Autonomous Drone Control using Reinforcement Learning

Progress Report
Kian Parnis

ABSTRACT

In recent years research using Machine Learning (ML) techniques for drone control has attracted much attention. These techniques generally either utilize Supervised Learning (SL) or Reinforcement Learning (RL). Reinforcement Learning tends to be the favored ML technique due its use of its agents' environment as a means of data collection, whereas SL requires a large dataset to be built which can be quite costly and time-consuming. As the area of drone control is quite broad, this study aims to focus solely on Obstacle Avoidance due to its importance in achieving tasks as well as recent widespread attention. Different RL techniques shall be tested using Discrete Actions followed by Continuous Actions to achieve better natural movement, which will utilise depth imagery. Different environment levels shall be made and split into seen and unseen environments to validate whether the agent is learning to achieve generalization. To achieve this AirSim, a high-fidelity visual and physical simulation shall be utilized in Unreal Engine to model the environments and conduct training and testing.

1 INTRODUCTION

1.1 Problem Definition

Currently, unmanned aerial vehicles (UAV's) depend on humans to manually pilot them using remote control. This may prove problematic when dealing with scenarios where a human pilot is unable to be within transmission range of the vehicle, such as in a search and rescue operation. Other examples include the need for drone navigation to be automated on a large scale such as having drones deliver healthcare related items on mass [1]. Thus, a lot of heavy research [2, 3, 4, 5] is being conducted in giving UAV's a level of autonomy to complete various crucial tasks.

The problem therefore, is managing to develop solutions which can carry out such tasks autonomously. However, UAV's can be limited to several constraints that may alter which technique would suit the problem best. One of these constraints is the drone's need to rely on its onboard cameras and sensors to capture knowledge about its environment. Apart from this, prior information regarding the environment might not be known. For example, for search and rescue operations, avalanches, earthquakes and other disasters may change how the environment is structured.

1.2 Motivation

The need for autonomous drone control is motivated due to the sheer number of different applications that can utilize such technology. In [6] the use of Reinforcement Learning with drone control is highlighted in different problems such as path planning, navigation and control. The following is a breakdown of some of the different applications highlighted by [6] which are Altitude control, Obstacle avoidance, Drone delivery, Unmanned Rescue etc. Respective research in these fields can be found in [2, 3, 4, 5]. This project starts by focusing on one of the most noticeable tasks UAV's need to overcome which is Obstacle Avoidance. The motivation behind focusing upon Obstacle avoidance is due to its overall importance in its involvement in various critical tasks. When carrying a search and rescue operation for example. The autonomous drones knowledge for navigating an adaptive environment is paramount to the overall success of the operation.

1.3 Challenges

This solution depends on the agent being able to navigate through static or dynamic environments that the agents had no prior knowledge of, depending on the limited data such as its depth sensor, as-well as its GPS signals to reach its targeted goal. The most prominent path-finding algorithm is the A* Search Algorithm. This algorithm guarantees optimality if its heuristic $h(n)$ is both Admissible and Consistent. A heuristic is said to be Admissible if it never over estimates the cost to reach the goal: $h(n) \leq cost(n, g_n)$ where g_n is the cheapest to reach goal node from n. It is Consistent if the estimated cost of a node n, does not exceed the cost to reach any successor, $n^{'} \in successors(n)$, and the estimated cost of the successor to reach the goal: $h(n) \leq cost(n, n^{'}) + h(n^{'})$.

A* typically relies on graph based approaches, it works by expanding nodes and backtracking if a better path exists. However, as one of the constraints the drone may encounter may be traversing with little to no knowledge of its surrounding environment, A* would require the drone itself to navigate the environment to gain knowledge about its surroundings. This can be problematic since the drone would be required to fly large distances and then be required to backtrack which can be detrimental in hazardous environments. Apart from this, if the environment shifts (obstacles are moving or wind interference is met) the algorithm would be required to recalculate its trajectory from scratch.

