Reinforcement Learning with Graph Neural Networks for Drone Collision Avoidance

Design Document

sdmay21-proj033
Swapp Lab / Iowa State University

Advisors
Deepak George Thomas
Dr. Ali Jannesari

Team Members/Roles
Karter Krueger (RL Team)
Joshua Kalyanapu (Sim Team)
Matthew Phipps (Sim Team)
Rithvik Menon (RL Team)
Ryan Howe (RL Team)
Thamir Al Harthy (Sim Team)
Zachary Mass (RL Team)

sdmay21-33@iastate.edu
https://sdmay21-33.sd.ece.iastate.edu/

Revised: April 25, 2021

Executive Summary

Development Standards & Practices Used

Our team followed an Agile development method throughout the last two semesters to organize tasks and development work.
Aside from this, the project does not have normal engineering standards because it is a research project meant to test the use of CNN and GNN in a simulated environment. Normal engineering standards like communication protocols, etc. really don't apply.

Summary of Requirements

Neural Networks

- Create a GNN that utilizes a double deep Q learning protocol
- Create a CNN that utilizes a double deep Q learning protocol

Training and Learning

- Optimize training environment by speeding up the rate at which we can train the model
- Gather and display data in ways to help garner insight into how the model is learning in the environments it is presented

Applicable Courses from Iowa State University Curriculum

- Math 207 Matrices and Linear Algebra
- Math 314 Graph Theory
- CS 575 Computational Perception
- CS 518 Computational Geometry
- CS 577 Applied Advanced Techniques for CS
- CS 574 Machine Learning
- CS 572 Artificial Intelligence

New Skills/Knowledge acquired that was not taught in courses

- Reinforcement Learning (Network Architectures, Reward Policy, Gym, etc)
- Deep Learning (Convolutional Neural Network and etc)
- Docker (Simulation Environment)
- Unreal Engine 4 (Simulation Environment)
- Airsim (Simulation Learning with Drone)

Table of Contents

List of figures/tables/symbols/definitions

Figure 1. Diagram of the Drone System

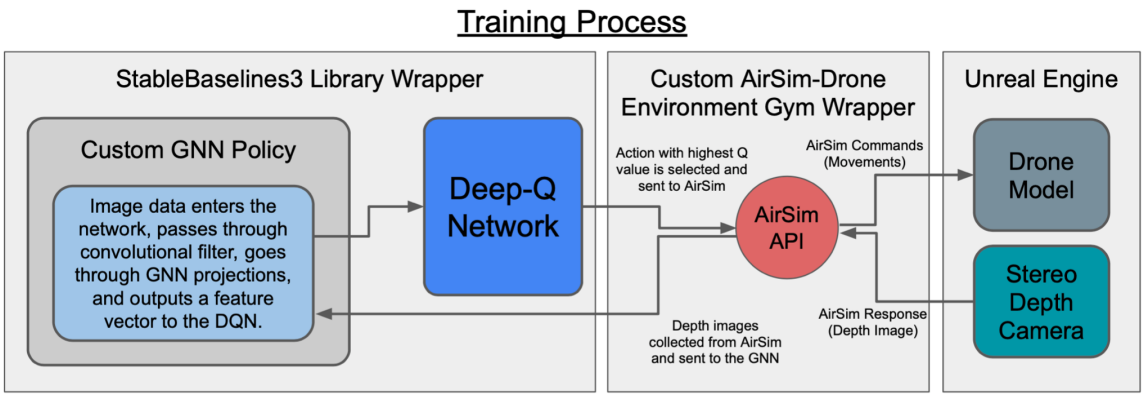
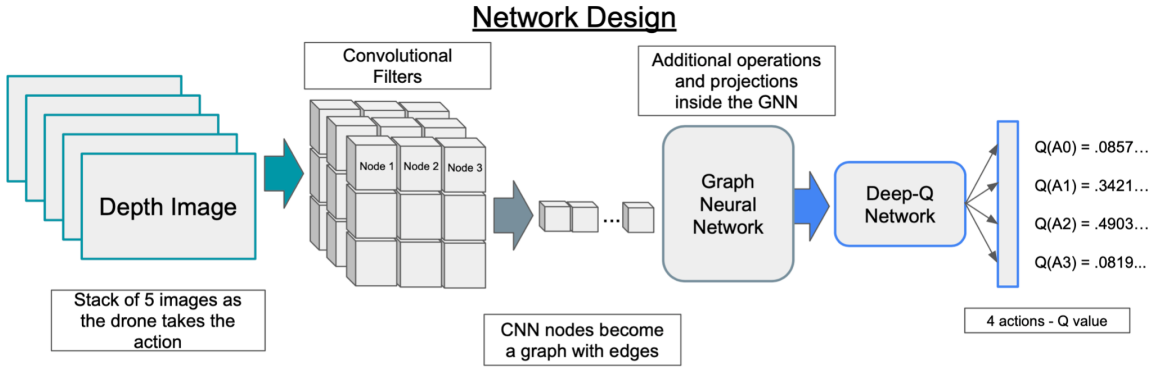


