EN.525.637.81.FA23 —- Foundations of Reinforcement Learning Project Proposal: A curriculum learning approach to coordinated multi-agent drone delivery systems

Final Project

Alec Portelli

***Abstract*—- Modeling the real world to each a RL agent is an extremely difficult task. There are often limitations to trying to train a policy to do learn challenging quickly. Curriculum learning helps the agent understand hard problems with more ease. Pairing curriculum learning with PPO, simulations environments were built using Unity ML-Agents to train a drone to move from a start location to a goal location. A test environment was built so that the drone could learn to navigate without curriculum learning. The policy was then evaluated. After, the policy was implemented with curriculum learning and evaluated against the control. The results gave us insight as to how reinforcement learning (RL) policies are affected by introducing a curriculum learning approach to real world environments.**

1. Introduction

The demand for rapid and efficient shipment services has led to the exploration of autonomous drone delivery systems. Despite their potential benefits, developing fully autonomous drone systems faces challenges, particularly in real-world scenarios. This paper proposes a reinforcement learning (RL) approach, specifically utilizing a curriculum learning strategy, to train coordinated multi-agent drone delivery systems. Curriculum learning is essential to gradually expose agents to increasingly complex environments, addressing the impracticality of training on real-world delivery challenges from the outside.

2. Delivery Task

The task was to train digital drones to navigate realistic environments for package delivery and return. The drone would navigate a voxel environment to get around obstacles and other drones to get to a ‘goal’, which in this context would be the delivery location. Getting from one place to another makes use of localization, navigation, and control, which are all the most important aspects of drone delivery systems.

A - Environment

The Unity Engine was chosen as the simulation platform, leveraging its physics engine, ML-Agents plugin, and the Proximal Policy Optimization (PPO) algorithm for training. Game engines allow flexibility and the ability to customize every single part about the simulation. The training process initiated with a simple 3D 36x36x5 voxel environments, as shown in figure 1. Voxel environments were chosen so the agents could navigate around the world with a little more ease. This also allows for the use of discrete PPO, which was deemed as the better choice to work with curriculum learning. As aforementioned in the project proposal, robot control was not the goal of this project, but rather being able to each an agent on how to navigate a dynamic and difficult environment.

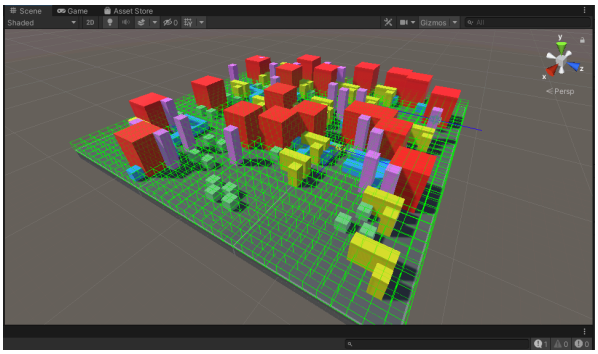


Fig. 1 - An example voxel world with spawned buildings

The obstacles in the world are a random configuration of buildings. There are five building prefabs, in which all of them have colliders on them for detection. All five buildings are in different shapes and sizes to add complexity. The

amount of buildings that are spawned depends on the stage of the curriculum learning process. The more buildings that are spawned the higher level of difficulty for the agent.

B - Agent

The agent within the simulation is a 3D model of a delivery drone. For this project, the drone does not have rigid body physics / kinematics, but does have a collider for collision detection. The drone also has a LiDAR with four beams extending in forward, back, left, and right directions. The beams can detect the distances to objects with colliders on them, which are the buildings and other agents.

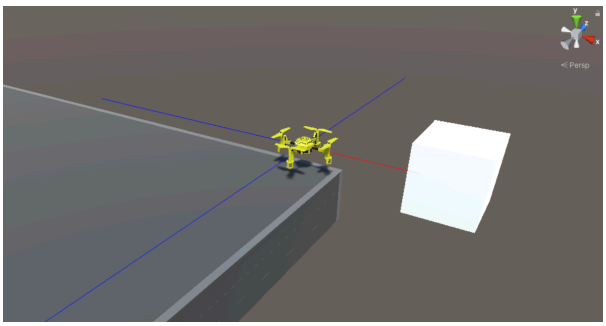


Figure 2 - Drone with LiDAR simulation detecting a cube with a collider

The agent's degrees of freedom within the simulation environment are to move: forward, backwards, left, right, up, down.