The Rapidly exploring Random Tree* algorithm (RRT*) is another widely used path planning algorithm. This algorithm succeeds in exploring a high-dimensional space by creating and expanding a space filling tree [7].

This algorithm manages to find non-optimal paths in static environments [8] but is unable to deal with environments that are dynamic as well as needing to reconstruct the tree if the agent moves through wind interference. Reinforcement learning aims to produce policies which can generalize well for both unseen and dynamic situations which is required to overcome such challenges.

2 AIMS AND OBJECTIVES

The aim of this project is to train an agent which is capable of navigating through various types of obstacles to a particular goal state using state-of-the-art Reinforcement Learning techniques, which will utilize a depth perception camera to collect data pertaining to its surrounding environment. This will be accomplished by fulfilling the following objectives:

1. **Objective (O1):** Design various environments for the agent to navigate through. These will be split into seen and unseen environments in which the drone will be trained on the former and then tested for generalization on the latter.

2. **Objective (O2):** Implement various state-of-art Reinforcement Learning algorithms and evaluate based on time taken for convergence, number of collisions and accumulated rewards.

3. **Objective (O3):** Adjust the drones kinematics from discrete actions to utilise continuous actions. These will then be compared to check whether the agent manages to reach its designated goal state for each created environment and the effect achieved with a smoother flight control while navigating through the environment.

3 BACKGROUND

3.1 Kinematics

In Figure 3.1.1 the different Kinematics the drone posses is illustrated. Quad-copters typically operate in two frames, the Inertial Frame and the Body Frame. The inertial frame is defined by the ground and is static pointing with the z direction towards gravity. The Body Frame is defined with the z, x and y axis changing with the orientation of the drone. The drone also possesses three different degrees of rotation. Its Roll angle is along its z axis with a Clockwise rotation defining a positive roll angle. Hence, rolling with a positive angle is equivalent to translating in the right direction. The Pitch angle is along its x axis with a Clockwise rotation defining a positive Pitch angle. Hence, rolling with a positive angle is equivalent to translating in the front direction. Finally, the Yaw angle is along its y axis with a Clockwise rotation defining a positive Yaw angle. Hence, rolling with a positive angle is equivalent to translating in the left direction. All these rotations are done with respect to our body frame which follow a Front Left Up (FLU) convention. The drones throttle controls the drones vertical movement along the z axis, typically a positive throttle will make a drone fly higher, while a negative throttle will make the drone fly lower.

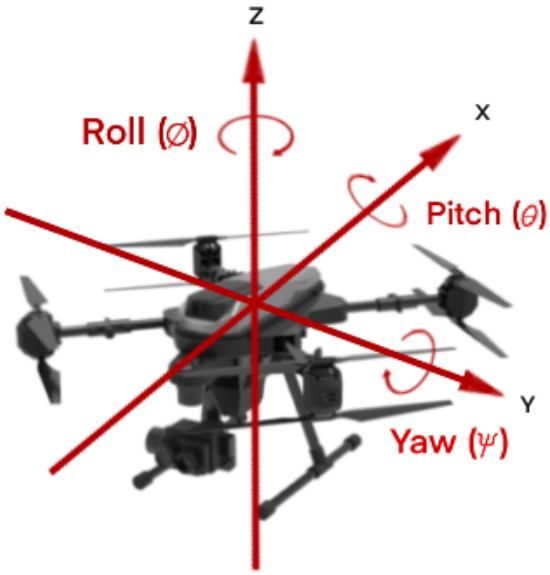

Figure 3.1.1: Drone's Kinematics

3.2 Reinforcement Learning

Reinforcement Learning (RL) is the process of learning what to do, by interacting with the environment. This involves the recognition of similarities and certain characteristics of certain situations and mapping these situations to actions, with the overall goal being to maximize some utility that is of interest. RL is typically broken down into an interaction loop (see Figure 3.2.1), this involves our agent taking some action in an environment based on some policy which transitions its current state to a new state while collecting some reward as well as an observation.

