OROGRAPHIC SOARING FOR SUAV FLIGHT IN URBAN ENVIRONMENTS USING REINFORCEMENT LEARNING

Prithvi Abhishetty

Department of Aerospace Engineering, University of Bristol, Queen's Building, University Walk, Bristol, BS8 1TR, UK

ABSTRACT

The aim of this research was to develop a reinforcement learning algorithm that could predict the path of minimum flight costs in an unseen urban environment. The path planner was optimised by considering both the shortest path and the most energy efficient route that was mapped by a wind field limited to orographic updrafts. A number of 2D grids were generated with randomly located updrafts and an agent, representing a SUAV, was trained to find the optimal path over a number of time-steps. The results showed that while the agent was able to find the goal using the minimum cost path in static environments, for changing wind-field environments the developed policy could not achieve this goal. Consequently, the investigation suggested that a Reinforcement Learning has a lower performance than the conventional, A path planner for these 2D environments. Future work, however, could seek to build upon this research by developing a Reinforcement Learning algorithm from scratch in order to overcome the limitations of existing optimisation policies outlined in this report. Additionally, a more realistic representation of urban areas could be obtained by considering an altitude factor in the environment.*

Keywords: Reinforcement-Learning, Environment, PPO, Path-Planning, Orographic-Lift

1 INTRODUCTION

Small Unmanned Aerial Vehicles (SUAVs) have developed a wealth of interest over the past decade, as they have the unique ability to operate in urban environments. With their high manoeuvrability, SUAVs are relied upon to complete a plethora of missions at lower altitude than that which is feasible for other aircraft. This includes surveillance [1], medical supply [2], and infrastructure inspection [3]. With the latter application, Mota *et al.* explore how the key characteristic of these vehicles, being unmanned, enables their use for the inspection of transmission lines, which would otherwise present safety concerns if carried out by humans. The autonomous performance of SUAVs is therefore an important area of research with the goal of improving aspects such as flight stability and efficiency in transit. While this can also be achieved by advancing the internal guidance, navigation and control system, the scope of this research paper is limited to the energy efficiency of travel. Having outlined the many advantages of SUAVs, the necessity of this investigation becomes evident when exploring their limitations due to energy storage. Batteries must be restricted due to the size and weight bounds for these small aircraft, yet even with such constraints they can occupy up to 40% of the mass which has direct implications on the flight range and endurance [4]. Consequently, for missions dependent on a long flight time such as search and rescue, or for time-critical tasks such as delivery of goods, the quality of performance is reduced.

With the need to minimise battery usage for a required flight, path planning techniques and collision avoidance methods become a critical fields of research to improve SUAV performance in urban environments. A path planning algorithm computes a trajectory from the UAV's present location to a desired future location [5]. The path which it generates is dependent on the design of the algorithm, therefore a good planner possesses several attributes which

are all relevant to the desired optimisation. A simple and common requirement is to compute the shortest path; the most direct route from a start point to a destination. In the context of efficient travel this may suggest least energy expended, however, as explored by Williamson [6], the shortest path does not consider the complexity of wind fields in an urban landscape. Therefore, wind-aware path planning algorithms must also be used to identify the energy costs and gains through unfavourable and favourable wind conditions respectively. In Harvey's study [8], simulations using Williamson's wind-based path planner have drawn results suggesting that energy can be saved when flying at low altitude using soaring techniques. Lawrance [9] defines these techniques as the ability of birds to extract energy from the atmosphere and remain in flight without propulsive force. As they are of similar size to birds in urban areas and share a similar energy storage limitation, SUAVs can employ bio-mimicry to autonomously utilise soaring for optimised flight (Figure 1).

Figure 1: Autonomous Soaring UAV. [7]

Bøhn *et al.* [10] explore how birds have developed this behaviour through a bottom up approach, learning through an 'evolutionary trial-and-error process' rather than using theory or logical means. Following the approach of bio-inspiration, using Reinforcement Learning (RL) methods to design a path planner holds great potential for this investigation, as they take a similar bottom-up style of learning. RL [11] is a sub-field of Machine Learning (ML) concerned with teaching an agent how to behave in an environment by telling it how well it is performing.

This research targets the exploration and exploitation of wind fields with natural Reinforcement Learning techniques in order to improve wind-aware path planning algorithms for SUAVs. The reward factor will be minimising flight energy costs between a start and end point, and will be achieved using the OpenAI Proximal Policy Optimisation (PPO) method [12] and OpenAI Gym spaces [13]. Opting for OpenAI tools is beneficial as they are open-source and used by many researchers in the field. This allows for the standardisation and benchmarking of results when comparing findings with similar projects. The scope of this investigation is limited to orographic lift in the wind field as the related work reveals a research gap for this soaring strategy, particularly from a Reinforcement Learning approach. Moreover, it is theorised that RL is most applicable to orographic effects as it can be approximated to a 2D wind field unlike gradient and thermal soaring, which are more dependent on the (third) altitude dimension. As a consequence, the path planner would be able to learn over more iterations without computing power as a limitation. This work concerns the hypothesis that Reinforcement Learning techniques achieve greater energy saving in urban environments than conventional path-finding algorithms.

The rest of the paper is organized as follows. Previous works on Reinforcement Learning, soaring strategies and path planning are presented in Section 2. Section 3 explains the approach taken to develop the RL path planning algorithm as well as the test environments required to assess the performance of the algorithm. In Section 4, the results of the investigation are displayed and in Section 5 they are evaluated against the original hypothesis. Section 6 outlines the limitations and potential for future work and Section 7 offers a summary with some final remarks.

2 BACKGROUND

The Reinforcement Learning feedback loop illustrated in Figure 2 [11] describes the framework used to solve most problems in this field of study. Sutton *et al.* define the agent as the learner

and decision maker in RL; they interact with the environment by taking actions and receive a reward based on their actions. The environment can then be understood as everything that is outside of, and hence cannot be arbitrarily changed by, the agent [11]. When the agent takes an action, A_t, the environment transitions from its current state, S_t to a new state S_{t+1}. As a result, feedback is then received by the agent in the form of the new state and a corresponding reward, R_{t+1} which acts as an indication of how good the chosen action was (Sequence 1). When an RL problem satisfies a condition known as the Markov Property [11], it can be modelled using this framework and is said to be a Markov Decision Process (MDP).