C - Problem Formulation

The problem is formulated using an episodic approach. The episode will end when the agent fails to complete the task or the agent takes too long to complete the task within the allowed amount of time, which is 5000 steps. There is a penalty given per step to ensure the agent is doing the task in as few steps as possible.

The Markov Decision Process (MDP) serves as the foundational framework. The MDP models the dynamic decision-making environment, defining states, actions, transitions, and rewards. Proximal Policy Optimization (PPO) is then applied to optimize the drone's policies within this MDP. PPO iteratively refines the drone's decision-making strategy, striking a balance between exploration and exploitation. This approach allows the drone to adapt to varying conditions.

For the project, the state, action, and reward modeling are as followed:

1. **State:** The state is discrete, which is represented by a 1x13 vector. The reward state includes the XYZ position of the agent, the XYZ of the other drone within the simulation, the XYZ position of the goal, and the four values of the LiDAR which represent the distances of nearby objects.
2. **Action:** The actions are discrete, which are represented in a 1x6 vector. The agent can move up, down, left, right, forwards, backwards.
3. **Reward:** The reward was shaped so that there was a major reward for reaching the goal, which is the ultimate objective of the task. However, there was a very large penalty for hitting the other drone or colliding into a building. This is because the first learned behavior of the drone is to not just go for the goal as the crow flies. The navigation skills that are needed to get to the goal come from obstacle avoidance and using the observations to localize the agent within the simulation.

   There is a smaller penalty for the done going out of bounds of the simulation environment.

   There is also a penalty for each step taken to make sure that the agent is taking the optimal path to get to the goal.

   The reward equation can be expressed as:

$$R_t = \begin{cases} 10 & \text{if the drone reaches the goal} \\ -10 & \text{if the drone goes out of bounds} \\ -100 & \text{if the drone crashes into a building} \\ -\frac{1}{\text{MaxStep}} & \text{for each step taken to encourage quick task completion} \end{cases}$$

Methods and Results:

To see the results of how curriculum learning affected the performance of the agent, a control simulation was conducted.

To start, six individual training environments were instantiated within Unity, all in which utilized domain randomization to create unique configurations for each episode. The density of spawned buildings was at a level 5, which was the most difficult. The most difficult was chosen because this was the equivalent of taking a drone and dropping it into a real world environment.

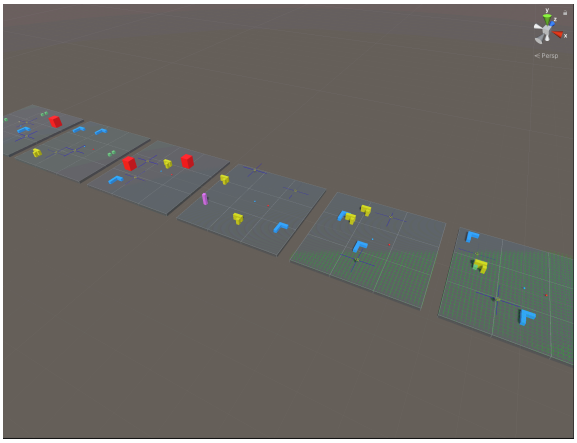


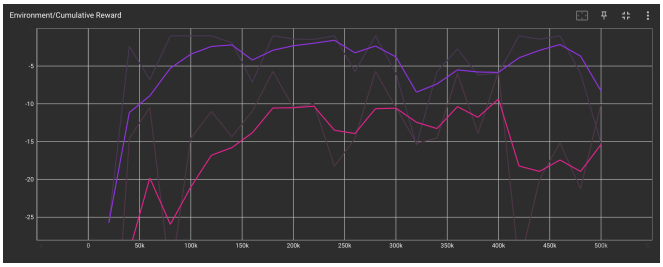
Figure 3 - Multiple training environments in parallel

The agent trained on the six parallel environments for 50,000 steps without any sort of curriculum learning.

Once the baseline training had been completed, the next step was to introduce the curriculum learning. Instead of starting at a level 5, the agents would start at a level 1, meaning that the density of the buildings would be reduced by 5 times. The main training script keeps track of the average reward, so as the reward gets better over time, then the difficulty of the environments increases as well. Every 10,000 steps the environment would increase a step in difficulty.