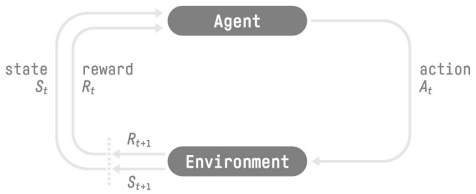

Figure 3.2.1: RL Interaction Loop [9]

The policy typically denoted as π, is a function that maps a state s_t to an action A_t (i.e., how the agent acts in each different environmental state by taking a decision to move into another state s_{t+1}, where t is the current time step). Rewards, are represented as numerical values given back to the agent as a result of the action taken in a state as R_{t+1}. Rewards R_t are used to improve the agent's policy

in deciding what action to take in a particular state to reach its target goal, the optimal policy π^* and is achieved by maximizing an expected cumulative reward (The sum of all rewards achieved after a particular time step) [10]. The environment is a scenario which is typically modelled around aiding the agent train efficiently to reach its desired objectives/goal. For example, the AlphaZero algorithm had managed to achieve superhuman level of play in Chess and Shogi within 24hours and this was achieved with the use of Deep Convolutional Neural Networks and playing self-play games utilizing RL [11]. In this scenario the environment can either be the real world or a simulation of it, with the latter being a cheaper, faster and safer way to conduct training.

Deep Reinforcement Learning (Deep RL) is a sub-field of machine learning that combines Reinforcement Learning (RL) and Deep Learning. Deep RL is often used when the state-space is not feasible to engineer due to space constraints. For example, the game of chess has 10^{47} states while continuous actions are considered to have an infinite state space. Deep Learning aids by utilizing an approximation function which is parameterized by a vector of weights and is typically characterized as Artificial Neural Networks (which includes CNN's and other variants of Deep Neural Networks).

Artificial Neural Networks (ANN) is a collection of perceptions and activation functions. The Network is comprised of an input layer, output layer and several hidden layers (see Figure 3.2.2) which map the input to its output. [12].

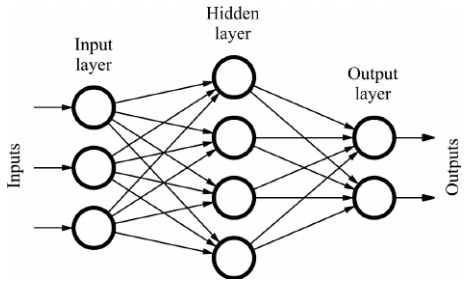

Figure 3.2.2: Artificial Neural Network

A Perceptron is a fundamental unit in ANN's, they receive a number of of real values with each input given a weight, with the weight determining the contribution of the input to the output. Its computations are the linear combination of these inputs plus a certain bias to produce a single output Y.

$$Y = \sum(weight * input) + bias$$

Since Perceptron's are linear [12], activation functions are also utilized to make neural networks nonlinear techniques which determines when a perceptron should fire. A number of these step functions exist with the simplest being the Step Function. The Step function [13] is a binary classifier that fires a one if the summation Y is greater than a certain threshold and a zero otherwise, which makes this activation function piece wise linear. The Sigmoid Function [13] is a smoothened version of the step function that's used to represent non-linear functions between 0 and 1. The Sigmoid function is denoted as:

$$\sigma(Y) = \frac{1}{1+e^{-Y}}$$

The Tanh Function [13] is a scaled version of the Sigmoid Function, this function maps the values between -1 and 1 which makes its gradients more stable. The Tanh function is denoted as:

$$\tanh(Y) = \frac{2}{1+e^{-2Y}} - 1$$

The problem with the Sigmoid and Tanh functions is that they fire all the time making an ANN heavy. The Rectified Linear Unit (ReLu) [14] is a less computationally expensive activation function that maps the input Y to the max of either zero or Y. This lets big numbers pass making few perceptrons stale. The ReLu is denoted as:

$$\sigma(Y) = max(0,Y)$$

Within RL, an ANN's input layer corresponds to states (for example an input image), while the outputs represent our actions (Move up, Move down etc.).