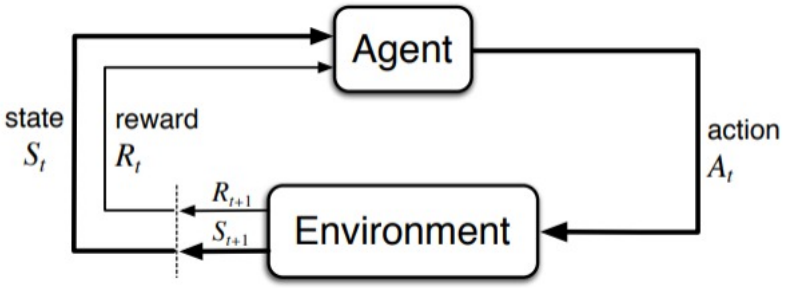


Figure 2: The Agent-Environment interaction in a Markov Decision Process. [11]

In a finite MDP, the set of states and the set of rewards are finite, and the process outlined above forms a sequence known as the trajectory.

$$\text{[11] Trajectory: } S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3... \tag{1}$$

This cycle may progress endlessly, for continuous problems, whereas trial-based tasks end with a terminal state where the finite sequence is defined as an episode. The goal of Reinforcement Learning is for the agent to maximise the cumulative reward across the trajectory. This is achieved by developing a policy -a strategy that maps current environment states to the probability of selecting each action. As an energy-saving path planner, this agent-environment interaction mimics the trial-and-error process of birds in order to minimise battery usage in flight.

To develop a strong understanding of soaring algorithms, the urban flight strategies of gulls can be studied. In doing so, Williamson [6] found that flight costs could be reduced by implementing static or dynamic soaring strategies. In static soaring, the pitch or bank angle of the air platform is held constant to receive energy gain in the updraft/lift region. In dynamic soaring on the other hand, the control angle must be adjusted to match the changing gradient of the local wind field in order to decrease energetic costs of flight. In upwind and downwind gradients, there is a variation in wind speed with altitude, therefore by taking routes in the direction of the upwind gradient and then changing to match the downwind path, gulls and by extension SUAV's can travel a distance from a start location with no-net energetic gain. This strategy could be difficult to implement in real-world missions as the unpredictable path generated when utilising wind gradients may increase time costs to reach a desired location.

When investigating the static strategies, Williamson also concluded that control costs could be reduced by orographic soaring, mentioned above. This is the ability of gulls to conserve energy by taking advantage of lift created close to buildings; a beneficial consequence of the complex flow generated between wind and tall urban structures. Shepard *et al.* [14] modelled the variability of this flow around buildings as can be observed in Figure 3. The red and blue regions indicate that the flow velocity vector is positive and negative on opposite sides of the buildings, which reveals the presence of updrafts and downdrafts in the wind field. Downdrafts are regions of airflow where additional energy is required to maintain altitude. The energy saving

at low altitude from wind-building interactions are coupled with control-based repercussions, as turbulent flows which are generated can highly perturb small air platforms. Thus, with the motivation that orographic soaring could be mimicked with a fixed-wing SUAV, Guerra-Langan *et al.* [15] investigated the strategy required to 'minimise the control effort required for a given level of orographic lift' and found that flying directly over the front corner of buildings could achieve this.

In contrast to orographic lift which harvests energy in forward travel, thermal soaring, is the upward displacement of air due to convection currents that allows small air platforms to maintain or increase altitude in flight without expending their energy storage. Williamson *et al.* [16] found that urban environments provide a significant level of thermal availability thus increasing the importance of applying this strategy to SUAVs in order to improve flight performance. The gulls used in Williamson's investigation show that in order to use the wind convection currents as a tool to gain altitude, air platforms must mimic a circular behaviour that results in a helical soar path as they move up columns of rising warm air. This effect has been explored with Reinforcement Learning, as Reddy *et al.* [17] set out to test the energy saving performance of a glider with a learned policy for thermal regions. Despite concluding that the 'real-world intricacies' of thermal soaring rendered the policy ineffective, the potential for RL with simpler soaring strategies is yet to be studied.

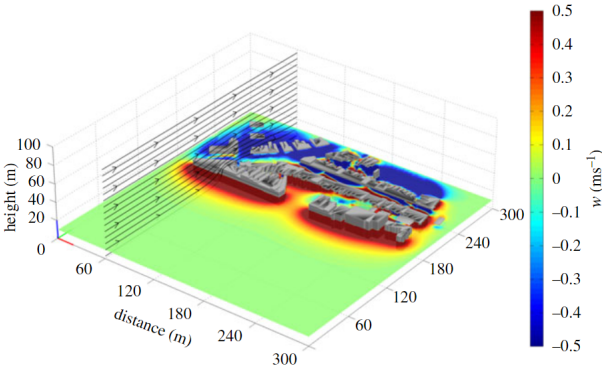


Figure 3: Orographic flow around buildings. [14]

The utility of orographic and thermal lift has also been displayed when analysing vulture behaviours. Williams *et al.* [18] showed that updrafts can be exploited for additional energy saving in path planning. This is due to an apparent social network created between the vultures in the urban areas that allow them to be informed on the location of updrafts far before encountering them. Taking thermal updrafts as an example, informed birds are capable of increasing their velocity when entering the 'inter-thermal glide' which would otherwise be a risky strategy if minimal information were present. The limitation of this strategy is the birds' ability to interpret social cues in this network, such as the fact that circling behaviour in other birds indicates the presence of this thermal energy. The birds can only receive information in a given range due to hearing and sight ability, which is especially restricted in urban areas. When applying this concept to SUAVs, however, the use of transmission signals that act as the medium for communication between the air platforms would allow for a greater range and speed of information transfer compared with birds.

