**Module 8: Portfolio Project**

**ML Agents Project**

Joshua Humphreys

Colorado State University Global

Course Code:CSC525

Dr. Pubali Banerjee

February 9, 2025

**Portfolio Project**

For my portfolio project, I built an ML Agent autonomous drone navigation system. The goal of the navigation system is to fly the drone from its point of origin to a designated target on the ground. The drone navigates an urban environment, including natural and man-made obstacles, and lands on top of the target. In the following pages, I will discuss the agent's architecture as well as the structure of gameplay. I will then discuss the construction of the reward signal, followed by how I decided on hyperparameters. Finally, I will discuss future improvements to include transfer and ensemble learning.

**Environment, Agent, and Game Architecture**

The environment I used for my agent is a demo environment provided by Unity. The environment comes preconstructed with an enclosed section of streets within buildings. Along the edges of the streets, there are benches, streetlamps, and trees. The environment included all necessary components for simulated physics interaction with other objects. The drone is a quadcopter prefab from a drone simulation package published by UAVs at Berkeley and Machine Learning at Berkeley (2018). The drone prefab produced by the Berkeley team originally included several components that weren't relevant to this exercise. The components I kept were the mesh structures to give the drone its quadcopter appearance. I kept the rigidbody associated with the drone but turned gravity off, as the purpose of this project is to create the navigation system rather than the flight control system. I removed several smaller, more detailed colliders, and replaced them with a single, large box collider. The original drone also came with multiple scripts for simulated drone flight that I removed since, again, the navigation system was the goal.

For inputs, I used 3D ray perception sensors. I used one sensor with 8 horizontal rays surrounding the drone. I also included a single ray looking up and another looking down. The drone also observes its position and velocity in space and its position relative to the target and the direction to the target. Its position relative to the target was separated into two separate measurements including 3D distance to the target as well as projected, 2D ground distance to the target. All told, the drone collects 19 observations not including the raycasts.

At the beginning of each episode, the drone and its target spawn at a random location along one of the centerlines of one of the streets. The drone and target's height are configurable. The drone then flies to the target location. The episode resets upon successful completion of the task, premature landing, or object strikes.

**Reward Signal**

Determining with which events to provide associated rewards was straightforward. Obviously, the drone should receive a reward for reaching its target. Object strikes were obvious criteria for negative rewards as well. Premature landings are a type of object strike. The more difficult reward signal to structure was rewarding the drone for moving closer to the target. I wanted to reward the drone for moving towards the target and penalize it for getting further away but didn't want to penalize it for gaining altitude. Gaining altitude, of course, means moving further from the target. The solution that worked quite well was to calculate distance in a plane and in three dimensions. Decreasing 3D distance results in a positive reward while increasing 2D distance results in a negative reward. In this way, the drone can gain altitude without consequence.

Tuning the reward signal did require some trial and error. The first few training iterations were very sporadic early on with minimal convergence and large negative cumulative reward. However, after a few adjustments based on mathematically proportional rewards and asymmetry favoring positive reward over punishment, the model appeared to respond well to training.

**Hyperparameter Selection**

The hummingbird agent tutorial provided by Unity was used as inspiration for this project. As such, much of the agent code and configuration was borrowed from that code. Given the hummingbird agent included 10 observations while the quadcopter included 19, I decided to double the number of hidden units in the model as well as the buffer and batch size parameters. Otherwise, the hyperparameter configuration used in the hummingbird project was based on PPO best practices, so I chose to replicate the same hyperparameters.

**Future Improvements**

The ultimate goal of this project is to create a navigation model that can be used to guide a real drone. For this project, the goal was simple enough and the inputs and outputs were simple enough that dividing the task into subtasks to be handled by multiple models didn't seem to make sense. However, future improvements could include the addition of a flight control system model that could receive the outputs of the navigation system and produce blade speed outputs to be passed to blade motors.

The decision not to use transfer learning was made mostly out of a desire to experience the process of training a model from scratch. However, in the future, the drone's raycast setup could be reconfigured and the observations could be simplified. The purpose of observation simplification would be to push the drone towards real-world application. In this case, rather than starting over, it would be best to start with the existing model and fine-tune it for the new inputs.

**Conclusion**

Finally, while the current implementation demonstrates successful autonomous navigation, there remains room for optimization. The occasional object strikes indicate that further training could improve performance. Additionally, as the operational environment continues to evolve, investigating advanced hyperparameter tuning methods could enhance the system's adaptability. This project has laid a solid foundation for future development, particularly in the transition from simulation to real-world application.

**References**

UAVs at Berkeley & Machine Learning at Berkeley (2018). *UnityDroneSim.* Github. <https://github.com/UAVs-at-Berkeley/UnityDroneSim>

Immersive Limit LLC (2025). *ML-Agents: Hummingbirds.* Unity Learn. <https://learn.unity.com/course/ml-agents-hummingbirds>

Raval, S. (2018, January 13). *Best Practises When Training With PPO.* Github. https://github.com/llSourcell/Unity\_ML\_Agents/blob/master/docs/best-practices-ppo.md