocityMean>0</windGustVelocityMean>
   <windGustVelocityMax>0.0</windGustVelocityMax>
```

```
    <windGustVelocityVariance>0</windGustVelocityVariance>
    <windGustDirectionMean>0 0 0</windGustDirectionMean>
    <windGustDirectionVariance>0</windGustDirectionVariance>
    <windPubTopic>world_wind</windPubTopic>
</plugin>
```

Table 3. Gazebo's wind plugin definition section used for simulation (windy.world, 2023). Major parameters used in the simulation are highlighted. Mean velocity of 5m/s horizontal random wind was used. The plugin also support gust with timing information, and it was set to 0 to use random wind.

Statement of Problem

When RWAV's are deployed, there are challenges to fly safely and efficiently under extreme conditions as their flight is sustained by static PID controllers. This project applies a DRL network to tune PID controllers in real-time to improve the target position hold performance under random wind over static PID controllers.

Hypothesis

Can a RWAV perform better when a DRL network in mission controller tunes PID controller dynamically in real time over static PID controllers under challenging conditions such as random wind? Utilizing the RWAV's sensors such as barometers, distance sensors, GPS, and gyroscopes, this project implements a dynamic tuning of the PID controllers managed by a DRL network in the vehicle's mission controller. RWAV should be able to achieve better flight mission when PID controller coefficients are adjusted dynamically by a DRL network after simulated training from the RWAV flight telemetry data.

Materials

Development Linux workstation (i9-12900K / 128GB / 1TB / RTX3090)

Open-source PX4 Autopilot multi-copter rotational aerial vehicle simulator

Gazebo physics simulator with wind plugin

QGroundControl control station

Procedure

Table 4 summarizes the sequences to engage PX4 RWAV SITL, Gazebo simulator, and Python mission controller for training and evaluation of the DRL network. The sequence is automated in mission controller program. The RWAV is set to OFFBOARD, where the mission controller sends position command, while flight controller loops maintain target position.

Step	Description
1	Set up MAVLink to PX4 simulation
2	Reset PID controller coefficients to typical values
3	Arm actuators
4	Request telemetry data
5	Set flight mode to OFFBOARD
6	Set target position
7	Reach target position within 0.3m
8	Set up DRL network
9	Initial evaluation
10	Collect episodes Target position and telemetry data are updated at 10Hz
11	Start training
12	Save checkpoint
13	Evaluate agent after every 50 training iterations
14	Go to #10 and repeat

Table 4. RWAV set up, training, and evaluation procedure implemented in the mission controller with DRL. PX4 SITL and Gazebo are launched in advance. Checkpoints were copied at every 10 iterations, and they are used to bring back known DRL network status. Evaluation runs 30 sessions of 120-second DRL engagement under the random wind, and it take one hour.

Simulation blocks are independent application in the design workstation, and they need to be launched individually first. Figure 4 shows intermediate steps when PX4 RWAV and Gazebo are initialized and connected through TCP port, then QGroundControl engages RWAV through UDP port to retrieve and set flight information.

Figure 4. Left: Gazebo and PX4 initialization at ground origin. Right: After actuators are armed and commanded to take off, RWAV is ready to take mission commands.

When RWAV and Gazebo are ready, the mission controller is launched to set up a DRL network and take over RWAV's control as shown in Figure 5. In the screenshot, the mission controller is in training mode at 1615th iteration, collecting three one-minute or 600-sample episodes per iteration. The RWAV is asked to keep 50m above ground origin against random wind. Interactive mode in Jupyter Notebook is useful to recover from a saved DRL checkpoint, when there is a crash or training errors.