Figure 2. Diagram of the System Architecture



Term	Definition
GNN	Graph Neural Network
CNN	Convolutional Neural Network
RL	Reinforcement Learning
Gym	OpenAI Gym Environment
DL	Deep Learning

1 Introduction

1.1 Acknowledgement

It is acknowledged that the SwAPP Lab, directed by advisor Dr. Ali Jannesari is the key source providing the GPU that the simulation is primarily training on, and the drone equipment that will be used for testing and executing the real-world demo of our project.

1.2 Problem and Project Statement

The problem the team was faced with for this project was designing a reinforcement learning (RL) model that will be trained in an environment in which the networks must control a drone that will explore an environment while performing obstacle avoidance. Furthermore this RL model needs to be designed so that it can utilize two different network architectures, a convolutional neural network (CNN) policy and a graph neural network policy (GNN). This is because our advisor's end goal with this project is to use the networks we develop to formulate a paper on the comparisons between the CNN and GNN and how well the drone handles new environments depending on which one is in control. To allow ease of use our advisor also tasked us with getting the project to run in docker.

1.3 Operational Environment

The drone for this project is expected to be able to avoid obstacles while exploring an environment. We will not assume certain times of day nor weather affecting lighting conditions in the final product. This means that when it comes to training, the team needs to be careful that it is not over training in one environment or another as this will result in overfitting and our model will not be able to work well in all environments.

1.4 Requirements

The project can be decomposed into a small set of key functional and nonfunctional requirements.

Functional requirements

- Ability to fly
- Ability to carry a microcontroller and GPU

Nonfunctional requirements

- Should be able to fly and avoid objects in the environment
- Neural network should be small enough to fit on our microcontroller

1.5 Intended Users and Uses

With the nature of this being strictly for a research project, we do not have a specific end user or use in mind. One obvious user of our project will be our advisor Deepak who

intends to expand on our work and use it to write a paper analyzing GNNs versus CNNs in the context of the environment in which we are using them for this project.

1.6 Assumptions and Limitations

Assumptions

- The drone will fly in an open area outside where there aren't safety concerns.
- There will be a kill switch on the drone to take over if there is concern.
- The drone will fly in a day environment with proper lighting to see obstacles and persons, but not too bright to blind the camera.
- The Docker container will be ran on Ubuntu for Windows Utilizing the WSL framework

Limitations

- The drone cannot be flown indoors, so the weather must meet the right conditions.
- The simulation training takes a long time to train, so development can be slow.
- The outdoor environment must be similar to the simulation for easier transfer learning.
- The drone is limited to a short flight time outdoors due to the battery size.
- Computing power for training the drone is quite limited due to the intensity of what is needed to run the simulation and the amount of harddrive space needed.
- The Docker container must be ran and build on the WSL framework unless users build a new image

1.7 Expected End Product and Deliverables

There are two main deliverables for this project, the double deep-Q learning CNN policy network as well as the double deep-Q learning GNN policy network. It is quite elementary to observe however that despite having two main deliverables, there are a lot of similarities between the two. Specifically when we consider the makeup of the two networks, the only thing that is different between the two is the underlying architecture. It is for this reason that our deliverables can be more practically broken into three separate things, creating software for handling a double deep Q learning approach, implementing a CNN policy and implementing a GNN policy.

2 Project Plan

2.1 Task Decomposition

This project can be divided into four main tasks that we used to divide our team into subgroups. One task is the creation of the CNN and the GNN. The second task is working on what heuristics to use and actions for our neural networks so that we can try to optimize how the models behave in the environment. The third task is to have the project dockerized so that it can be used on any machine. The final task is managing the training environment as well as trying to streamline the training process to speed up the rate at

which our networks train. For our group this has mainly been the task of trying to create a docker container that contains the training environment so that we can easily train multiple models in parallel.