3.3 Convolutional Neural Networks

Most drones come with cameras attached to them, this is normally used to either record feed which can be accessed later on or to transmit live feed to the controller the pilot is using. Drone imagery can be used as one of the observations that can be fed to the Reinforcement Learning Algorithm to receive information about its surrounding environment. However, If a regular feed forward neural network were to be used for this imagery, its large size would result in a huge number of neurons being needed. For example, an image with size 300x300 px and 3 color channels would result in the network having 270,000 weights (300x300x3), which is not only computationally expensive, but leaves the network prone to over-fitting (learning to memorize its data-set) due to the large number of parameters.

Convolutional Neural Networks (CNN) are based on the idea of Artificial Neural Networks but aim to use the idea of mathematical convolution to perform well with image, speech, or audio signal inputs. CNNs gained popularity based of the ImageNet architecture during 2012 [15]. The following is the breakdown of the architecture used:

1. Input layer which holds image pixel values.
2. Hidden layers composed of two sets of Convolutional layers with ReLU [14] (to achieve non-linearity) followed by Pooling layers.
3. Classification which flattens the result and passes through two fully connected layer's which can be found standard in a normal ANN.

Convolutions layers utilize Kernels (Also referred to as Filters and Feature Detectors) which are small matrices known as Edge Detectors that stride across the image starting from the top and progressively going downward per stride. For each stride a dot product between the Kernel and the current position on the image is taken and saved into a Feature Map. This Feature Map would then reduce the image (depending on the kernel size) with specific patterns of information in an image being highlighted.

Pooling Layers are used to down-sample the feature map along its spatial dimensionality [16] to further reduce parameter size while maintaining important features which helps with the previously described over-fitting problem.

In [17], Reinforcement Learning, Deep Learning and Convolutional Neural Networks where successfully integrated together which used sensor data and game scores to learn successful control policies. The modified RL algorithm Deep Q-Learning agent[18] managed to beat several Atari Games such as Space Invaders, Pong and Breakout while achieving a level comparable to that of a professional human players.

4 LITERATURE REVIEW

In this section, we will review the state-of-the-art techniques used in Autonomous Drone Navigation.

In [3] a study was carried out for drones with obstacle avoidance capabilities. The study aimed to compare different Reinforcement Learning techniques and split there uses for both discrete actions and continuous actions. For discrete actions a comparison was done between DQN [18] and its variants, Double DQN [19] , Dueling DQN [20], and Double Dueling DQN (D3QN). Along with these different algorithms, the actual input was also tested for the difference between RGB and Depth imagery. Three main environments were used to test these algorithms. A woodland that had a random number of trees being distributed in a path, a block word that had 3D Objects such as cubes, cones and spheres distributed and finally, the third environment built upon block world to introduce a curved trajectory which utilizes yaw control. The action space utilized by these algorithms where split into five: up, down, forward, left and right. The result of this first experiment showed that Double Dueling DQN (D3QN) gave the best performance through each environment. The reasoning why this is the case is since D3QN combines both Double DQN and Dueling DQN which are both improvements on the vanilla DQN. It was also shown that combing both RGB and Depth imagery yielded slightly better results than depth imagery, while depth imagery yielded significantly better results to RGB. It is also noted that for the final environment, the UAV's utilizing these techniques did not manage to complete their races. These results can be shown in 4.1.