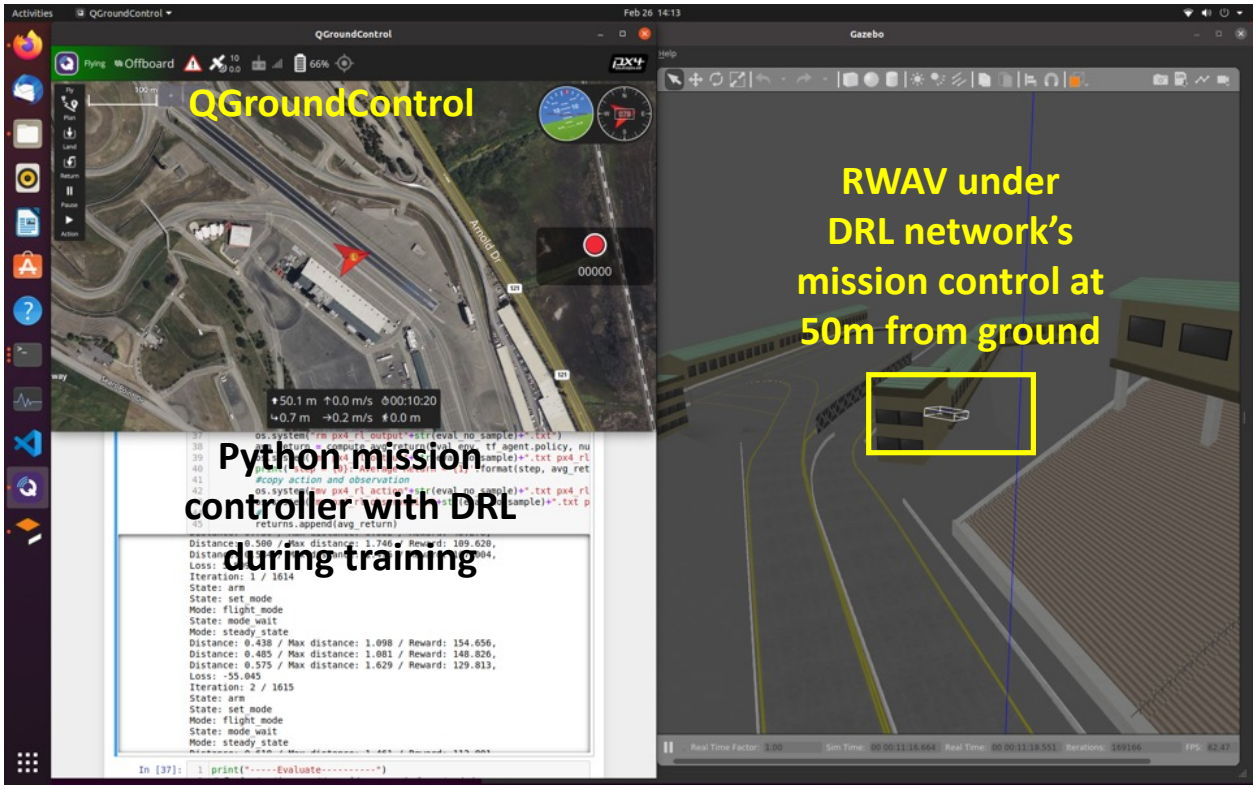


Figure 5. Simulation and training set up with QGroundControl, Gazebo physics simulator, and mission controller python program for DRL training and evaluation. QGroundControl provides visual status of the simulation as an optional interface.

The DRL network has inner layers with 200, 400, 200, 100, and 50 nodes. Input has 50 nodes for telemetry observation. Output has either 8 or 19 nodes for two cases of PID coefficient set experiments. Learning rate was 0.0001, and three one-minute episodes were collected at each iteration for training. Training loss entropy in log scale was recorded after each training, and the TF agent was saved to a checkpoint. After every 50 training, the DRL network and RWAV position hold were evaluated through 30 sessions of 2-minute position hold flight.

Results and Statistical Analysis

There are three experiments compared. The first experiment examines static PID control RWAV's position hold capability under increasing wind velocity to analyze the vehicle's flight and to set reference wind statistics for following experiments. The second and the third experiments introduce mission controllers with a DRL network to update two different groups of PID controller coefficients dynamically.

Static PID RWAV Performance with Gazebo Wind Plugin

To establish a comparison baseline, static PID controller's position hold performance is measured under random wind with controlled mean velocity. It reveals the RWAV's operation limit in simulation environment, and the wind environment set up where the performance of DRL-engaged PID controllers should be tested can be derived.

For the experiments, standard deviation was set at 0.5m/s from mean velocity sweep, and the maximum velocity is limited to 2m/s from mean velocity. Wind direction is random horizontally. Average distance is obtained from target over 1200 samples with 10Hz data for 2 minutes. 3 episodes are recorded per wind set up to observe repeatability. RWAV is put within 0.3m from the target position before data collection starts.

Wind velocity sweep results in Table 5 and Figure 6 show wind plugin and PID RWAV's behavior and limits. The data and plot show that the maximum distance after 6m/s mean velocity increases significantly. It suggests static PID RWAV's capability is limited to 6m/s wind, and stable statistics will need a larger number of samples. RWAV crashed multiple times at 9m/s and failed to recover.

