Unmanned Aerial Vehicle Explorer

Project Report
CMPE260

By

Aryan Jadon (aryan.jadon@sjsu.edu)
Harika Nalam (harika.nalam@sjsu.edu)
Shreya Nimbhorkar (shreya.nimbhorkar@sjsu.edu)
Swathi Anandram (swathi.anandram@sjsu.edu)

Submitted to
Professor Jahan Ghofraniha

December 2022

Table of Contents

List of Figures

Chapter 1. Executive Summary

Different application areas such as traffic monitoring, search and rescue operations, logistics, and military combat engagements are facing difficulties as sometimes the specific area is humanly unreachable. The unmanned aerial vehicle (UAV) is used to navigate the environment autonomously - without human interaction, to perform specific tasks and avoid obstacles. The ability of UAVs to reach inaccessible and high-altitude areas makes them ideal for environmental monitoring.

This project implementation is intended to design and develop a UAV that can be then utilized in the above-mentioned applications. UAV navigation is commonly accomplished using Reinforcement Learning (RL), where agents act as experts in a domain to navigate the environment while avoiding obstacles. It is always important to understand the navigation environment and algorithmic limitations play an essential role in choosing the appropriate RL algorithm. Obstacles vary based on the environment UAV will be operated. UAV first learns in a created learning environment and then uses its learning in the actual environment where it faces real-time complexities and uses these as learning in the future.

Chapter 2. Introduction

In recent times, UAV technology has developed rapidly. UAVs are widely used in the military for monitoring restricted areas, and performing rescue operations. These operations require a technology that functions with maximum accuracy without any casualties and at the same time, these should be cost-effective.

A UAV is an aircraft without any human pilot, crew, or passengers on board. UAVs are a component of an unmanned aircraft system (UAS), which includes adding a controller and a system of communications with the UAV. The flight of UAVs can operate with various degrees of autonomy, such as autopilot assistance, up to fully autonomous aircraft that have no provision for human intervention.

Modern UAVs debuted as an important weapons system in the early 1980s, when engineers developed small flying machines resembling large model airplanes with trainable television and infrared cameras and with target designators for laser-guided munitions, all downlinked to a control station. Rendered undetectable by their small size and quiet engines, these vehicles proved effective in surveillance and target designation.

Chapter 3. Problem Statement

UAVs are becoming increasingly popular due to their ability to operate in a wide range of environments and their potential to perform tasks more efficiently and safely than humans. In some cases, UAVs can also access areas that are difficult or impossible for humans to reach. In a variety of mission scenarios, UAVs are required to safely fly to designated locations and map the location without human intervention. Therefore, finding a suitable method to solve the UAV Autonomous Motion Planning (AMP) problem can improve the success rate of UAV missions to a certain extent.

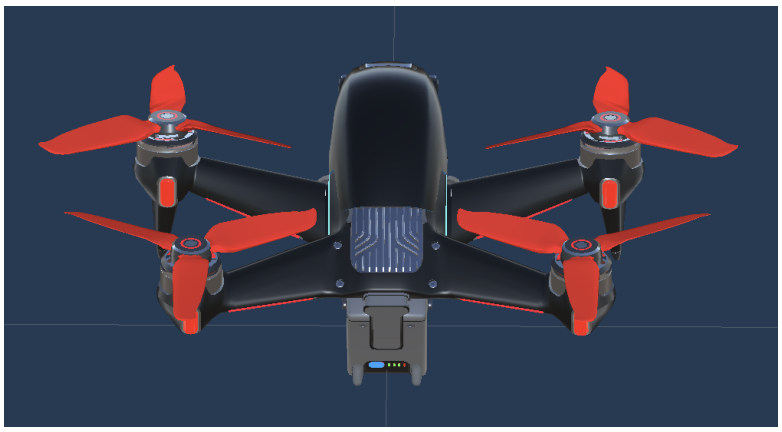


Fig 1. Unmanned Aerial Vehicle Explorer

Chapter 4. Motivation

An unmanned aerial vehicle is a military aircraft used for intelligence, surveillance, ground attacks, electronic countermeasures, and destruction or suppression of enemy air defenses. These can be potentially used in the military as well as in commercial, scientific, police, and mapping applications. They are also efficient in remote observation of hazardous environments that are inaccessible to ground vehicles; they can also be used in aerial photography. Military service is all about the safety of people and nations any technology used is expected to be performed with no casualties.

For UAVs to learn based on the environment and make decisions based on past learning is very important to perform their activities with accuracy. UAVs are also used in mining to learn and understand the deep-down locations where it is really difficult and out of human reach. These are some important applications and motivations for our project to build the best UAV using the best Deep RL algorithms without human intervention.