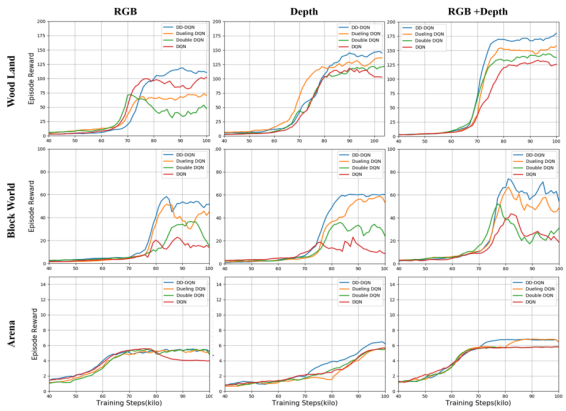

Figure 4.1: Results portrayed for Discrete actions [3]

The second experiment utilized continuous actions to improve upon the unnatural movement caused by discrete actions. The algorithms used were from the family of policy gradient algorithms which consisted of Trust-Region Policy Optimization (TRPO) [21], Proximal Policy Optimization (PPO) [21], and Actor-Critic using the Kronecker-Factored Trust Region (ACKTR) [21] that all utilize actor-critic networks. Within this section of the study, it was highlighted that a U-Net-Segmentation model [22] is used. This type of model used to be done through supervised learning, but the labeling of all this data was not only time-consuming but was not perceived as optimum. Thus, the structure of this experiment utilized the policy gradient methods to solve the manual labelling of data. The outcome of this would be a vision generated segmentation map as well as Actor-Critic RL for drone control. The result of this second experiment highlighted two important findings, firstly for the third environment, unlike discrete actions the smoother flight achieved by continuous actions would lead to the completion of the actual course. The second result showed that the ACKTR algorithm performed best, achieving the smoothest and most stable trajectories.

Within this study an experiment was also made to compare each algorithm with human pilots of various experience levels. This was achieved through a race. The most skilled of these pilots managed to be the fastest in the first environment. In the second environment, the expert pilot and the ACKTR model achieved similar results, while in the final and most challenging environment ACKTR came in first place while the human pilot came in second. These results are shown in Figure 4.2.

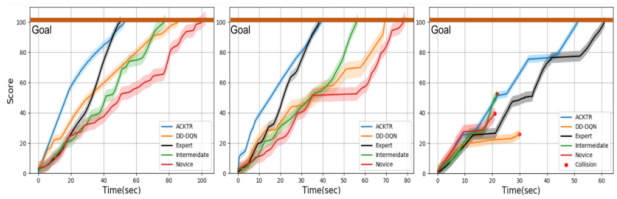

Figure 4.2: Results of algorithms compared with human pilots [3]

Further research [23] proposed a UAV trained via RL that

can avoid both moving and static objects within a 3D urban environment. This study chose to focus solely on continuous actions using the Deep Deterministic Policy Gradient algorithm (DDPG) [24]. A customized reward function was proposed that gives penalties if the drone collides with environment and rewards based on the minimization of the distance between the UAV and its destination. The result of this is an efficient framework that allows the drone to navigate its urban environment and effectively reach its target goal.

5 PROPOSED METHODOLOGY

5.1 Air-Sim

The modelling of different environments for the drone to navigate through shall be done using Unreal Engine to create a simulation environment grounded in realism. For the integration of the drone, Microsoft's AirSim [25] offers a multi rotor model with realistic physics that can either be controlled by using an RC controller or programmatically. AirSim also offers methods for different levels of control. At the lowest level the drones Yaw, Roll and Pitch can be controlled. At a higher level, the drone's movements can be controlled to move along the x, y and z axis.

5.2 Implementations (O2, O3)

For the implementations of O2 and O3, the AirSim API is used to interact with the drone simulation programmatically using Python. OpenAI Gym [26] is used to wrap all necessary functions such as (step, reset, act) through an interface which allows each state-of-the-art Reinforcement Learning Algorithm to be easily interchangeable. Stable Baselines3 (SB-3) [27] is a set of reliable RL algorithm implementations in PyTorch. SB-3 makes it simpler to train and compare the different RL algorithms while easily interfacing with the created gym wrapper.