Wind mean velocity (m/s)	0	0	0	1	1	1	2	2	2	3	3	3	4	4	4	5	5	5	6	6	6	7	7	7	8	8	8
Average distance (m)	0.07	0.06	0.06	0.17	0.14	0.15	0.23	0.20	0.23	0.31	0.26	0.33	0.41	0.34	0.43	0.51	0.42	0.54	0.58	0.56	0.66	0.79	0.70	0.89	1.13	1.01	1.23
Maximum distance (m)	0.13	0.11	0.12	0.48	0.40	0.47	0.73	0.51	0.80	1.01	0.71	1.09	1.24	0.91	1.44	1.95	1.40	1.82	1.91	2.42	1.70	4.83	4.17	5.89	7.01	8.05	8.36

Table 5. Static PID RWAV's position hold measurements under increasing mean velocity random wind.

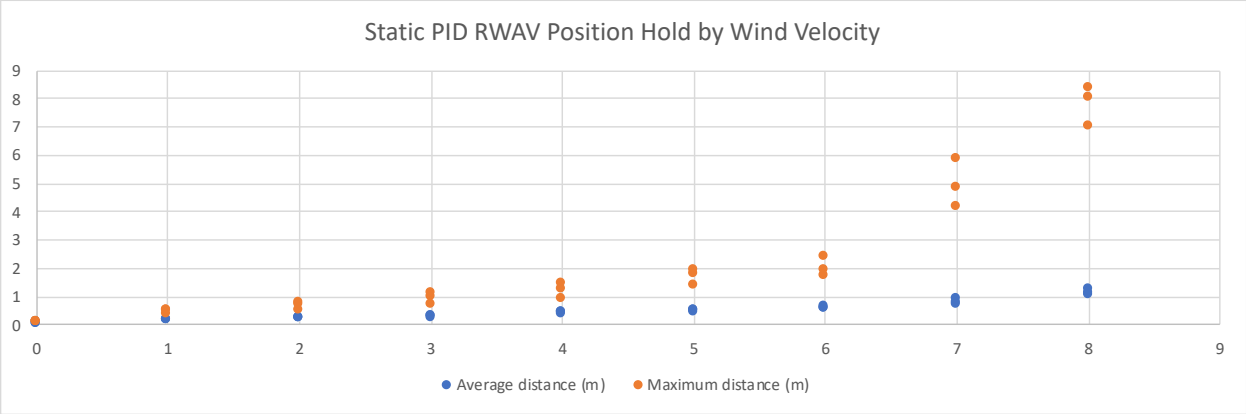


Figure 6. Static PID RWAV position hold measured as mean distance from target over wind mean velocity sweep. RWAV fails to keep the position effectively against wind beyond 6m/s wind.

For the results, wind mean velocity at 5m/s is used to compare the baseline static PID RWAV and RDL-engaged RWAV. The next step is to ensure the baseline reference is stable with multiple evaluation sessions. At 5m/s mean wind velocity, average distance was observed from total 30 episodes for 60 minutes in Table 6 and Figure 7.

Iteration	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Average distance (m)	0.50	0.41	0.52	0.45	0.58	0.50	0.48	0.51	0.50	0.50	0.48	0.48	0.43	0.56	0.51	0.46	0.50	0.50	0.52	0.50	0.48	0.45	0.49	0.49	0.50	0.53	0.50	0.55	0.54	0.45
Maximum distance (m)	1.85	1.15	1.71	1.19	1.72	1.36	1.37	1.33	1.36	1.58	1.33	1.13	1.35	1.57	1.30	1.27	1.64	1.38	1.21	1.49	1.83	1.45	1.26	1.35	1.68	1.55	1.83	1.74	1.53	1.30

Table 6. Static PID RWAV's mean and maximum distance measurement from 30 samples of 120-second evaluation.

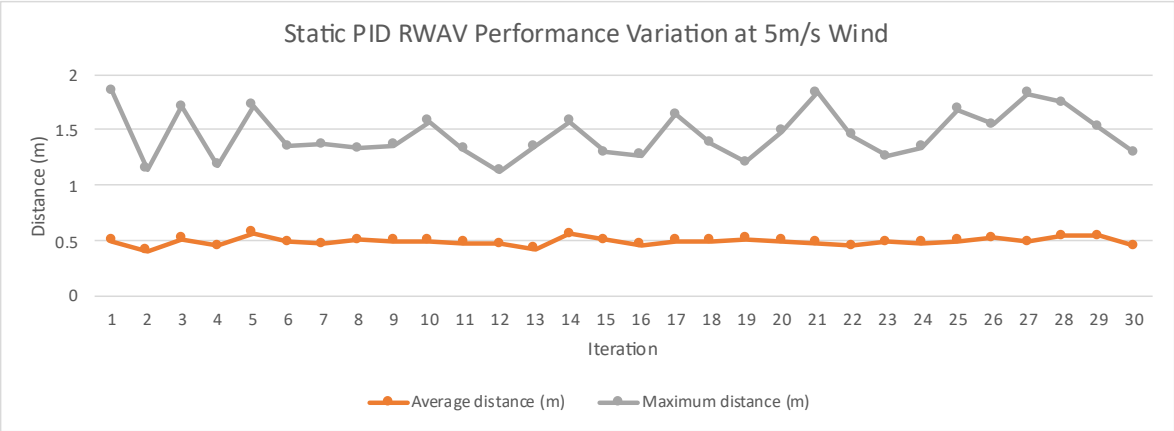


Figure 7. Static PID RWAV's mean and maximum distances from target at 5m/s wind velocity from 30 samples of 120-second evaluation. Total evaluations takes one hour.