Chapter 5. Differentiator

The collision-avoidance issue in the context of both crewless ground vehicles and aerial vehicles has been extensively studied in the literature. Many studies use traditional methods to solve the UAV Autonomous Motion Planning (AMP) problem. Yang et al.[1] introduced an improved sparse A* search algorithm which can enhance the planning efficiency and reduce the planning time. Khuswendi et al.[2] designed a path- planning algorithm based on the potential field method and the A* algorithm and compared it with other algorithms in terms of time and distance. Ren et al.[3] proposed a three-dimensional (3D) path-planning algorithm for UAVs and verified the safety and adaptability of the algorithm. These non-learning-based methods have achieved good results when all the information about the environment is known, but they cannot perform well in unknown environments.

Most solutions to the UAVs collision-avoidance problem (UCAP) rely on precise methodologies, such as analytical modeling and optimization techniques. To address its modeling and computational complexity, the existing works, which are based on precise approaches, typically consider some of the UCAP features. However, many factors need to be considered to offer an accurate and valuable model of the UCAP.

- **Detecting obstructions:** The UAVs need to be fitted with onboard sensors in order to be able to detect both stationary and moving objects. Due to many outside circumstances, such as a particular scenario and the autonomy of UAVs, the quantity of these sensors and their precision may be impacted and constrained. For instance, GPS might not function in settings like the enterprise that are indoors. Other sensors, such as radars [4], might be too

bulky, expensive, and energy-intensive. The specific configuration of onboard sensors in UAVs then depends on the scenario and application.

- **Sensor Errors:** Object detection onboard sensors are not error-free. They all have minor precision inaccuracies that environmental factors could impact. For instance, the weather influences GPS error, which has a Gaussian distribution [5].
- **Complex Control:** The movement of the UAV is controlled by several factors, including direction, velocity, and acceleration. Additionally, these variables are highly dependent on outside influences like wind speed.
- **Diverse Approaches:** Path planning and sensing and avoiding [6] methods are two distinct approaches to solving UCAP. While sensing and avoiding (online) approaches predict the movement of the UAVs for small time steps based on the surroundings, path planning solutions compute the trajectories of the UAVs offline. Sensing and avoidance techniques are more adaptable and suitable for a larger variety of situations. Only in situations where the environment is largely unchanging for the duration of the UAVs' mission can path planning work well. The main disadvantage of online approaches is that they often require the UAV to execute the algorithm (for example, due to latency restrictions), which could result in high computational complexity and energy consumption.

In light of those mentioned above, the UCAP necessitates a high domain knowledge, and its modeling results in difficult or impossible optimization algorithms. The research community has recently given machine learning approaches much attention since they are particularly appealing for tackling UCAP and can help solve these issues [7]-[12].

Choi and Cha [13] presented a thorough analysis of ML-assisted solutions for autonomous flight. They concentrate mainly on the control technique for UAVs and object recognition. The authors conclude that, while some outstanding issues still require careful attention, ML is a promising approach to enabling stable flight in uncertain situations. One of them is that ML is only applied to some of the UCAP's problems in the existing studies for autonomous flight. Therefore, comprehensive solutions that address most real-world issues and are appropriate for a more extensive range of circumstances are needed. Furthermore, big data sets are required for training using the current technologies. In this regard, they promote fresh ideas with lighter training that are less data-hungry.

Deep reinforcement learning (RL) has made it possible to solve complex robotics problems using neural networks as function approximators. However, the policies trained on stationary environments suffer in terms of generalization when transferred from one environment to another.

In our project, we suggest using ML-Agents Library methods and a straightforward but effective deep reinforcement learning (DRL) approach to self-train in various situations. Our ideas apply to a wide range of circumstances, but they require smaller data sets to converge and provide better results.

Chapter 6. Methodology

We want to focus on creating a solution-oriented UAV application created out of Reinforcement and Deep Learning Techniques. The features of the UAVs will help in navigating the environment autonomously - without human interaction, perform specific tasks and avoid obstacles. We have used unity and unity ML agents to build UAVs. Unity offers tools to create virtual simulated environments with customizable physics, landscapes, and characters. Unity ML-Agents is a new plugin for the game engine Unity that allows us to create or use pre-made environments to train our agents.