2.2 Risks And Risk Management/Mitigation

Current risks include drone collisions when deployed, reward function not producing desired behavior in all environments, and overfitting of the neural networks. Drone collisions are expected during training, due to the nature of reinforcement learning, so training will always be done virtually in order to save money and effort. Since almost all these issues are tied to possible inadequacies of the neural networks, the team has placed a strict emphasis on analyzing the viability of the reward function we use as well as testing the trained network in diverse environments to check for overfitting.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

- Virtual environment setup and usable on all team members workstations
    - Clone AirSim
    - Clone Unreal Engine
    - Install Anaconda or Pip
    - Clone Project Repo
- Develop reward function to help drone learn efficiently
    - Research different reward functions for vision based RL
    - Implement new function(s)
    - Run and test until satisfactory results are obtained
- Develop Neural Networks
    - Research implementations of double deep Q learning approaches
    - Research CNN network design and motivation
    - Research GNN network design and motivation
    - Implement CNN and GNN utilizing double deep Q learning
- Create Docker image for project
    - Research docker implementation with airsim and unreal
    - create docker image for the container

2.4 Project Timeline/Schedule

Setting up Unreal Engine Environment	October 2020

Learning Reinforcement Learning Strategies	October 2020
Training the proposed algorithm on the environment and testing novel environments	November 2020 - January 2021
Switched main focus of our project. (See Appendix II)	January 2021
Configure and design the reward function.	February 2021 - April 2021
Get Gym integrated into the project and running without errors.	March 2021
Completed design and training of the CNN	March 2021 - April 2021
Completed design of the GNN	March 2021 - April 2021

2.5 Project Tracking Procedures

Our team utilized GitLab, and discord to communicate with each other throughout the course of the project. Git was used as our version control system and as our wiki where we share findings and information. Zoom was used for video-based conferencing with our project supervisor along with Group me being used for text communication among the team. Throughout the course of the second semester of our project the team made the decision to switch from the mixture of Zoom and Group me to using Discord. This was primarily a result of Discord being able to handle the features of both Group Me and Zoom at the same time. It is due to this centralized tool that the team was able to accomplish a lot more than it had initially been achieving using the old framework of communication.

2.6 Personnel Effort Requirements

Task	Description	Project Effort and Time
Development of environments for testing	The simulation subteam for our project needs to develop robust environments for our drone to train in.	This task will end up taking quite a bit of effort on the part of the team members working on it. This is primarily a factor of the none of the team members having prior experience with Unreal Engine. Estimated 40 man hours.
Creating the Reward Function for the Drone	The reinforcement learning subteam needs to develop a reward function for the neural networks to utilize for training.	This task required a large amount of time since the team in general did not have extensive RL experience. Estimated 10 man hours.
Designing and implementing the CNN and GNN	This task is mainly concerned with developing the network architectures that will be trained and analyzed.	Much like almost all the tasks on this project, the team has little to no experience in this field and thus has required extensive amounts of research on the subject in order to start accomplishing tasks. Estimated man hours 80-90 hours.
Training the proposed algorithm on the environment and testing novel environments	Running the python script on the simulation environment created in order to train the neural network.	This will not require a significant amount of interaction, and will mostly be a computer executing the code hence the low amount of person-hours. It is noteworthy that the expected time to train the model is upwards of 3-5 days.

2.7 Other Resource Requirements

This project additionally required the configuration of a private server that was procured and managed by the Swapp lab. Alternatively, a personal computer can be used if it has

sufficient computing power, such as a GPU for training the neural networks with the simulation environment. Hard Drive space is also a concern at the full size of the project clocking in at about 130 Gigabytes. The team implemented a dockerized version of our simulation environment which allowed us to shrink the space requirements down to 55 Gigabytes.

2.8 Financial Requirements

The drone equipment and server for training were provided by our advisor's lab. Although no funds were expended throughout the course of this project for reference the project stood to benefit greatly from more investment in more servers or computing resources that could be utilized by the team. This was most likely a limitation of COVID-19 as if there was a lab the team could have used many of the issues regarding computing requirements would have been mitigated.