Having discussed the various energy-saving regions in wind fields, analysis may be conducted to generate energy-saving paths for SUAVs. This is the objective of such path planning techniques mentioned earlier. When finding a route these algorithms can take one of two approaches: global optimisation or local optimisation. Williamson's research effectively demonstrates these; the global optimiser 'A* algorithm with frontier expansion' makes route choices having received full knowledge of the map environment, whereas the local optimiser Depth First Search (DFS) branches at each tree node [6] hence reacts to the local area. Global optimisation is beneficial for pre-planned missions as having full map knowledge beforehand would allow more effective path finding, whereas local optimisation may be necessary for on-board navigation where wind conditions are changing and the air platform is dynamically soaring.

Understanding these advantages and disadvantages, Lawrance [19] introduced an exploitation and exploration planning method that sought to make use of both optimisers. Exploitation is the process of decision-making based upon existing information whereas exploration is the act of seeking new information that could allow the overall goal to be met faster. Though exploration has the potential for more desirable information, there is the corresponding risk that the outcome is costly, as it is unknown, therefore a careful balance of the two methods is required. Lawrance's path planning algorithm manages the SUAV's energy by evaluating whether to locally exploit known information on the wind field for energy saving or to globally explore unknown areas in the environment, for map improvement.

3 METHODOLOGY

To support the methods described in this section, a link to the GitHub repository containing my code for the RL Path Planning Algorithm can be found in appendix A.

3.1 The Experimental Set-Up

The environment of the SUAV needed to be well designed in order to accurately assess the performance of the path planner for real-world urban areas. The term 'environment' is used here cautiously as it is not the same environment used for interaction in the RL algorithm (that will be defined later with the observation space). The term is being used here to describe the visual render of the wind-field area and this set-up enables the positions of the updrafts, obstacles and endpoints to be established. Furthermore, a range of map designs were generated, so the UAV could be evaluated in multiple city layouts and controlled experiments could be taken. The latter served to demonstrate the performance of the algorithm when individual reward parameters were varied.

The city maps were produced by modelling them as a two-dimensional grid where every cell represented a possible position in the environment. Discretising and approximating the urban area was necessary because the process of learning from experience in Reinforcement Learning required a high number of trials and choosing a continuous space or high dimensions would rapidly increase the computational cost. Multiple iterations were needed so that the agent could develop an optimal policy through 'reinforcing' actions that led to high rewards and discouraging actions that resulted in low rewards [11]. As mentioned before in section 1, orographic lift can be more accurately approximated in two dimensions than other soaring effects. This is supported by the work of Shepard *et al.* [14] who carried out an investigation on how gulls exploit airflows around 'fine-scale features' for energy saving. They found that gulls soar parallel to the face of buildings (Figure 3) and attempt to maintain their height with orographic lift rather than increase it. This allowed the wind field data to be simplified into two dimensions by averaging the vertical (altitude) component along the direction of flight. For thermal soaring on the other hand, the unique helical behaviour to harvest energy from the rising columns of air can only be defined with all [x,y,z] directions, and for gradient soaring the altitude cannot be averaged as it is needed to define the magnitude of ascent/descent.

3.2 Features of the Grid Environment

Figure 4 displays an example of the 2D grid with wind-field data. This map was designed on Python 3.10.2 and plotted using the matplotlib package (data visualisation and graph plotting library for Python). The grid is 50x50 cells giving a total of 2500 different positions in the environment. Therefore, though still discretised, this high number of cells allowed the wind field to be modelled to a greater degree of accuracy in order to better represent the physical world. Firstly, these grid positions were used to define the start and end positions where the green

circle is the start and the blue cross is the goal. In Figure 4, these were selected at cell [48,20] and [8,44] respectively. Secondly, the obstacles were represented by black circles that each occupy a cell position. Practically, the obstacles act as any buildings or infrastructure that the SUAV must navigate by modelling them as rectangles of different dimensions. The decision was made to manually design a limited number of maps with different obstacle layouts, because randomising the rectangle sizes and locations would prevent the portrayal of real urban landscapes. Additionally, the cell lines at 0 and 50 in the x and y directions were occupied by obstacles in order to define the boundary of the city which the path planning agent was restricted to.

Finally, the wind-field data for the orographic lift was presented in this grid with a Gaussian filter model. A Gaussian process (GP) is advantageous for wind mapping because wind fields are effectively boundless [19] therefore require a continuous distribution to make predictions at any location. Furthermore, the variance specified in a GP represents the confidence interval of the optimal path proposed by the learned planner [19]. In order to design this filter, a number of updrafts were plotted as sources with red and magenta circles to appear in random locations of the grid. The red sources represent updrafts, whereas the magenta present downdrafts. All sources are restricted to appear within the boundary and not on any obstacles since orographic lift would not occur inside any buildings. Considering them as point sources, the Gaussian filter served to model the spread of the lift to surrounding cells with a cumulative normal distribution. This can be interpreted as the intensity of the lift ranging from 1 to -1, one being at the positive source, minus one at the negative source and zero at the furthest distance from both. Running the filter over the grid required an approximation of the continuous distribution to the discrete cells therefore the intensity was calculated at each cell individually. This was achieved using the normcdf function [20] which takes three parameters: the point of evaluation, the mean and the standard deviation. The point of evaluation was taken relative to the source thus for each cell it is the minimum distance to the closest updraft. The mean was chosen as zero so it was a standard distribution and the standard deviation was chosen as five through an empirical method, observing if there was enough spread of the data relative to the size of grid.