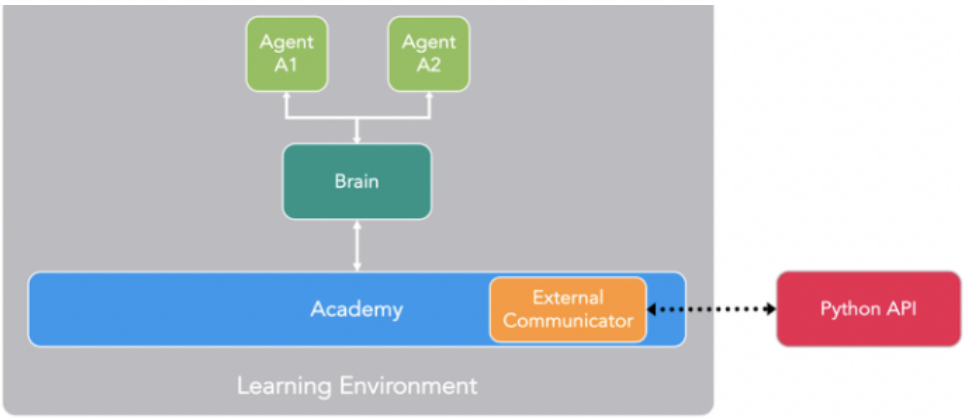


Fig 2:Components of Unity ML agent

The three important components of the Unity ML agent are

- Learning Environment
- Python API
- External Communicator

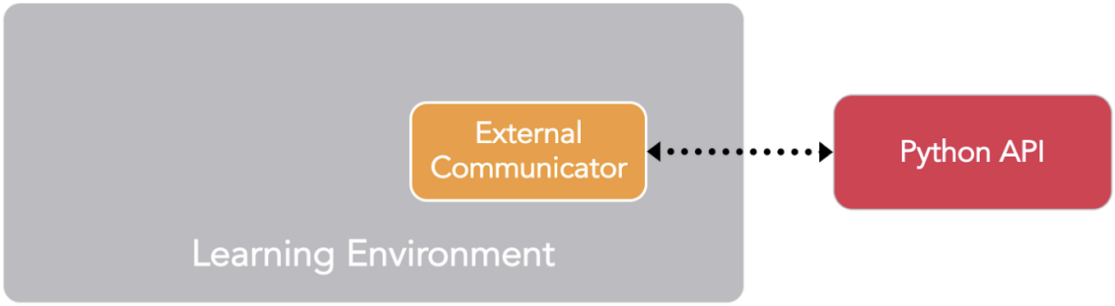


Fig 3. Inside Learning Component

The first is the Agent, the actor of the scene. We aim at training the actor by optimizing his policy (that will tell us what action to take at each state) called Brain. Finally, there is the Academy, this element orchestrates agents and their decision-making process. Think of this Academy as a maestro that handles the requests from the python API. Academy will be the one that will send the order to our Agents and ensure that agents are in sync:

- Collect Observations
- Select your action using your policy
- Take the Action
- Reset if you reached the max step or if you're done.

Learning Component, which contains the Unity scene and the environment elements. The important elements in Learning Environment are the Agent, the actor of the scene, and the academy, this element orchestrates agents and their decision-making process. This Academy acts as a master that handles the requests from the python API. Python API is the component we load our RL algorithms. This API is used to launch the agent to train and test with the learning environment and an external communicator is used to communicate with the environment.

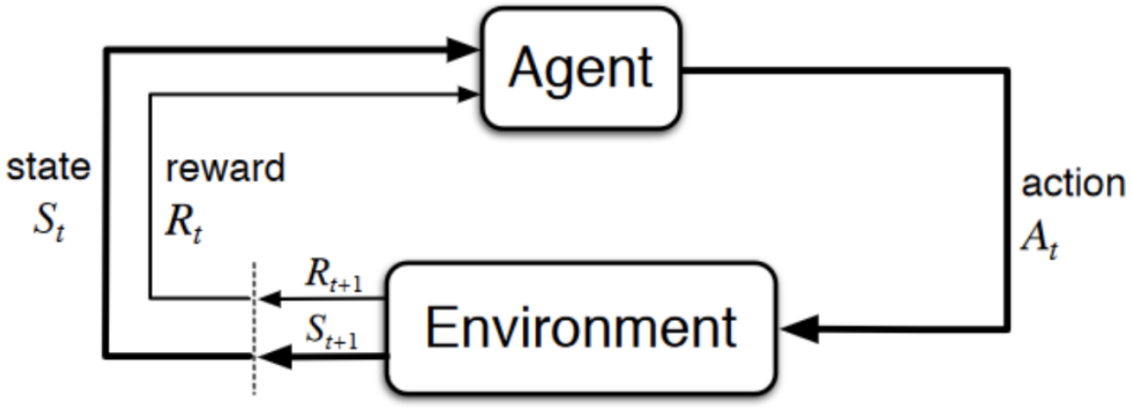


Fig 4. Reinforcement Learning Architecture

Our agent receives state S0 from the environment — we receive the first frame of our game (environment).

- Based on the state S0, the agent takes an action A0 — our agent will move to the right.
- The environment transitions to a new state S1.
- Give a reward R1 to the agent — we're not dead (Positive Reward +1).

This RL loop outputs a sequence of state, action, and reward. The goal of the agent is to maximize the expected cumulative reward.