3 Design

3.1 Previous Work And Literature

At the start of the first semester of our project, our task was to develop a neural network that could be summed up as "Object Tracking using Reinforcement Learning". It is important to note that all of the first semester and a portion of the second semester was focused on this different goal. Eventually our advisor came to our group and told us that this goal is not the one we should be working on, and proposed a different project that we could work on. This resulted in large amounts of work done by our team being scrapped, though it is important to note that not all of it was. Some things that were not scrapped were that our task would still include an environment nearly identical to the one we are currently working in, that being a virtually simulated drone that will be driven via reinforcement learning. Another static part of the project was that we would be utilizing deep Q learning, however we did switch from deep Q learning to double deep Q learning. This is not a very large change as the added implementation is almost negligible.

Due to the nature of this project being involved with research as well as touching on topics almost none of the team was familiar with, there was a large body of reference material that the team utilized. Concerning the network architecture the team made extensive use of two different textbooks on the subject of deep learning: these include "Hands-on Machine Learning with Scikit-Learn and Tensorflow" by Aurelien Geron, and "Deep Reinforcement Learning in Action" by Alexander Zai, Brandon Brown. Some other noteworthy resources include the blog AI Summer (theaisummer.com), the Reinforcement Learning Discussion discord channel, and materials that are offered by Deeplizard.com.

Regarding resources used for developing the docker container the team utilized several sources to get the project running in a docker container. Mainly the wiki page from airsim

would be used which would point to several other sources needed to get all of the different]technologies to cooperate. This was also a difficult task because the team was mostly unfamiliar with docker and the tutorials found were deprecated. This resulted in many hours testing with different ideas as to why the container would not run with the project.

3.2 Design Thinking

Define Stage

Since we started a new goal, the decisions made for which technologies and libraries to use were given to us, since it is a research based project. So, there was not much discussion for our define stage. The point-of-view can be stated as "Design a model for a UAV that can follow a mobile object." To elaborate on this point-of-view the model design referred to is a Machine Learning model, the UAV can be any unmanned aerial vehicle, and the mobile object can be any object that can move. The UAV will be trained to follow the object in a safe manner through deep reinforcement learning.

Ideate Stage

Since our project is a research based project we did not have say in what ideas we were to implement to solve the point-of-view statement. Ultimately though our advisor's decision on using CNN and GNN allowed us to build on the ideas of the research and encourage different approaches.

Prototype Stage

We have developed rough models to share with our advisor. Our models include the implementation of the GNN (Graph Neural Network) in the SB3 (StableBaselines3) library, improved the simulation environment for the drone to be tested, and established the reward policy for the reinforcement learning. Through frequent iterations, we have also managed to improve the rough models. Also, we understood certain parts of GNN from learning from the iteration and failures. We managed to prototype early and often by using a variety of prototyping formats.

3.3 Proposed Design

The design we proposed at the end of the first semester of this project is irrelevant as of now and thus has been omitted. This is tied back to what was stated earlier with how our project was changed in its entirety at some point in February 2021. If the reader would like to see what the design of the project looked like at the end of last semester please refer to Design Document V2 on our teams website.

In February our new design did need to share many of the same features as the previous version did, so many parts, though importantly not all parts, were reused. The most noteworthy feature that was reused was the environment set up. Since our task was largely similar just with a change in the goal for the neural network, not much was changed aside from the implementation of the reinforcement learning algorithm. So our main design considerations were as follows...

- Use a Neural Network to drive a drone in a virtual environment that can avoid obstacles
    - The Neural Network would need to be implemented utilizing a CNN policy and GNN policy and double deep Q learning
- The virtual environment is made up of a mixture of UE4 and Airsim
    - UE4 is the environment itself
    - Airsim simply acts as a "controller" of sorts so that network can drive the drone

Since this senior design project is largely based on a research project where the author seeks to garner the results between two different implementations of a project, what to use for our project and how to have all the parts interface with each other was largely handed to us. Like in many parts of this document, it is important to recall that the team has almost no experience with deep learning; this made a large part of this project understanding implementations of the networks we were tasked with implementing, as well as how to use and the limits of deep learning in general.

3.4 Technology Considerations

There are a plethora of software for making and training machine learning models. Our team decided to use PyTorch since it is a very simple to use machine learning framework. Not to mention it is very popular and widely used lending to a large amount of documentation on the tool being developed. Another very important technology we made use of for our project is Stablebaselines_3 (SB3) reinforcement learning library. SB3 is a library that attempts to make highly optimized, and highly standardized, neural networks that can be easily shaped and adapted to any Gym environment. This tool was very helpful for standardization of the project since it allowed us to very conveniently highlight the differences between the CNN implementation and the GNN implementation as well as provided an easy to use library of highly optimized state of the art CNN network implementations.