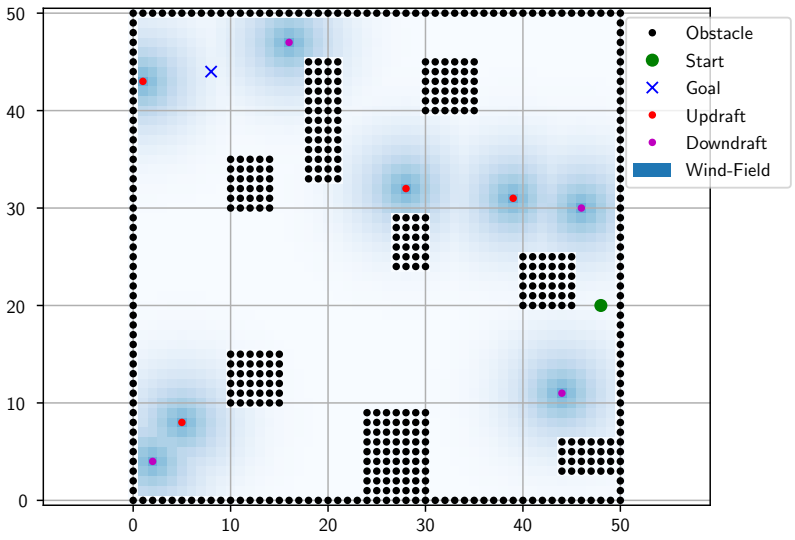


Figure 4: Example Wind-Field Grid Environment

3.3 Action Space, Observation Space and State

Having defined the initial features of the map, the environment class that the agent would interact with in the Reinforcement Learning paradigm could be created. The basic structure of

this was defined by the OpenAI gym observation and action spaces.

The state space describes the condition of the environment completely. The observation space describes what the agent is able to observe about the state at a given time-step so is dependent on how the agent and environment interact [11]. The agent and environment are defined under the finite MDP framework therefore their interaction is modelled by Figure 2. Within a finite MDP, the state is the same as the observation if the environment is completely observable and without noise [21]. For this Reinforcement Learning problem, the agent was able to completely observe the state that defines the status of the environment as this information was only the grid position, which was received by the agent at each time-step. Although there are other features of the environment such as the wind-field, obstacles and endpoints, these were not included in the state space as they do not change in a given episode and were only required to define the rewards (Section 3.6). There was also no noise in the system due to the policy definition discussed in Section 3.5. An additional point to note, is that OpenAI gym uses the term observation space instead of state because gym environments are modelled as Partially Observable MDPs (POMDPs); the agent partially observes the base gym 'env' class although it provides no information of the state of our environment [13]. Ultimately, in this grid world, the states and observations constituted only the position of the agent. Due to the fact that both 'x' and 'y' directions define the 2D grid position, the observation was specified with the OpenAI 'MultiDiscrete' space.

The action space was necessary to provide the set of actions that the agent can take in the environment. Given the discrete nature of the observation space and state, the actions remained discrete and compromised eight moves: North, North-East, East, South-East, South, South-West, West and North-West. These moves were required to appear as coordinate transforms in order to change the state (grid position), however, for simplicity, the actions were initially specified as one of eight integers with the OpenAI 'Discrete' space.

3.4 Design of the Reinforcement Learning Path Planner

The design of the RL path planner constituted transforming the environment into a form that was suitable for the MDP framework. In the initialisation and reset functions, the state was initialised to a specified starting coordinate in the grid (Section 3.2). An additional counter variable was initialised which was decremented every time-step in order to enforce a terminal state for the episode if the agent did not hit any of the obstacles or the goal (the other terminal states). It was set to 2500 as this was the total number of positions in the grid, however this was arbitrary as its purpose was to prevent the agent from traversing the environment continuously. Once initialised, the agent took a step in the environment. This involved taking an action and returning the updated state, associated reward and the counter variable which is decremented at every step. The method of choosing the actions and rewards will be discussed in Section 3.6. Once chosen, the value, either 0,1,2,3,4,5,6 or 7, was mapped to one of eight coordinate transforms. These functions alone were sufficient to compute the MDP trajectory, however after the terminal state was reached an additional render function was included to allow visualisation of the grid environment episodes. The useful data displayed includes the (randomly generated) wind field data and the path taken by the agent. Finally, the environment was reset so that further trials could be carried out.

3.5 Policy Algorithm and Neural Network

The policy was previously defined as the method of selecting an action in a given state. In a finite MDP, using a policy to compute the trajectory shown in sequence 1, the state-value

function [11], $v(s)$, under a policy π is found to be:

$$v_\pi(s) = \mathbb{E}[\sum_{k=0}^{n} R_{t+k+1}|S_t = s] \tag{2}$$

This function computes the expected reward by following a policy starting at the state, s, and achieving rewards given by the set of rewards, R, until the terminal state is reached at some finite value, n. An optimal policy is one where the expected return over all states is the greatest, and there may be multiple optimal policies for an MDP however they will all have the same (optimal) state-value functions. In order to develop an optimal policy to achieve the goal of minimum flight cost in the wind field, Proximal Policy Optimisation (PPO) was introduced. PPO is a current state-of-the-art policy-training RL algorithm, created by OpenAI and supplied by Stable-Baselines.

This algorithm was chosen for the path planner due to a number of reasons. Firstly, PPO is a model-free method. As the name suggests, this means the agent is not given access to a model of the environment which would allow prediction of state-transitions and rewards before taking actions. Though models have benefits, such as increasing the learning rate, they are complex to implement and tune [11]. Moreover, they suffer from bias which often causes performance in simulation to differ from reality. On the other hand, model-free methods learn in the environment purely by interaction, which is more suitable for this task as the agent doesn't have knowledge of the grid world before taking actions. Furthermore, with the orographic updrafts changing location each episode, the environment is dynamic so a model-free approach is recommended. With the greater ease of implementation and modification, this method is more popular among researchers in the field.

Crucially, PPO was chosen as it is a variant of policy gradient methods. Policy optimisation or training process can be thought of as maximising the reward with respect to the policy parameters. This cannot be calculated with a straightforward differentiation because the reward is a function of the actions and states, which are discrete. Policy gradient methods are therefore utilised to achieve this through approximation; the gradient is estimated by sampling large numbers of trajectories and calculating the gradient along them [12]. This is a trial-error-process where for actions that lead to a high average reward, their probabilities of occurring are increased using a gradient ascent algorithm. The large amount of trajectory sampling creates a drawback, however, as the method is sensitive to step size: too small steps causes policy gradient methods to become computationally expensive and too large steps results in the gradients becoming noisy hence reduced policy performance. Trust Region Policy Optimisation (TRPO) was introduced to overcome this by constraining the size of policy update, yet, in doing so it became complicated to compute. Consequently, PPO serves to retain the performance of TRPO [10] whilst being simpler to implement like gradient methods.