As a first step UAV has to learn to fly in the learning environment avoiding obstacles. For this, we have created an environment in which 'X' is an obstacle for our agent which is UAV. The important components in the unity Reinforcement learning environment are Behavior parameters and Decision requester. The behavior parameter has action, model, and behavior type are the important parameters that are tuned according to the problem we are trying to solve. The model we used in our learning environment is the 'Rotor Model'. There are three different types

of behavior types - **Default, Heuristic, and Inference**. The default is the learning behavior which is used when the agent has to learn the environment. Heuristic in this type of behavior type programmers thinks of ways AI should behave and is added to it. In the Inference behavior type learned model is used after training and no changes are made to the model in this type. The other important component is Decision Requester which provides a convenient and flexible way to trigger the agent decision-making process, without this it will be just a manual operation. The learning environment, we designed consists of 'x' the obstacles and the camera which follows the agent.

During training, the external Python training process communicates with the Academy to run a series of episodes while it collects data and optimizes its neural network model. When training is completed successfully, the trained model file is added to your Unity project.

The training process in unity -

- Calls the Academy's OnEnvironmentReset delegate.
- Calls the **OnEpisodeBegin**() function for each Agent in the scene.
- Gathers information about the scene. This is done by calling the **CollectObservations**(VectorSensor sensor) function for each Agent in the scene, as well as updating their sensor and collecting the resulting observations.
- Uses each Agent's Policy to decide on the Agent's next action.
- Calls the **OnActionReceived**() function for each Agent in the scene, passing in the action chosen by the Agent's Policy.
- Calls the Agent's **OnEpisodeBegin**() function if the Agent has reached its Max Step count or has otherwise marked itself as **EndEpisode**().

The environment designed for UAVs is free from external factors like wind, speed, lighting conditions, and moving objects. These external factors make the environment complex for the agent to learn. Transfer of knowledge is another risk where an agent is not guaranteed to perform similarly because of the difference in the environment's nature and conditions. These are the main risks involved in our project.

The rewards of our project are - the UAV can move in the intended direction via an agent which controls its movement by processing states from the environment. For this project, we used the Proximal Policy Optimization(PPO) algorithm. This algorithm uses a novel objective function called the "**Clipped surrogate objective function**" as follows :

$$L^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)]$$

Where,

- θ is the policy parameter
- E^t denotes the empirical expectation over timesteps
- rt is the ratio of the probability under the new and old policies, respectively
- A^t is the estimated advantage at time t
- ε is a hyperparameter, usually 0.1 or 0.*2*

This function will constrain the policy change in a small range using a clip. The central idea of Proximal Policy Optimization is to avoid having too large policy updates. To do that, we use a ratio that will tell us the difference between our new and old policies and clip this ratio from 0.8 to 1.2. Doing that will ensure that our policy update will not be too large. PPO improves the stability of the Actor training by limiting the policy update at each training step.

Chapter 7. Implementation and Results

7.1 Results

Recorded project demo video can be viewed using the link -

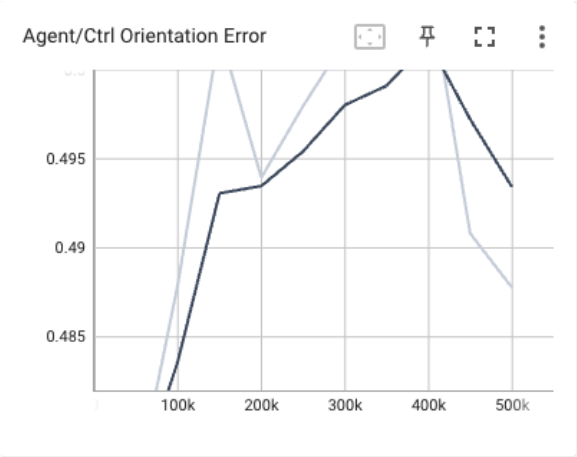
https://drive.google.com/file/d/1B82DRj2fhIglXJuTbn6se6TJknsprw8o/view?usp=share_link

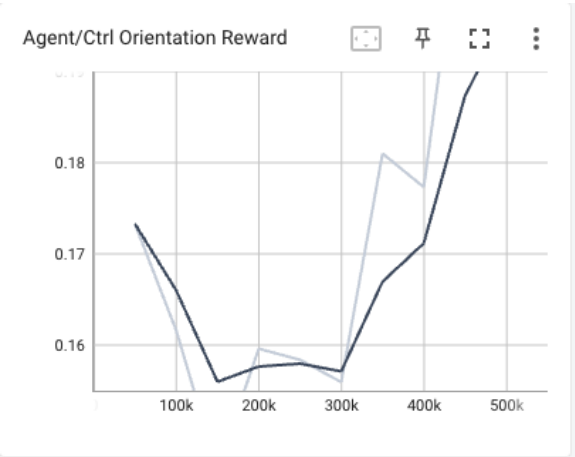
```yaml
default_settings:

  trainer_type: ppo
  hyperparameters:
    batch_size: 2048
    buffer_size: 20480
    learning_rate: 0.0003
    beta: 0.005
    epsilon: 0.2
    lambd: 0.95
    num_epoch: 3
    learning_rate_schedule: linear
  network_settings:
    normalize: false
    hidden_units: 256
    num_layers: 2
    vis_encode_type: simple
  reward_signals:
    extrinsic:
      gamma: 0.99
      strength: 1.0
  keep_checkpoints: 100
  time_horizon: 1000
  summary_freq: 10000
  max_steps: 50000000

behaviors:

  RotorControl:
    network_settings:
      hidden_units: 32
      num_layers: 3
    max_steps: 100000000

  VisualPilot:
    reward_signals:
      gail:
        strength: 0.01
        gamma: 0.99
        demo_path: VisualPilot.demo
    behavioral_cloning:
      demo_path: VisualPilot.demo
      strength: 0.5
      steps: 10000000