The 30 episodes' wind statistics show there is up to 17% average distance changes and 7.4% standard deviation as summarized in Table 7.

	Average distance	Maximum distance
Mean (m)	0.50	1.46
Standard deviation (m)	0.04	0.21
Minimum distance (m)	0.41	1.13
Maximum distance (m)	0.58	1.85
Standard deviation/mean (%)	7.4%	14.7%

Table 7. Summary of static PID RWAV position hold under 5m/s wind. Statistics were obtained from 30 samples of 120-second evaluation.

Results with DRL-Engaged Rate PID Controllers

For the first case, rate PID controllers' coefficients are dynamically update by the DRL action. The rate controllers are the fundamental and fastest (1kHz) control loops in RWAV. The rate pitch and roll controllers have K, I, and D coefficients as depicted in Figure 3, and yaw rate controller has K and I coefficients. Total 8 coefficients are updated by the DRL network at 10Hz rate. Figure 8 shows gyroscope's X- and Y- axis accelerations plotted over time as observations to the DRL. The acceleration should reflect wind force and actuator thrust applied to the body. Figure 9 shows the output action of DRL corresponding to K coefficient of roll rate controller at

the first iteration. It has narrow range of random outputs based on the observation to initiate reinforced learning.

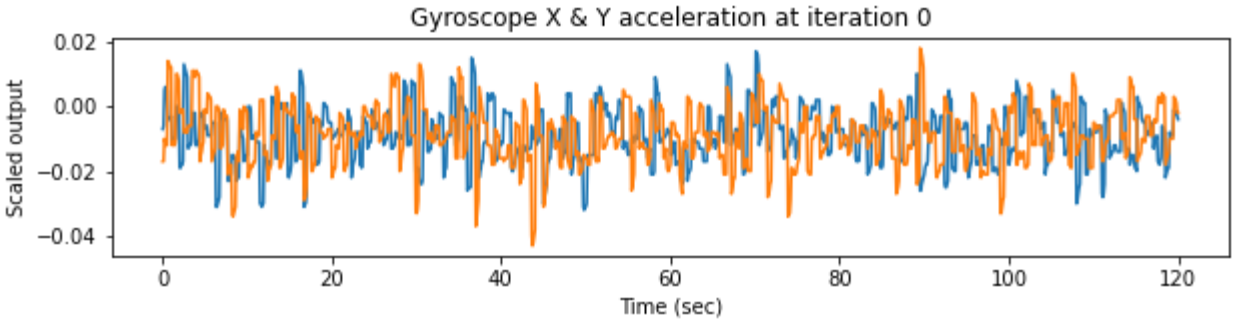


Figure 8. An example of RWAV telemetry data. Gyroscope X- and Y-axis accelerations at the first iteration.

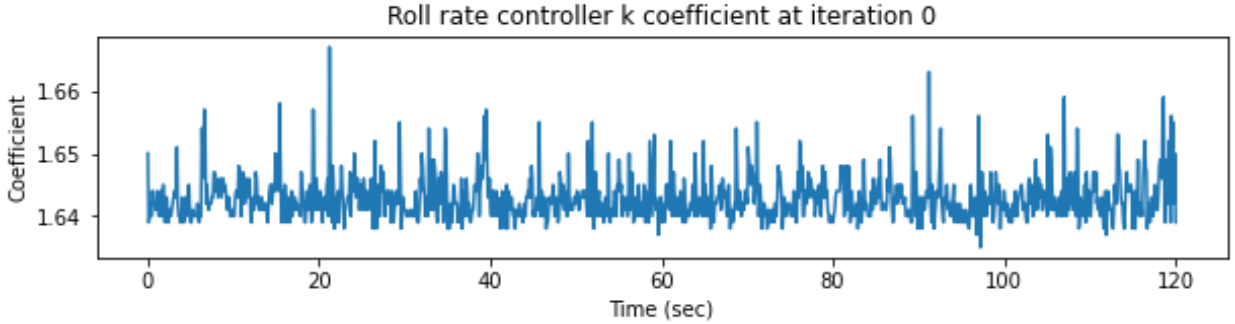


Figure 9. Roll rate controller K coefficient from the DRL output action at iteration 0.