As discussed during the previous sections, one of the observations the drone shall receive is depth imagery, thus, a CNN Policy will be utilized for each algorithm. The following are the current proposed rewards the drone shall receive:

- Negative reward given when drone collides with a wall or obstacle.
- Positive reward given when drone successfully reaches target location.
- Positive intermediary reward given if the distance between the drone and the goal point decreases.
- Negative intermediary reward given if the distance between the drone and the goal point increases.

5.3 Simulation of Different Scenarios (O1)

As discussed previously, training will be conducted through simulations. Unreal Engine is used to create the different scenarios that the drone would need to navigate through. Each of these scenarios will consist of a starting point, where the drone will be activated (refer to Figure 5.1.1) and a goal point the drone would need to navigate towards.

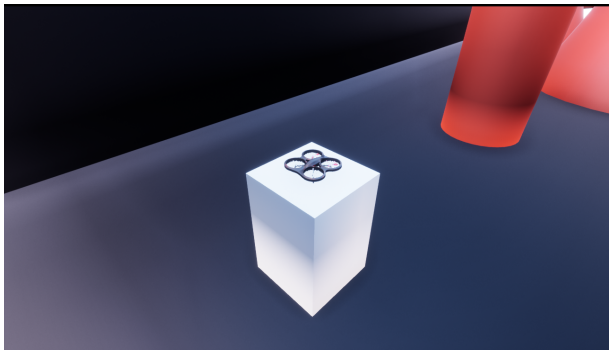

Figure 5.3.1: Drone Staring Point

Each environment will contain various obstacles such as trees, walls with holes that the drone can pass through and columns to avoid. These environments will be instantiated either randomly or in a specific order, depending on the results obtained. The unseen environments will not be shown during training and will be used for evaluation by the trained model.

5.4 Discrete Actions (O2)

A higher level of control will be used with Discrete Actions, these actions will allow the drone to move around the x, y and z axis freely with the combination of seven different actions, six actions to move positively or negatively in any direction and an action to stay as the same coordinate. The following algorithms will be compared for discrete actions:

- DQN, Double-DQN, Dueling-DQN, D3QN

5.5 Continuous Actions (O3)

When alternating to continuous actions, the custom gym wrapper will be modified to handle these sort of actions. Thus, there are currently four proposed continuous actions, the drones Yaw angle, Pitch angle and Roll angle given as radians. Along with the drones rotation, the final action is the drones desired throttle between 0.0 and 1.0. The following algorithms will be compared for continuous actions:

- TRPO, DDPG, PPO, ACKTR

6 EVALUATION PLAN

The main tool used for comparing the results of each model will be done via Tensor-Board. Tensor-Board offers visualizations and tracking metrics such as loss and accuracy. This tool will indicate during and after training, whether each algorithm is managing to learn to navigate through their surrounding environments. A number of metrics are offered by Tensor-Board which vary from algorithm

to algorithm depending on their hyper-parameters. The main metric which will be used to contrast each algorithm depicts the number of steps taken with respect to the reward received (Reward Mean). Typically an RL algorithm is shown to be learning if the sum of reward received increases with respect to the amount of steps taken until the model converges towards a plateau.

The higher the sum of reward is, the more optimized the drone's route to the goal would have been. This metric will also be used to measure the amount of time taken for each algorithm to successfully converge and learn the target policy, which may show a trade-off between how well a model performs versus how long the model took to train. After the drone is fully trained, a second metric will be used to see what the collision rate is on both seen and unseen environments, to measure if the drone is learning to generalize and will be calculated as follows:

$$CollisionRate = \frac{TotalCollisions}{TotalAttempts} * 100$$

Where 'Total Collisions' is the amount of times the drone has been unsuccessful and collided with an object and 'Total Attempts' is the number of times the drone has navigated through the environment. If a low collision rate is met on just the seen environments, then an indication would appear that the drone is over-fitting and learning its training data. However, if a low collision rate is met on both seen and unseen environments, then it can be shown that the drone is learning to generalise.