```

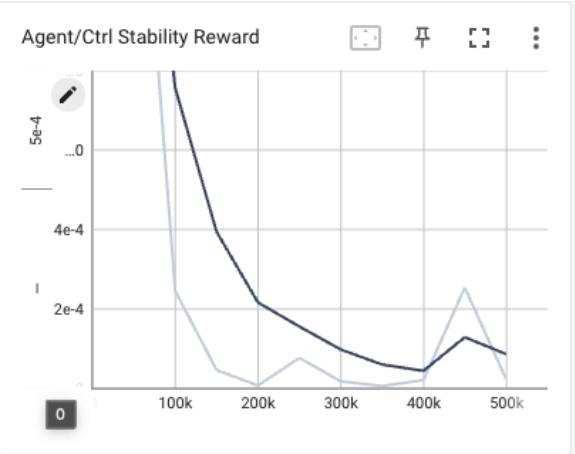
Fig 5. Training Configuration YAML file

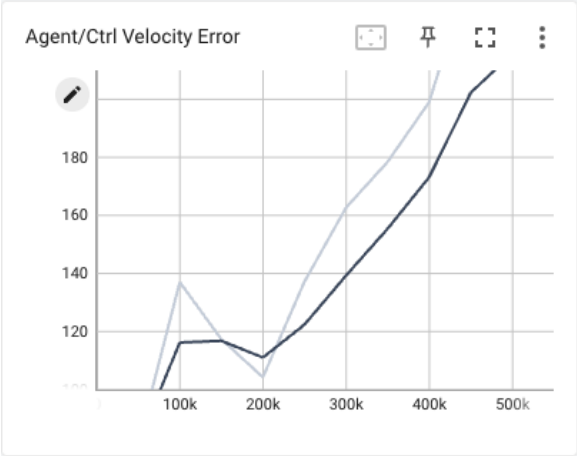
7.2 TensorBoard Logs

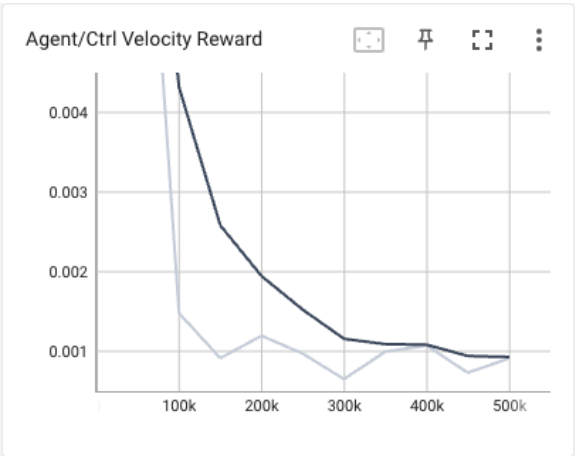
Agent:



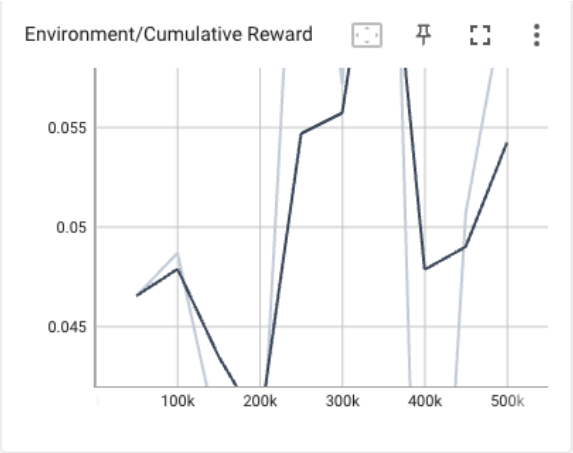


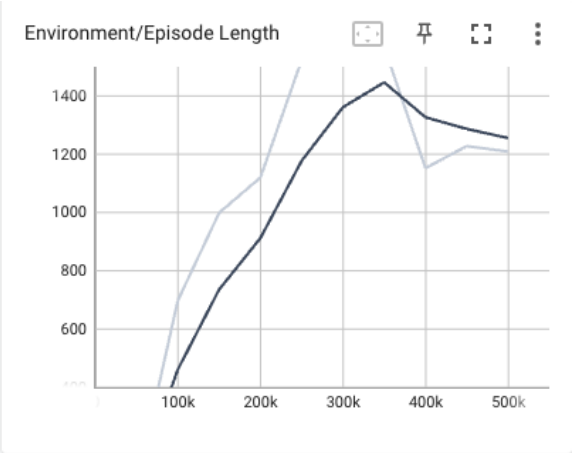




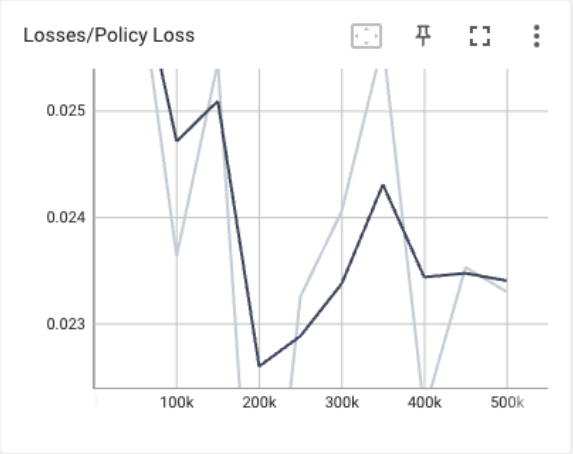


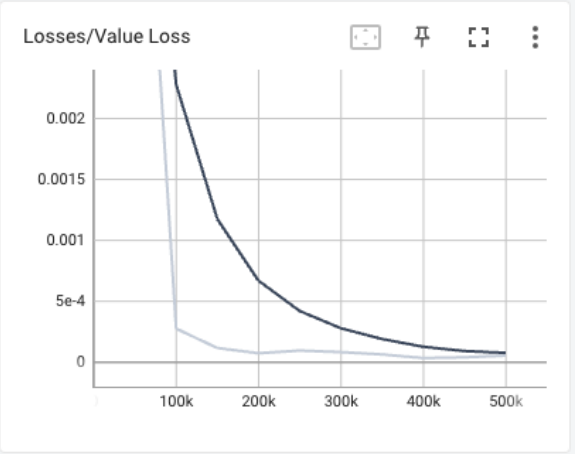
Environment:



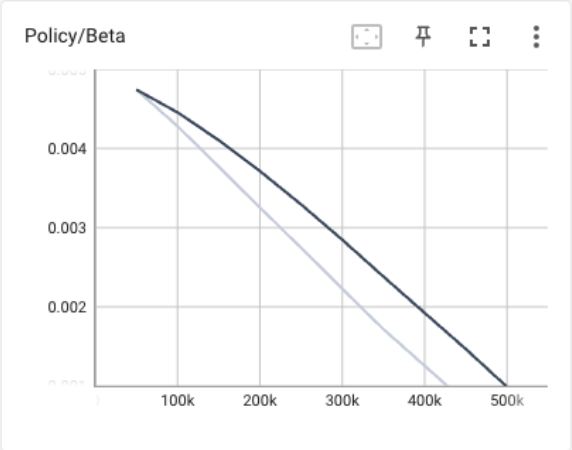


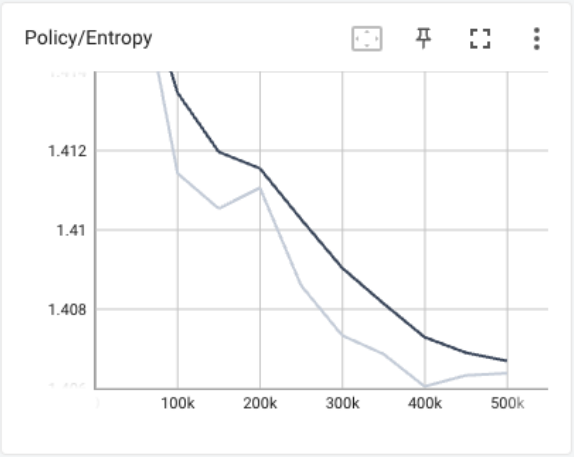
Loss:

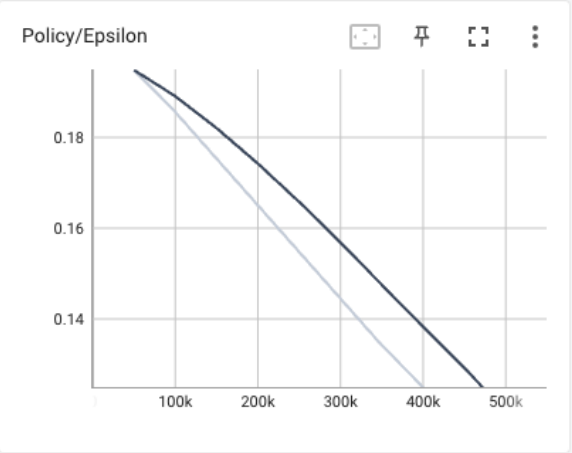




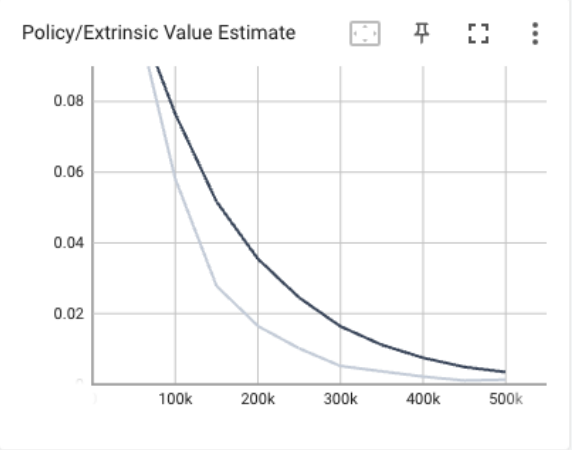
Policy:

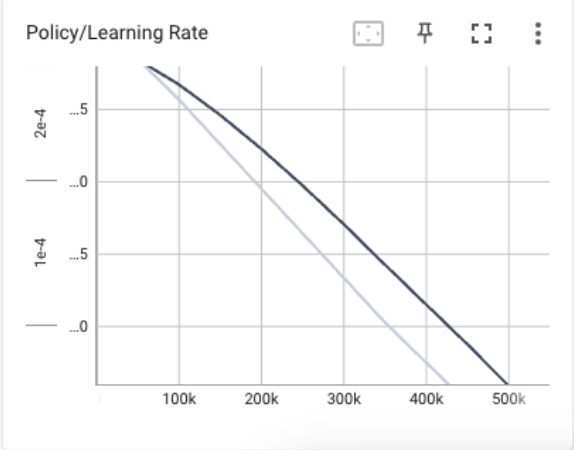

Policy/Beta
0.004
0.003
0.002
100k
200k
300k
400k
500k


Policy/Entropy
1.412
1.41
1.408
100k
200k
300k
400k
500k


Policy/Epsilon
0.18
0.16
0.14
100k
200k
300k
400k
500k


Policy/Extrinsic Reward
0.055
0.05
0.045
100k
200k
300k
400k
500k


Policy/Extrinsic Value Estimate
0.08
0.06
0.04
0.02
100k
200k
300k
400k
500k


Policy/Learning Rate
2e-4
1e-4
100k
200k
300k
400k
500k

7.3 Team Contribution

Initially, we all worked on domain and topic finalization where we all read various research papers. After finalizing the topic we started learning Unity's basic frameworks - basic movement scripting, understanding collisions, rigid bodies, and physics. Next, we all learned about ML agents and went through various projects using ML agents and imitation learning using ML agents.

Aryan Jadon

- Performed Installation Environment & Project Setup - Unity, Machine Learning, and PyTorch Dependencies.
- Worked on Environment and Training/Testing UAV with just a single raycast detection.
- Performed testing on UAV with hierarchical setup and sensors with depth textures

Harika Nalam

- Designed Machine Learning Agents
- Worked on UAV MLAgents-hyper params.
- Performed Testing Work of Unity Inference Engine on UAV

Swathi Anandram

- Designed Learning Environment.
- Worked on Environment and Training/Testing UAV with just a single raycast detection.
- Performed Testing on UAV with hierarchical setup and sensors with depth textures

Shreya Nimbhorkar

- Performed Testing by Using Executable Environment.
- Worked on UAV MLAgents-hyper params.
- Performed Testing Work of Unity Inference Engine on UAV

Link for the project-planning excel sheet: Reinforcement-Learning-Project-Planning.xlsx

Chapter 8. Conclusion

We successfully built the Unmanned Aerial Vehicle Explorer using Unity and trained the agent which travels in the environment by avoiding obstacles. This will help reduce manual tasks and save resources by training and testing policies in real-world environments.

In the future, this project can be enhanced by working on its gamma values. The gamma value used in this project is 0.99. However, for better results, different gamma values can be set along with other hyperparameters like beta, epsilon, lambda, and batch size.

Apart from PPO, different existing algorithms like Soft Actor-Critic(SAC) can also be used to implement this project. SAC is an off-policy actor-critic deep RL algorithm based on the maximum entropy reinforcement learning framework. Here, the actor aims to maximize the expected reward while also maximizing entropy. That is, to succeed at the task while acting as randomly as possible. Here the policy is incentivized to explore more widely while giving up on clearly unpromising avenues. The policy can capture multiple modes of near-optimal behavior. In problem settings where multiple actions seem equally attractive, the policy will commit equal probability mass to those actions.