Schulman *et al.* [12] demonstrates the superior ability of PPO over other RL algorithms as observed in Figure 5. On average, PPO was shown to outperform all the other methods including a2c and TRPO therefore is the default algorithm for OpenAI and was selected for our wind-aware path planner.

Within PPO, various parameters that determine the performance can be optimised depending on the task. An example of this is the discount factor which controls how the agent balances the maximisation of long term reward against immediate reward. For finite MDPs this is critical for continuous tasks as they ensure the reward tends to a finite value even if the trajectory is infinite, as illustrated by Sutton *et al.* [11] in equation 3.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4}... \tag{3}$$

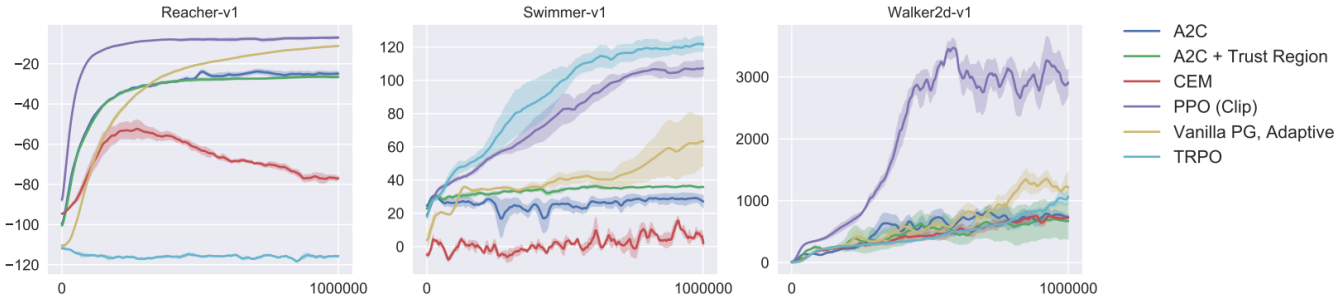


Figure 5: Reward comparison over one million time-steps for several algorithms in several environments. Modified from [12]

When the discount factor is in the range, $0 < \gamma < 1$, the rewards diminish over time and the smaller this factor is, the faster the set of rewards converge to a finite value. The goal of training a path planning policy for energy-based optimisation in the wind field environment requires trials with finite duration as the agent cannot make decisions continuously in a finite grid. Therefore, this task is episodic -the agent-environment interaction naturally breaks down into episodes without the need of a discount factor to balance immediate and future rewards. The parameter was therefore left at a value of 0.99, having minimal effect on the returns.
Other hyper-parameters, such as the learning rate of 0.003 and the step size of 2048, were left at the default value because the performance of PPO from Figure 5 indicates that they are robust hence tuning is not essential.

In order to execute the RL algorithm on the RL controller designed in Section 3.4, the environment was wrapped in neural network (NN) policy. The two main policy options that are suitable for PPO are the Convolutional Neural Network (CNN) and the Multi-Layer Perceptron (MLP). The MLP policy was chosen because it is useful for tabular data-sets and Markov Decision Processes are tabular solution methods [11].
Despite retaining the default configuration of two layers and 64 nodes for the neural network, the decision was made to evaluate the policy as deterministic instead of stochastic. A stochastic system is one which introduces randomness in the development of future states, therefore stochastic policies only maximise the probability of choosing actions, based on a probability distribution [11]. This can be interpreted as introducing noise into the system which is necessary when the agent's actions do not uniquely determine the outcome. An example is the OpenAI frozen lake environment [22] where the mission is to traverse an frozen lake with holes to reach the goal. Due to the slippery surface, there is a probability that the agent's action to move in a particular direction may be ignored. Based upon this definition, the energy-based path planner of this paper follows a deterministic policy since the agent's actions always change the state as expected. This has no noise or randomness so when the policy is evaluated it will converge to one solution it considers optimal in the environment.

3.6 Learning a Path Planning Strategy

After designing and wrapping the environment in a policy and algorithm, the Reinforcement Learning path finder was ready to learn a strategy in the wind-field. The main elements of the Markov Decision Process that influence the performance of the algorithm are the rewards.

As mentioned before, the rewards are the method of indicating to the agent how well it is doing [11], reinforcing good behaviour and discouraging bad behaviour in order to achieve a goal. Although the purpose of this research was to minimise the flight cost in urban areas, this differs from the reward. The flight cost is the interpretation of how energy efficient the learned

path of the agent is. To find this value, a cost of travelling through each cell must be determined. The designed formula to do this was,

$$\text{Local cost}(x, y) = 2 - 2 \cdot intensity(x, y) \tag{4}$$

where, $[x, y]$, was the cell position in question. The local cost and the intensity were both functions of the cell position in the Gaussian distribution (wind-field). The total flight cost was therefore the local cost at all cells along the path, or can be thought of as the average cell cost multiplied by the total path length. Since the intensity ranged from 1 to -1, the local cost ranged from 0 to 4; at a downdraft source the cost was 4, at an updraft source the cost was zero (ideal soaring). At cells far enough away hence unaffected by the orographic wind-field the cost was 2. Thus, the flight cost was be minimised by finding the path(s) that followed local costs closest to zero. This could not be used calculate the reward because the flight cost does not account for the fundamental requirement of a path planner -the agent must reach the goal. In Reinforcement Learning, the reward needs to encourage visiting new cells to aid exploration of the environment and locate the goal. Only once a strategy to find the goal is developed can the flight cost formula be utilised to optimise the path taken. In practice, however, it is not possible to 'change strategies' mid-way through training. Sutton *et al.* [11] explains how the reward must be set up only to indicate what we truly want to accomplish, not how to achieve it.