After training, we expect the controller coefficients to respond to the varying telemetry data to improve position hold flight performance. Figure 10 compares gyroscope's x-axis acceleration and pitch rate controller coefficients as the DRL output action at 1300th iteration. They show subtle tracking behavior as the telemetry observation changes.

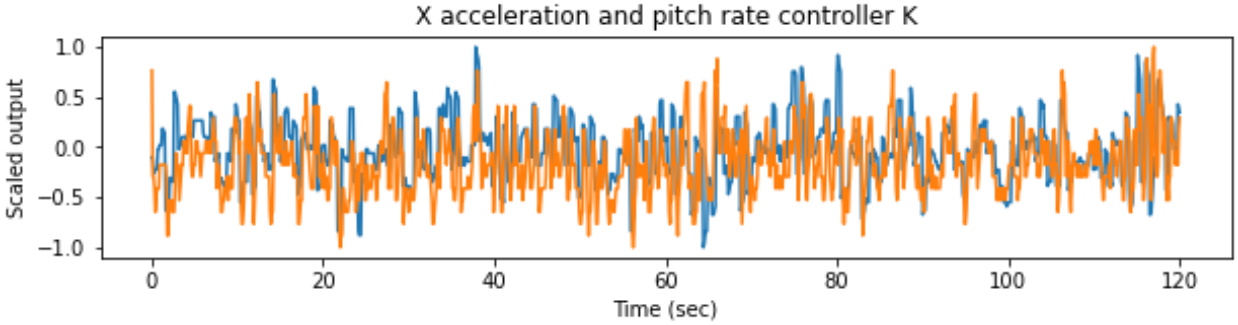


Figure 10. An example of telemetry observation and DRL action. Gyroscope X-axis acceleration and pitch rate controller K coefficients show subtle tracking behavior at iteration 1300.

During training, the DRL network showed continued learning with loss entropy in log scale as plotted in Figure 11. The rate controllers showed no significant changes in position hold evaluation as summarized in Table 8. The mean distance from target position was similar, and the standard deviation decreased slightly. While the DRL network was responding to the telemetry data as shown in Figure 10, the trained DRL network performed similarly to the static PID controllers.

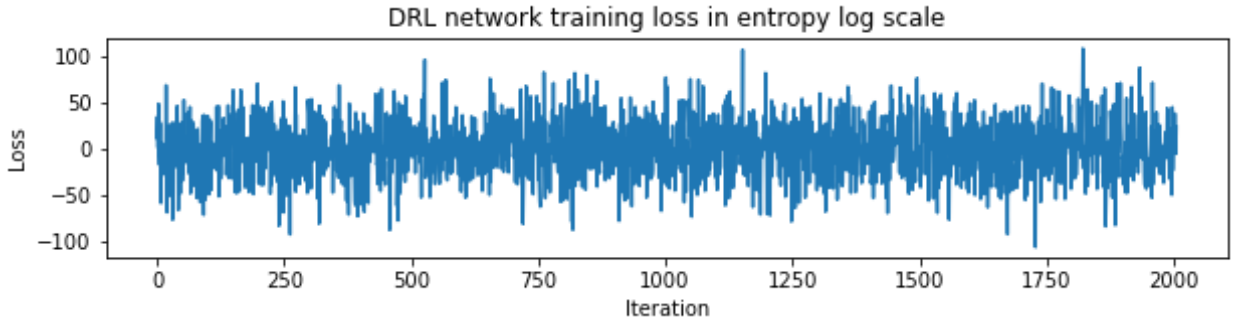


Figure 11. The DRL network loss after training in entropy log scale.

Average distance	Static PID	DRL with rate controllers
Mean (m)	0.50	0.50
Standard deviation (m)	0.04	0.03
Standard deviation / mean (%)	7.4%	6.2%

Table 8. Position hold performance summary and comparison between static PID and rate controllers with DRL network.

Results with DRL Engagement with All PID Controllers

In addition to the rate controllers, attitude, velocity, and position controllers are directed by the DRL network to assess if there are further improvements from additional DRL involvement. There total 19 coefficients updated by the DRL network output action. Figure 12

shows initial XY velocity controller I coefficient updated from DRL at the first iteration, and its behavior is random as expected.