Chapter 9. Appendix

a. Code

Unity Project Files can be found at -

https://github.com/aryan-jadon/CMPE-260-Reinforcement-Learning-Project.

b. References

[1] Zijian HU, Xiaoguang GAO, Kaifang WAN, Yiwei ZHAI, Qianglong WANG, Relevant experience learning: A deep reinforcement learning method for UAV autonomous motion planning in complex unknown environments, Chinese Journal of Aeronautics, Volume 34, Issue 12, 2021, Pages 187-204, ISSN 1000-9361, https://doi.org/10.1016/j.cja.2020.12.027.

[2] Tong GUO, Nan JIANG, Biyue LI, Xi ZHU, Ya WANG, Wenbo DU, UAV navigation in high dynamic environments: A deep reinforcement learning approach, Chinese Journal of Aeronautics, Volume 34, Issue 2, 2021, Pages 479-489, ISSN 1000-9361, https://doi.org/10.1016/j.cja.2020.05.011.

[3] C. Wang, J. Wang, Y. Shen and X. Zhang, "Autonomous Navigation of UAVs in Large-Scale Complex Environments: A Deep Reinforcement Learning Approach," in IEEE Transactions on Vehicular Technology, vol. 68, no. 3, pp. 2124-2136, March 2019, doi: 10.1109/TVT.2018.2890773.

[4] Y. K. Kwag and C. H. Chung, "UAV based collision avoidance radar sensor", *Proc. IEEE Int. Geo-Sci. Remote Sens. Symp. (IGARSS)*, pp. 639-642, Jul. 2007.

[5] T.-H. Yi, H.-N. Li and M. Gu, "Experimental assessment of high-rate GPS receivers for deformation monitoring of bridge", *Measurement*, vol. 46, no. 1, pp. 420-432, Aug. 2012.

[6] B. M. Albaker and N. A. Rahim, "A survey of collision avoidance approaches for unmanned aerial vehicles", *Proc. Int. Conf. Tech. Postgraduates (TECHPOS)*, pp. 1-7, Dec. 2009.

[7] S. Y. Choi and D. Cha, "Unmanned aerial vehicles using machine learning for autonomous flight; state-of-the-art", *Adv. Robot.*, vol. 33, no. 6, pp. 265-277, 2019.

[8] A. Singla, S. Padakandla and S. Bhatnagar, "Memory-based deep reinforcement learning for obstacle avoidance in UAV with limited environment knowledge", *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 1, pp. 107-118, Jan. 2021.

[9] S.-Y. Shin, Y.-W. Kang and Y.-G. Kim, "Obstacle avoidance drone by deep reinforcement learning and its racing with human pilot", *Appl. Sci.*, vol. 9, no. 24, pp. 5571, Dec. 2019.

[10] P. Fraga-Lamas, L. Ramos, V. Mondéjar-Guerra and T. M. Fernández-Caramés, "A review on IoT deep learning UAV systems for autonomous obstacle detection and collision avoidance", *Remote Sens.*, vol. 11, no. 18, pp. 2144, 2019.

[11] D. Wang, T. Fan, T. Han and J. Pan, "A two-stage reinforcement learning approach for multi-UAV collision avoidance under imperfect sensing", *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3098-3105, Apr. 2020.

[12] K. Wan, X. Gao, Z. Hu and G. Wu, "Robust motion control for UAV in dynamic uncertain environments using deep reinforcement learning", *Remote Sens.*, vol. 12, no. 4, pp. 640, 2020, [online] Available: http://dx.doi.org/10.3390/rs12040640.

[13] S. Y. Choi and D. Cha, "Unmanned aerial vehicles using machine learning for autonomous flight; state-of-the-art", *Adv. Robot.*, vol. 33, no. 6, pp. 265-277, 2019.

[14] Simonini, T. (2020, March 25). An introduction to unity ML-agents. Retrieved December 6, 2022, from https://towardsdatascience.com/an-introduction-to-unity-ml-agents-6238452fcf4c

[15] Unity-Technologies. (n.d.). Unity-Technologies/ML-Agents: The Unity Machine Learning Agents Toolkit (ML-agents) is an open-source project that enables games and simulations to serve as environments for training intelligent agents using deep reinforcement learning and imitation learning. Retrieved December 6, 2022, from https://github.com/Unity-Technologies/ml-agents

[16] Simonini, T. (2020, March 25). Diving deeper into unity-ml agents. Retrieved December 6, 2022, from https://towardsdatascience.com/diving-deeper-into-unity-ml-agents-e1667f869dc3

[17] Simonini, T. (2020, March 25). Unity-ML agents: The mayan adventure. Retrieved December 6, 2022, from https://towardsdatascience.com/unity-ml-agents-the-mayan-adventure-2e15510d653b

[18] Simonini, T. (2022, June 10). Proximal Policy Optimization (PPO) with sonic the hedgehog 2 and 3. Retrieved December 6, 2022, from https://towardsdatascience.com/proximal-policy-optimization-ppo-with-sonic-the-hedgehog-2-and-3-c9c21dbed5e