With regard to training we utilized UE4, Airsim, and Gym. Since this was largely expanded upon in section 3.3 please refer to that section for more information on the purpose of these tools.

3.5 Design Analysis

The design we have implemented for this project is very modular. This can easily be seen by the separation of concerns presented in Figure 2. Since each of these tools are communicating with each other at a very bare minimum level it has been easy for us to find bugs in our implementation of different design aspects. A great example of this was when we were working on the Gym implementation we ran into some weird problems regarding how Gym handles interfacing with the GNN / CNN. Since the errors were very clearly coming from the communication between these two aspects we were able to quickly narrow our focus on a small set of places the problem could be arising from. Without this separation of concerns the team does not think we would have been able to accomplish the project on time. When it comes to the design of the CNN and GNN

themselves we are not permitted to get into very specific detail and apologize for that at this time.

3.6 Development Process

The development process of this project drew on much of the routines in the Agile development practice. Nowhere is a better example of this found than the weekly meetings we held with our advisor. At this meeting we would discuss plans for the next week as well as each team member would have a standup where they would discuss the tasks that they worked on in the previous week.

Another integral part of the teams development process is the server that we would make use of for training the drone. Since getting the environment up and running was wrought with difficulties the team mainly utilized the server for both training and development of the neural networks. This was relatively problematic however since these tasks could not be done in parallel. Likewise the server couldn't really have multiple people working on it at the same time resulting in even more issues. The team would like to note that we feel this problem is largely derived from the environment we find ourselves in with respect to the COVID-19 pandemic. Since most of the team members do not have computers that can run the environment working on the project was very largely constrained by this issue.

The creation of Docker images to run the project required very specific hardware with an Nvidia GPU, large amounts of hard drive space, specific drivers, and the latest versions of Windows. We therefore used a desktop computer in an on-campus lab that team members connected to via AnyDesk. The team utilized the ue4-docker tool to create a ubuntu-based docker image with Unreal Engine and AirSim baked in.

3.7 Design Plan

The plan for this semester with regard to steps in the design process would look like the following...

1. Get the Environment running without bugs on the server
2. Design the heuristics and the neural networks that the drone will utilize
3. Train the networks
    a. Develop methods for speeding up the training process
    b. Track information regarding how fast the drone is learning and the behavior the drone is taking

Note for bullet 3, the two tasks are done in parallel, as well both these tasks being done in parallel with step 2.

4 Testing

4.1 Unit Testing

The unit testing done for our project was mainly done to make sure different components of the project are hooked up properly. A great example of this is we had a unit test that made checking if the connection between Airsim and Gym is working properly. This was achieved by making the drone move forward by one unit using Gym and then checking with Airsim if that is indeed what happened.

4.2 Interface Testing

Since there really is no user interface for this project there is no user interface testing that was conducted.

4.3 Acceptance Testing

Acceptance testing was mainly handled with the naive approach of checking the results we are getting from the drone. This was handled by a mix of simply seeing if the drone seemed to be behaving properly as well as plotting out the paths it was tacking through the environment it is in to be sure that it was not doing unwanted behavior, or otherwise unlearning things it had learned.When a given input was passed such as moving forward we would check that the drone moved as it should. This approach would continue through the bulk of the project.

4.4 Results

In the virtual training environment, the drone is focused on obstacle avoidance. So far, the drone is having problems properly avoiding the wall obstacles, furthermore it also sometimes crashes into the floor. As part of this iterative development process, we are revising the reward policy for the reinforcement learning and looking to add a better network for determining the altitude of the drone to avoid hitting the ground.

5 Implementation

Neural Network

The neural networks for this project were implemented using a popular machine learning library for Python called PyTorch. On top of PyTorch our team utilized a neural network library called StableBaselines_3 (SB3). SB3 is a tool that seeks to try and standardize the code surrounding different policies throughout a project by acting as a wrapper for the networks themself. For example, as pointed out in the deliverables section we needed a double deep Q learning method that was identical between the GNN and the CNN so that it can be easily concluded that the difference in results between the two networks was a product of the CNN versus the GNN, and not the implementation of the double deep Q learning software. This is in essence the reason we are using SB3.