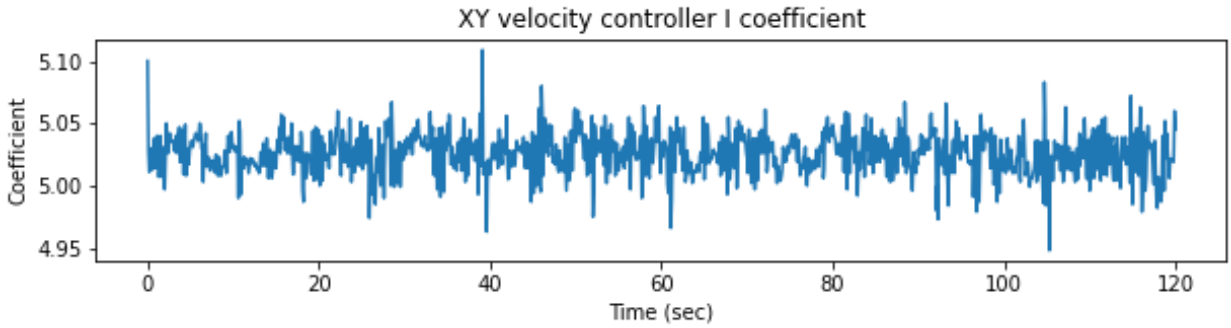


Figure 12. Initial DRL network action output for XY velocity controller I coefficient.

After training, the DRL network showed bimodal behavior in the set of PID coefficients along the fine changes at each mode. Figure 13 shows the XY velocity controller I coefficient making a transition at two different modes at around 5 minutes and 35 seconds. Still, the output has random element around the modes. A histogram in Figure 14 with two modes has variation around it.

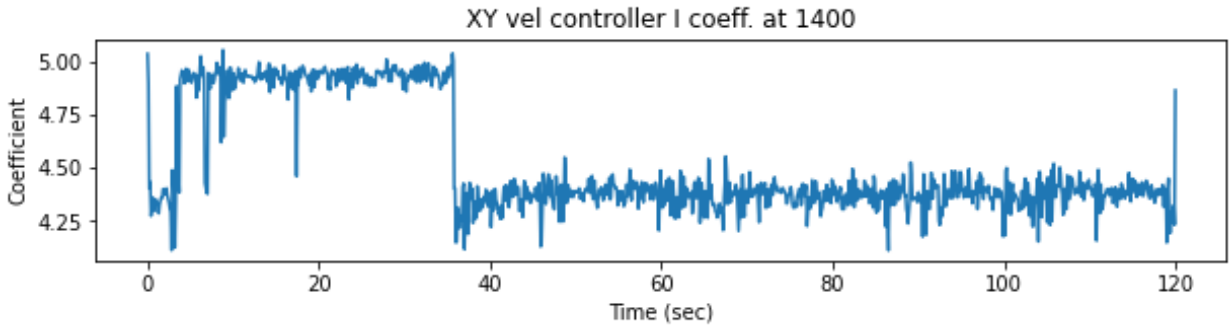


Figure 13. XY velocity controller I coefficient output from the DRL network at iteration 1400.

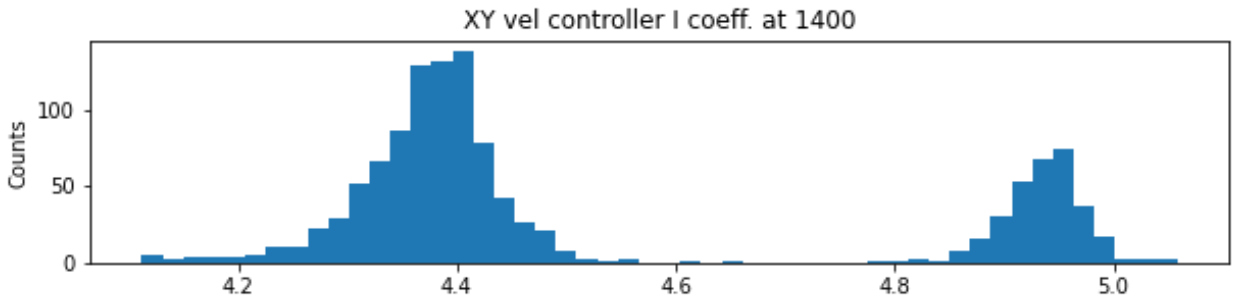


Figure 14. Histogram of XY velocity controller I coefficient at iteration 1400.

In addition to XY velocity coefficients, other PID controller coefficients show similar bimodal behavior. Figure 15 compares X position telemetry data and XY position controller P coefficient over evaluation. The tracking behavior is subtle, and the P coefficient makes transitions at similar timelines as XY velocity coefficients, while variation is larger at each mode.