The solution for this was to define multiple reward parameters that established optimal and sub-optimal behaviour for our goal, and to vary their weighting in the singular returned reward value through testing. For this task, these were divided into two types: basic path planning parameters and energy-based parameters.
Firstly, for the basic parameters, a visit reward was created to encourage visiting new cells and discourage returning to a previously visited location. This aids exploration as discussed above. Secondly, a goal reward was necessary to ensure that once the agent first reaches the goal, it continued to do so. Thirdly, an obstacle reward was designed to prevent the algorithm from crashing into buildings. The latter two conditions were also set as terminal states. Terminal states are indirectly a form of reward as they prevent the agent from interacting with the environment further in the episode. The basic path planning parameters alone were necessary to plan an optimised path with PPO in a grid without a wind field (as will be shown in Sections 4 and 5).
The energy-based rewards were necessary when the wind field was introduced. The orographic reward represented the intensity of the orographic lift at each cell, hence was a function of the Gaussian distribution. The time-reward parameter was introduced to model the finite battery capacity of the small unmanned aerial vehicle. The battery will discharge during the journey of the aircraft, therefore this parameter discourages the agent's behaviour with greater impact as the path length increases. This was achieved by setting the time-reward as negative (indicating a discouraging factor) and increasing the magnitude of the value as a function of the route length, shown in equation 5e. The final reward parameters found, after tuning, are shown below.

$$\text{Visit reward} = 1(\text{for new cell}) \text{ or } -1(\text{for previously visited cell}) \tag{5a}$$

$$\text{Goal reward} = 1000 \tag{5b}$$

$$\text{Obstacle reward} = 0 \tag{5c}$$

$$\text{Orographic reward}(x, y) = 0.2 \cdot intensity(x, y) \tag{5d}$$

$$\text{Time reward}(length) = -0.002 \cdot length \tag{5e}$$

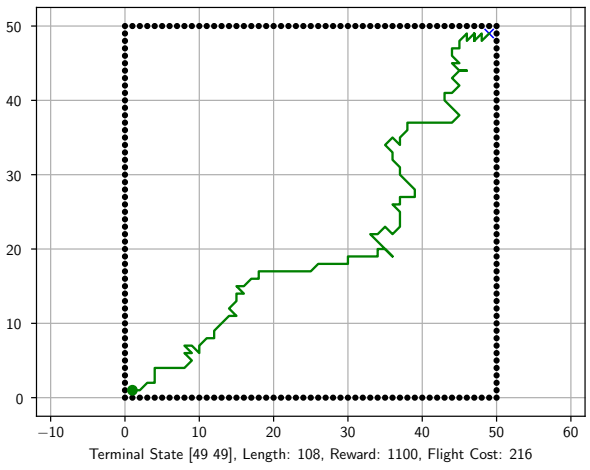
The process of learning a strategy over a number of time-steps, and training a policy through agent-environment interaction, should ultimately result in more desirable actions taken at different states. The progress was verified by testing the RL algorithm for ten episodes in the environment before and after learning; before the agent took random actions and after the agent predicted the most optimal action. This transition was achieved within the policy as it balanced taking random and predicted actions during training. Though in training the policy was evaluated using a deterministic method (Section 3.5), in the post-learning tests it was more beneficial to predict the actions stochastically. This is because the deterministic alternative only shows the path that the policy determines optimum and this is based on the reward. With a stochastic approach, the agent was able to plan paths with slight variations from this so it could be observed if any of the other solutions had a lower flight cost.

4 RESULTS

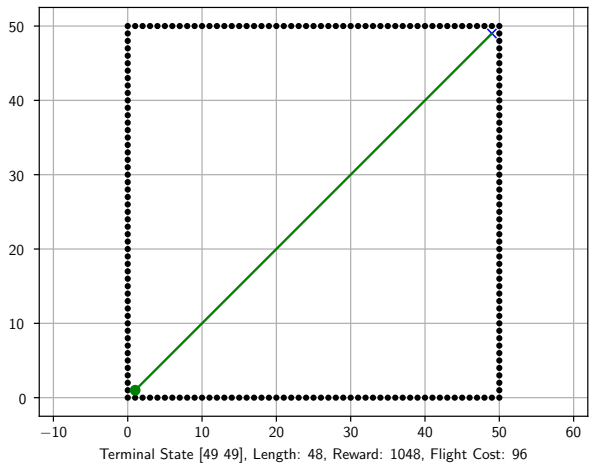
The RL algorithm was tested in multiple environments which included both controlled and variable-wind-field experiments. Figures 6a-14a showcase the performance of the agent, by displaying grids maps with the path taken for different tasks. Figures 6b-14b plot the episode length mean and reward mean achieved by the agent over the time-steps which training was completed. These plots were obtained by uploading the 'roll-out' data from the algorithm's training to Tensorboard and then presenting the figures with MATLAB. Due to the fact that the raw RL training data was noisy, a smoothing function was run over the data with a factor of 0.04. This value was chosen by trialling different factors, observing whether the curve clearly illustrated the agent's behaviour whilst minimising loss in the averaging process.

4.1 Control Experiments

The controlled experiments had no random features and each test served to evaluate a particular element of the RL algorithm's performance. These are given in Figures 6a-11a, where the right set of grid maps reveal the path determined optimal by the policy, at the end of training, and the left set of grid maps provide some useful observations.

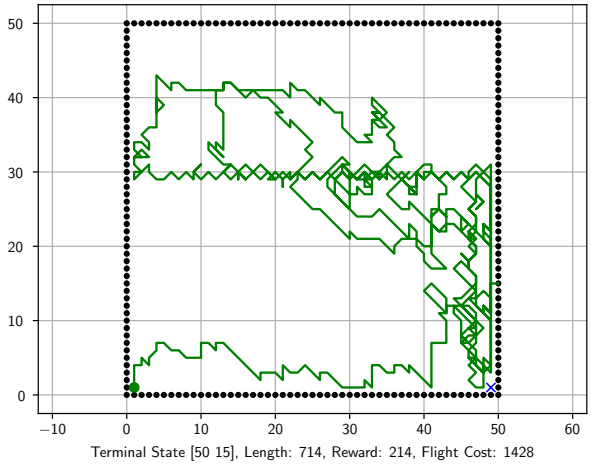


Sub-optimal path, during training.