Training Environments

Training the models on the physical drone is both slow, and expensive. It is for this reason

that we utilized a virtual environment for training the drone in. This virtual environment was made up of an amalgamation of different tools. The virtual environments themself were made up of some simple worlds developed in Unreal Engine 4 (UE4). An example of one such world is attached can be seen in figure 1 to the below.

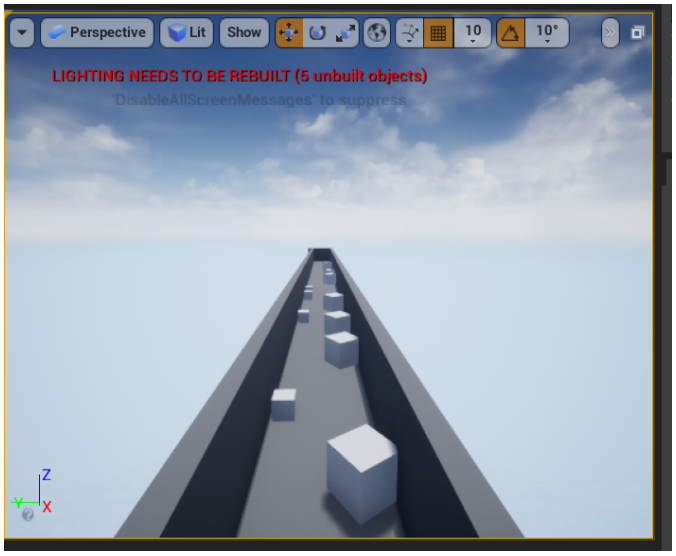


Figure 1: A photo of an environment used for training the Neural Networks.

Aside from UE4 our project makes use of the tool Microsoft Airsim. Airsim can be loosely described as a tool that creates an API that is used for interfacing with a drone that exists in the environment developed in UE4. The third tool involved here is known as OpenAI Gym, often referred to as Gym for short. Gym is a python library for standardizing and simplifying how an RL model interacts with an environment by simplifying the actions that can be taken by the model down to a few concrete methods, and standardizing the output of what the actor might see in any given situation. In figure 2 below you can see a rough diagram of how all the tools communicate and data flows throughout our project.

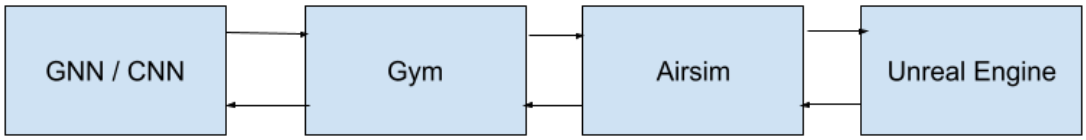


Figure 2: A rough diagram of how all the tools in our project communicate with each other.

Docker

The use of docker was an added part of the project which was assigned to us by Deepak about a third of the way through the second semester. This was to help mitigate any issues that we as a team went through in the first semester in regards to how to get the project running and which operating system to use. Now that the project is able to be run in a docker container these issues should be non existing as it is now compatible with any

modern machine. The main challenges of getting this goal completed were navigating somewhat deprecated tutorials from microsoft's airsim docker wiki along with just general steps that are not detailed for the beginner level in regards to docker. We found that that image needed to be run in Ubuntu for windows utilizing the WSL framework which solved most of these issues. Additionally, the somewhat specific instance of running AirSim within Docker required the use of Cuda and Nvidia drivers that were specific to Nvidia GPUs on newer versions of Windows 10. This approach is somewhat limited by these factors and the need to use Docker within the WSL command line environment rather than the more traditional Docker desktop environment. It appears that Nvidia and Microsoft are working on these issues, so future versions of docker and Cuda may support more traditional ways of running AirSim within Docker.

6 Closing Material

6.1 Conclusion

This project has taken a significant amount of man hours on behalf of the members of all the team. The team thinks we have done a great job on this project and are at large satisfied with the level of knowledge we all gained throughout this semester. At the onset of the semester only two team members had any experience with deep learning, and none of the team members had any experience with reinforcement learning. This made the project at large a learning experience, and after enduring the sisyphean task of learning about deep learning many of the members look forward to what these technologies can bring for the future. Sadly we regret to inform the reader that at the request of our advisor, at the time of writing, neither the GNN or the CNN are available to the public, we hope the reader understands. If the reader is wondering more about the specifics of the tools used in the project, almost all of them are open source and available to the public. If the reader is interested more in our project itself, our advisor hopes that the paper will be done in the near future, as at the time of handoff for our project there is not much that needs to be added before our advisor has all the material they need for their paper. For instructions on how to get the environment we utilized set up, the reader can refer to appendix I.