The agent was then trained for 50,000 steps using the six parallel environments ***with*** curriculum learning, and the results are as followed:

Figure 4: Cumulative reward over time (Purple is curriculum, pink is baseline)



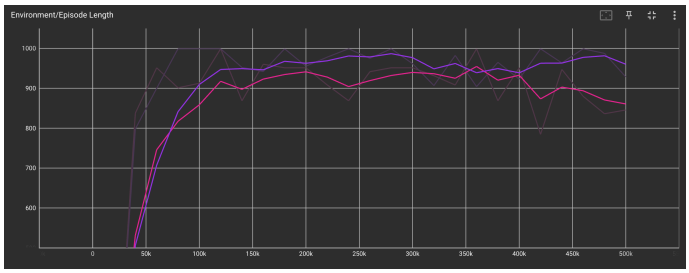


Figure 5: Episode length over time (Purple is curriculum, pink is baseline)

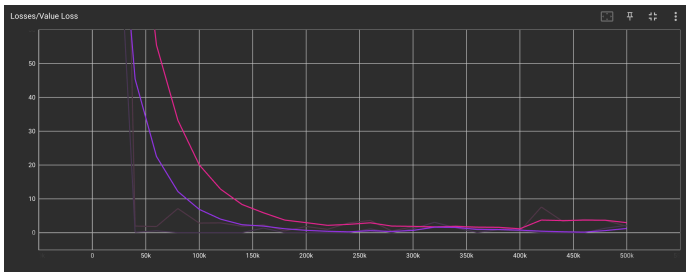


Figure 6: Loss over time (Purple is curriculum, pink is baseline)

The baseline approach exhibited fluctuations in reward scores without a clear plateau, indicating ongoing learning throughout the training process. However, the overall reward score was comparatively lower than the curriculum approach. The training process without a curriculum yielded more inconsistent results, reflecting the challenge for the RL algorithm to converge on a solution without progressive learning stages. Similar episode length convergence times were noted, suggesting that, in the absence of a curriculum, the algorithm faced a demanding task, making it challenging to achieve optimal performance.

In contrast, the curriculum learning approach demonstrated promising outcomes, showcasing a higher overall reward score compared to the baseline. Initial training phases exhibited a notable improvement, with the drone swiftly adapting to simpler environments. However, an observed plateau indicated that, after an initial surge, the reward remained consistent, revealing a challenge in further improvement.

The algorithm encountered difficulties and became "stuck" when confronted with

increasingly complex scenarios. This observed challenge raises considerations for the scalability of this approach. This suggests that while curriculum learning accelerates early-stage learning, it may struggle to adapt to higher levels of difficulty. Addressing these challenges is essential for ensuring that the RL algorithm can handle increasingly complex real-world delivery environments. Despite this limitation, the curriculum facilitated exposure to a diverse range of environments, enriching the model's adaptability.

The results highlight a potential trade-off between the rapid early learning facilitated by curriculum strategies and the sustained adaptability demonstrated by the baseline approach. Balancing these factors is crucial for achieving optimal performance in diverse and challenging delivery scenarios.

Future Considerations

Moving forward, the reinforcement learning (RL) approach for drone training can be refined by fine-tuning curriculum parameters, striking a balance for adaptability in diverse environments. Hybrid Approaches can be utilized to explore models integrating curriculum learning with non-curriculum methods.

It may be possible to leverage strengths from both approaches to address observed limitations. It helps with scaling to real world environments. This can be done by evaluating the generalization of learned behaviors, considering real-world dynamics and environmental conditions.

Contributions

The main, and only, author is the only contributor to this project. The following was completed:

1. Came up with project idea and proposal
2. Set up Unity ML-Agents environment
3. Designed experiment and environment
4. Wrote the RL training algorithms
5. Results and evaluation

References

[1] Muñoz, Guillem, Cristina Barrado, Ender Çetin, and Esther Salami. 2019. "Deep Reinforcement Learning for Drone Delivery" *Drones* 3, no. 3: 72. https://doi.org/10.3390/drones3030072

[2] Bi, Zhiliang, Xiwang Guo, Jiacun Wang, Shujin Qin, and Guanjun Liu. 2023. "Deep Reinforcement Learning for Truck-Drone Delivery Problem" *Drones* 7, no. 7: 445. https://doi.org/10.3390/drones7070445

[3] Yu, Chao, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. "The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games." arXiv, eprint 2103.01955. Published in 2022.