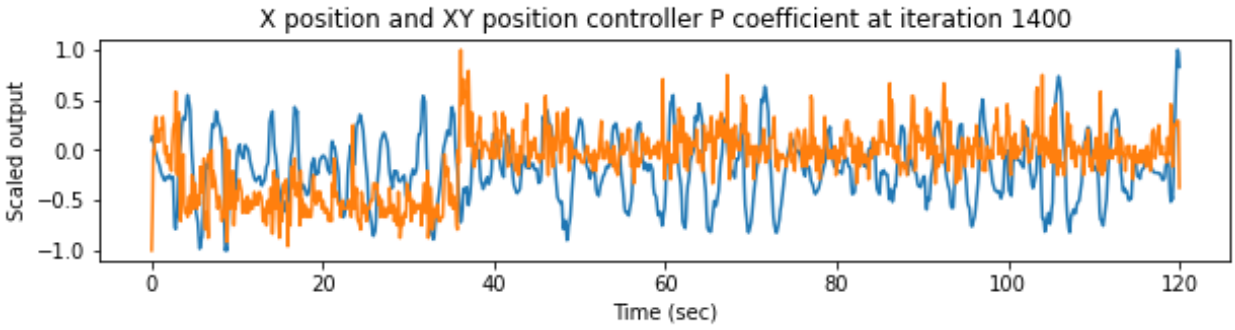


Figure 15. Comparison of X position telemetry data and XY position controller P coefficient at iteration 1400.

After 1500 training iteration, the DRL network with all PID controller coefficients improved position holding by 45% at 0.27m from static controllers' 0.50m as summarized in Table 9.

Average distance	Static PID	DRL with all PID controllers
Mean (m)	0.50	0.27

Standard deviation (m)	0.04	0.02
Standard deviation / mean (%)	7.4%	7.9%

Table 9. Position hold distance comparison between static vs. PID updated by the DRL network.

Analysis

The DRL network was integrated to update dynamic rate controllers and all PID controllers to examine if deep machine learning can tune PID controllers in a physics simulation with training. Table 10 summarizes the RWAV's position hold results after 1500 iterations. The DRL network with rate controllers did not improve average distance, while the DRL with all PID controllers improved position hold performance by 45% under random wind.

Average distance	Static PID	DRL with rate controllers	DRL with all PID controllers
Mean (m)	0.50	0.50	0.27
Standard deviation (m)	0.04	0.03	0.02
Standard deviation/mean (%)	7.4%	6.2%	7.9%

Table 10. Comparison of static PID, DRL with three rate PID controllers, and DRL with all PID controllers.

The DRL network updates started from random varying output when telemetry observations are provided. After training, the DRL network action output showed tracking behavior as PID controller coefficient. Also, DRL network formed bimodal coefficient sets across multiple controllers. Even after the training and bimodal distribution, the coefficients showed variation along the time. Further study would be needed to understand how the PID controller coefficient updates maintain or improve position hold flight mission performance.

During training, the DRL network action generates random perturbations to explore optimization space even with the known observation. Also, insufficient training can put the RWAV PID controllers in a status where the vehicle cannot sustain flight. The resulting crash and flipping have been noticed during training. In such case, the simulation set up is refreshed to recover. The 50m above the origin is useful to give the RWAV time to recover temporary altitude loss during training. No crash or flipping was reported during evaluation sessions. Figure

16 shows example PID controller tracking behavior during training and evaluation. With the DRL network's output variation, a PID controller loop can set oscillatory response. With further training, the tracking becomes similar to a static and stable PID controller response.

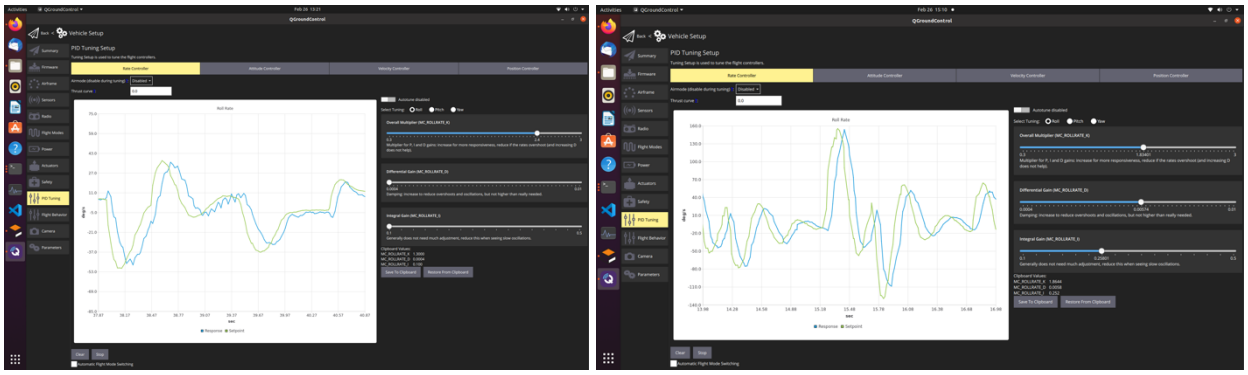


Figure 16. Rate controller set point and response tracking behavior observed through QGroundControl. Left: set point shows oscillatory behavior. Right: mature training brings up more stable and conventional tracking.