6.2 References

[1]S. Bhagat and P. Sujit, "UAV Target Tracking in Urban Environments Using Deep Reinforcement Learning", *arXiv preprint*, 2020. Available: 2007.10934 [Accessed 4 October 2020].

[2]K. Ko, "Visual Object Tracking for UAVs Using Deep Reinforcement Learning", 2020. [Accessed 4 October 2020].

[3]K. Lee, B. Vlahov, J. Gibson, J. Rehg and E. Theodorou, "Approximate Inverse Reinforcement Learning From Vision-Based Imitation Learning", *arXiv preprint*, 2020. Available: 2004.08051 [Accessed 4 October 2020].

[4]W. Luo, P. Sun, F. Zhong, W. Liu, T. Zhang and Y. Wang, "End-to-End Active Object Tracking and Its Real-World Deployment via Reinforcement Learning", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 6, pp. 1317-1332, 2020. Available: 10.1109/tpami.2019.2899570.

[5]J. Michels, A. Saxena and A. Ng, "High Speed Obstacle Avoidance using Monocular Vision and Reinforcement Learning", *Proceedings of the 22nd International Conference on Machine Learning*, pp. 593-600, 2005. [Accessed 4 October 2020].

[6]L. Xie, S. Wang, A. Markham and N. Trigoni, "Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning", *arXiv preprint*, 2017. Available: 1706.09829 [Accessed 4 October 2020].

[7] Brown, Brandon, and Alexander Zai. *Deep Reinforcement Learning in Action*. Manning Publications Co., 2020.

[8] Géron Aurélien. *Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly, 2019.

7 Appendix

7.1 Operation Manual

Operating the project can be done either locally or within an Unreal Docker container. Installing Docker will require installation as we have detailed. Alternatively, you can setup and run the system without Docker for a more visual approach.

7.1.1 Installing Docker

Using Docker with AirSim and Unreal engine will require a minimum of 4 GB of RAM although 8 is preferred. Another requirement is the ability to enable BIOS-level hardware virtualization support in the BIOS settings. If on windows, ensure WSL 2 is enabled in optional features. The docker image is built on Ubuntu and windows will need Ubuntu for windows installed along with the WSL 2 framework. Download and install Docker for windows as well as NVIDIA CUDA Drivers. These drivers will require a corresponding GPU and version of Windows as documented in the wiki. Create or pull the docker container from https://hub.docker.com/r/zmass/airsim and run the commands regarding the NVIDIA container toolkit found in the Docker Unreal AirSim WiKi page. Then pull the project from git into the docker container and run it from there. For the step by step process of setting up docker for Unreal AirSim follow the wiki page: https://git.linux.iastate.edu/dgthomas/graph_neural_reinforcement_learning/wikis/Docker-Unreal-Air-Sim

The above wiki page contains the basic steps along with the commands you will need to use to set up docker for Unreal Airsim. The project can then be operated from the command line within the Docker which has Unreal / AirSim running. You will be unable to see the virtual drone move, as in Figure 3 below, but you will be able to run multiple containers to learn simultaneously.

7.1.2 Setup and Running without Docker

Similarly to the Dockerized version of the project, if a user wanted to have everything on their local machine they could pull the project from git. The wiki details the steps necessary to get everything installed locally but there are some outdated steps, which will be omitted/revised here. The first thing needed is a computer with a GPU, preferably an NVIDIA GPU. The one used virtually utilized a Titan-GPU. You will need to get registered on www.epicgames.com to get the required privileges of using unreal in this project. The next steps will be to clone unreal and airsim using the commands found in the wiki. Lastly Python 3 and visual studio 2019 will need to be installed.

7.2 Older versions of the design

At the start of this project, in semester 1, the project was focused on both obstacle avoidance as well as object tracking. The current version of the design only focuses on object avoidance, with object tracking as a potential future aspect.

There additionally was a previous physical drone aspect to the project with the hope that training the drone virtually would allow for future use of the model on the physical drone. The physical drone is currently outside of the scope of our senior design project due to the limits of time and the ability to test physically due to COVID but may be picked up at a later stage by future teams. For now, we focused only on virtual training as shown below in Figure 3.

Figure 3: An image of the drone training within the virtual environment