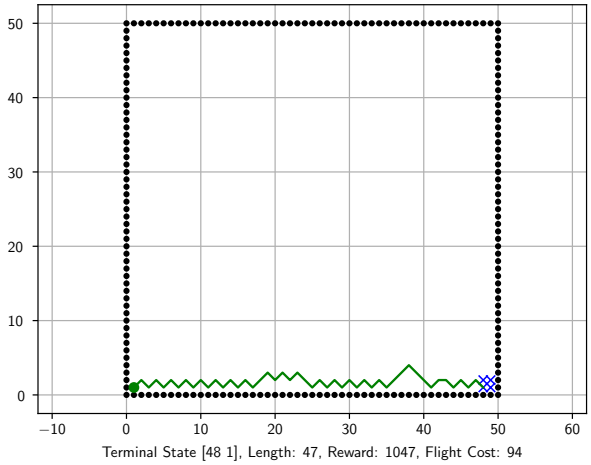


Optimal path, post-training.

Figure 6a: Simple grid maps; start position is in bottom left corner, goal is in top right.

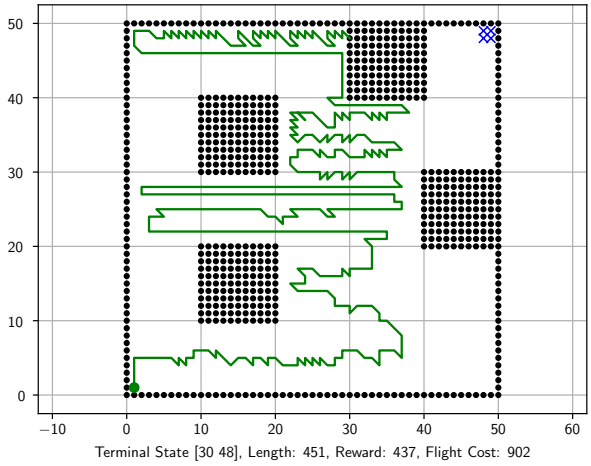


Exploration, during training.

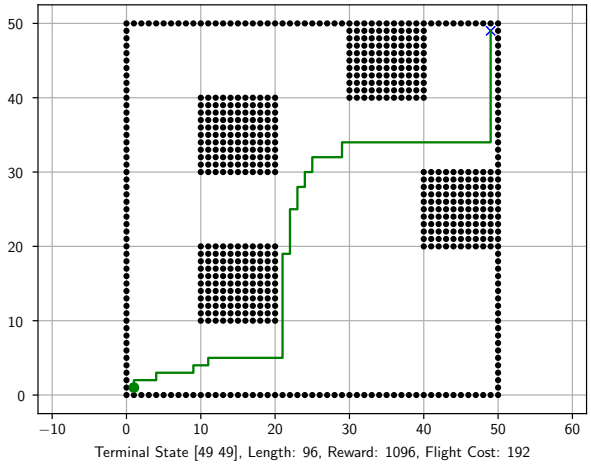


Optimal path, post-training.

Figure 7a: Simple grid maps; start position is in bottom left corner, goal is in bottom right.

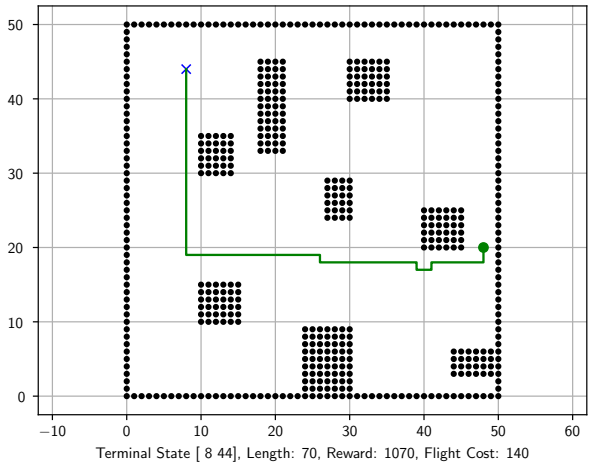


Exploration, during training.

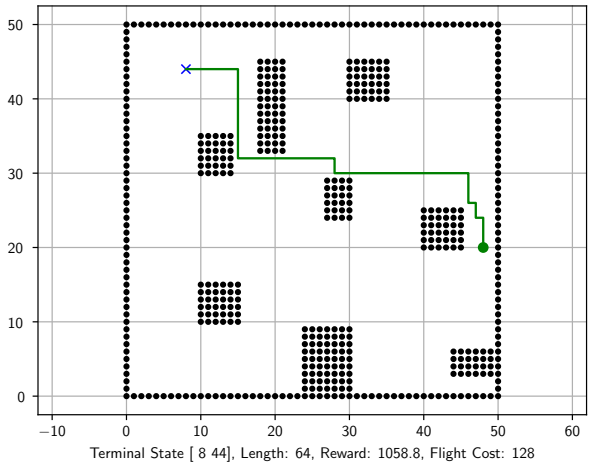


Optimal path, post-training.

Figure 8a: Simple obstacle maps.

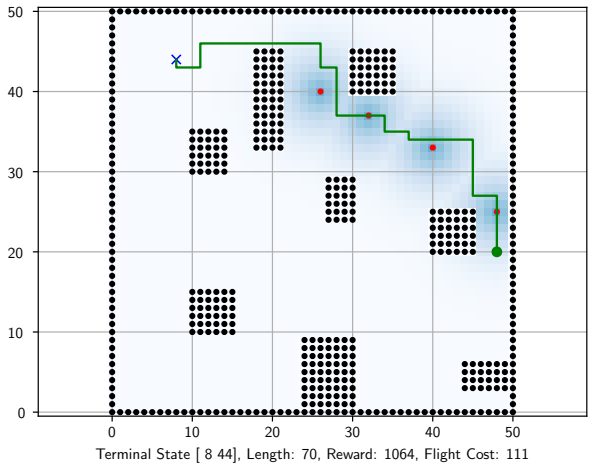


Sub-optimal path, during training.