7 CONCLUSION

Through this report, it has been highlighted that there is a need to develop solutions which can carry out drone control tasks autonomously. The anticipated result of this project is to focus on one of these control tasks; obstacle avoidance and with objectives highlighted in section 2, train models which can (1) learn to reach target goals through generalisation and (2) ultimately use continuous actions to take more precise steps to reach the target goal. The implementation and evaluation carried out will follow what has been mentioned in sections 5 and 6 unless any issues arise.

References

[1] J. E. Scott and C. H. Scott, "Drone Delivery Models for Healthcare," in *Hawaii International Conference on System Sciences*, 2017.

[2] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement Learning for UAV Attitude Control," *ACM Trans. Cyber-Phys. Syst.*, vol. 3, 2 2019.

[3] S.-Y. Shin, Y.-W. Kang, and Y.-G. Kim, "Obstacle Avoidance Drone by Deep Reinforcement Learning and Its Racing with Human Pilot," *Applied Sciences*, vol. 9, no. 24, 2019.

[4] G. Muñoz, C. Barrado, E. Çetin, and E. Salami, "Deep in Reinforcement Learning for Drone Delivery," *Drones*, vol. 3, no. 3, 2019.

[5] W. Cao, X. Huang, and F. Shu, "Unmanned Rescue Vehicle Navigation with Fused DQN Algorithm," in *Proceedings of the 2019 International Conference on Robotics, Intelligent Control and Artificial Intelligence*, RICAI 2019, (New York, NY, USA), pp. 556–561, Association for Computing Machinery, 2019.

[6] A. T. Azar, A. Koubaa, N. Ali Mohamed, H. A. Ibrahim, Z. F. Ibrahim, M. Kazim, A. Ammar, B. Benjdira, A. M. Khamis, I. A. Hameed, and others, "Drone deep reinforcement learning: A review," *Electronics*, vol. 10, no. 9, p. 999, 2021.

[7] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," *CoRR*, vol. abs/1105.1186, 2011.

[8] C. Zammit and E.-J. Van Kampen, "Comparison between A* and RRT Algorithms for UAV Path Planning," Proceedings of the 2018 AIAA Guidance, Navigation, and Control Conference, 11 2018.

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[10] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[11] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, and others, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[12] S. Haykin, "Neural Networks and Learning Machines," Pearson Education India, 2010.

[13] T. M. Mitchell and T. M. Mitchell, *Machine learning*, vol. 1. McGraw-hill New York, 1997.

[14] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[15] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Neural Information Processing Systems*, vol. 25, 11 2012.

[16] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, and others, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[19] H. v. Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pp. 2094–2100, AAAI Press, 2016.

[20] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling Network Architectures for Deep Reinforcement Learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 1995–2003, JMLR.org, 2016.

[21] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, 2015.

[22] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, 2015.

[23] O. Bouhamed, H. Ghazzai, H. Besbes, and Y. Massoud, "Autonomous UAV Navigation: A DDPG-Based Deep Reinforcement Learning Approach," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, IEEE, 10 2020.

[24] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2016.

[25] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*, pp. 621–635, 2018.

[26] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *CoRR*, vol. abs/1606.01540, 2016.

[27] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

Thesis	**0h**	**47%**
Thesis	**0h**	**47%**
Research / Learning the Tools	0	100%
Setting up Artifact (Setup UnReal, Ai...	0	100%
Writing Progress Report	0	100%
Submission of a Progress Report	0	100%
First Semester Assignments / Exams	0	0%
Continue work of Artifact (Achieve O...	0	0%
Thesis paper	0	0%
Submission of abstract for ICT Exhibit...	0	0%
Submission of first draft of dissertati...	0	0%
Refining paper based on feedback	0	0%
Submission of FYP dissertation.	0	0%