Conclusion

The DRL network tuned RWAV PID controllers dynamically, and its position hold was compared with static PID controllers under random 5m/s wind. The DRL-engaged rate PID controllers with 8 coefficients showed comparable position consistency at 0.50m as static PID controllers. When the DRL managed all PID controllers with 19 coefficients, position consistency was improved by 45% at 0.27m.

This project investigated basic aspects of DRL-engaged PID controllers for RWAV. There are extensive variables to explore: order of PID controller training, wind direction, range of variation, selective flight telemetry data, vehicle oscillation, DRL algorithm, longer training, more realistic flight mission scenario, etc. SITL acceleration will help extend the coverage with computational resource.

Recommendations

Many parameters used in the project are rather arbitrary. Examples are telemetry parameters, telemetry update rate, PID controller support variables, PID controllers to update coefficients, size and depth of the DRL network, DRL optimizer and policy, number of DRL network among different group of PID controllers, etc. The fundamental mechanism that PID controllers behave as static and dynamic coefficients needs further investigation to understand tuning and training process. The simulation runs in real-time, and it slows down training process limited by time scale regardless of computational capability. The simulation should be scaled in time to reduce training latency.

Bibliography

AlphaGo. (2023, March 1). Retrieved from DeepMind: https://www.deepmind.com/research/highlighted-research/alphago

Controller Diagrams. (2023, March 1). Retrieved from PX4 User Guide: https://docs.px4.io/main/en/flight_stack/controller_diagrams.html

Drone operations. (2023, March 1). Retrieved from U.S Government Accountability Offices (GAO): https://www.gao.gov/drone-operations

Gazebo. (2023, March 1). Retrieved from Gazebo Simulator: https://gazebosim.org/home

Introduction to RL and Deep Q Networks. (2023, March 1). Retrieved from TensorFlow: https://www.tensorflow.org/agents/tutorials/0_intro_rl

MAVLINK Common Message Set. (2023, March 1). Retrieved from MAVLink: https://mavlink.io/en/messages/common.html

MAVLink Developer Guide. (2023, March 1). Retrieved from MAVLink: https://mavlink.io/en/

MAVLink Developer Guide. (2023, March 1). Retrieved from MAVLink Developer Guide: https://mavlink.io/en/

Parameter Reference. (2023, March 1). Retrieved from PX4 User Guide: https://docs.px4.io/main/en/advanced_config/parameter_reference.html

PX4 autopilot user guide. (2023, March 1). Retrieved from PX4 autopilot user guide: https://docs.px4.io/main/en/

PX4 Autopilot User Guide. (2023, March 1). Retrieved from PX4 User Guide: https://docs.px4.io/main/en/

PX4 Drone Autopilot. (2023, March 1). Retrieved from PX4 Drone Autopilot Source Code: https://github.com/PX4/PX4-Autopilot

PX4-SITL_gazebo-classic. (2023, March 1). Retrieved from gazebo_wind_plugin.cpp: https://github.com/PX4/PX4-SITL_gazebo-classic/blob/main/src/gazebo_wind_plugin.cpp

pymavlink. (2023, March 1). Retrieved from pymavlink: https://pypi.org/project/pymavlink/

QGroundControl. (2023, March 1). Retrieved from QGroundControl: http://qgroundcontrol.com/

REINFORCE agent. (2023, March 1). Retrieved from TensorFlow: https://www.tensorflow.org/agents/tutorials/6_reinforce_tutorial

Simulation. (2023, March 1). Retrieved from PX4 User Guide: https://docs.px4.io/main/en/simulation/

Unmanned Aircraft Systems. (2023, March 1). Retrieved from Federal Aviation Administration (FAA): https://www.faa.gov/uas

windy.world. (2023, March 1). Retrieved from PX4 Autopilot: https://github.com/PX4/PX4-SITL_gazebo-classic/blob/main/worlds/windy.world