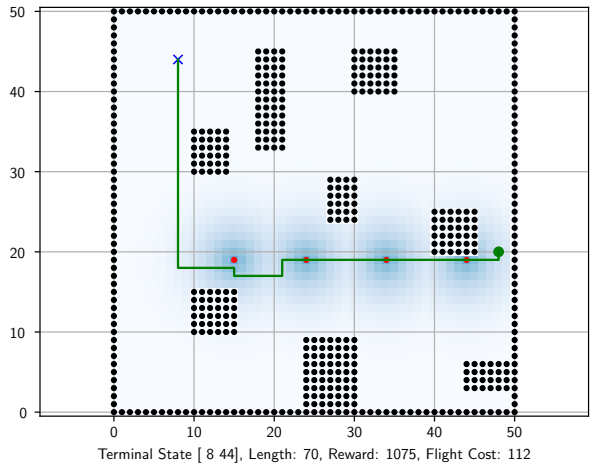


Optimal path, post-training.

Figure 9a: Complex obstacle maps.

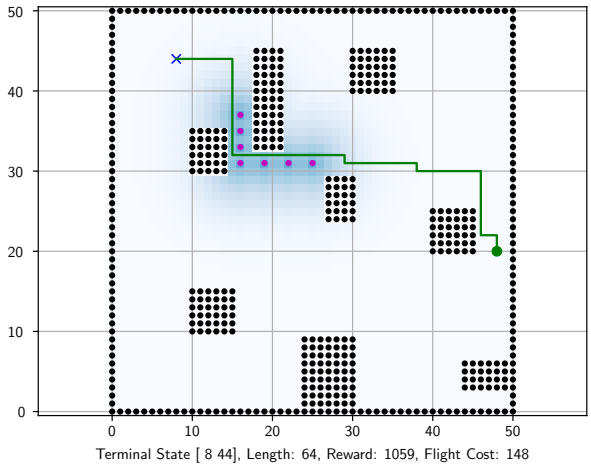


Updrafts above start point, post-training.

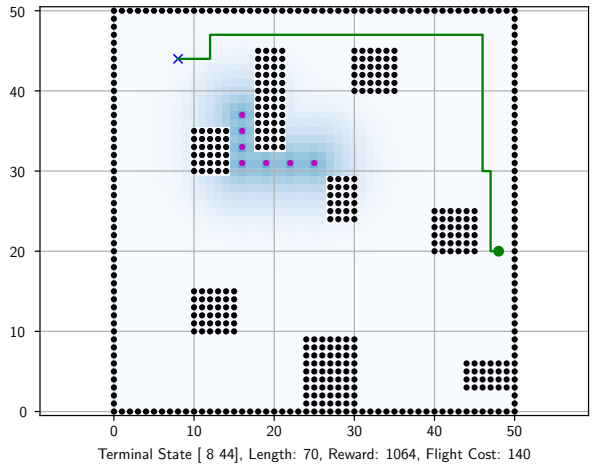


Updrafts below start point, post-training.

Figure 10a: Complex wind-field maps with updrafts.



Result of taking shortest path, ignoring downdrafts.



Downdrafts along shortest path, post-training.

Figure 11a: Complex wind-field maps with downdrafts.

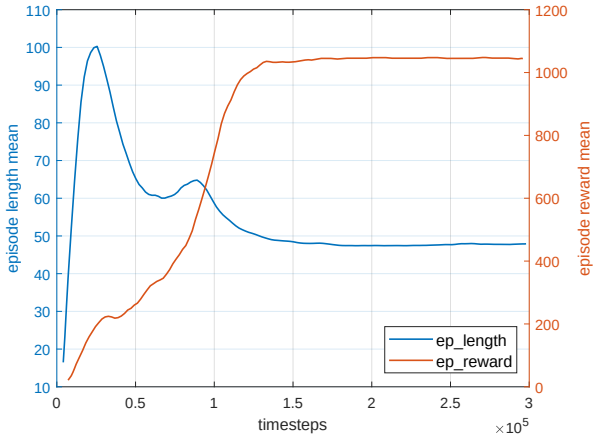


Figure 6b: Training data for simple grid ([1,1] to [49,49]).

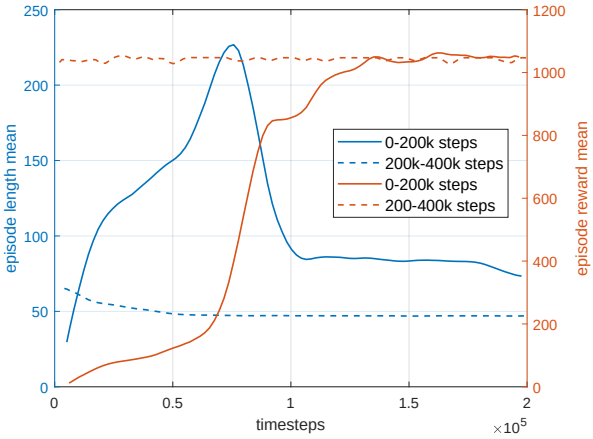


Figure 7b: Training data for simple grid ([1,1] to [49,1]).

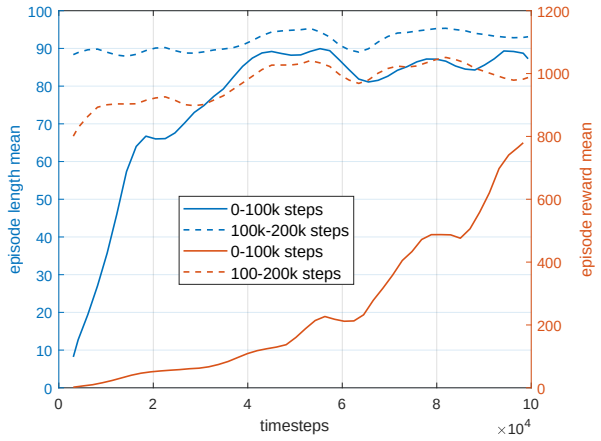


Figure 8b: Training data for simple obstacle map.

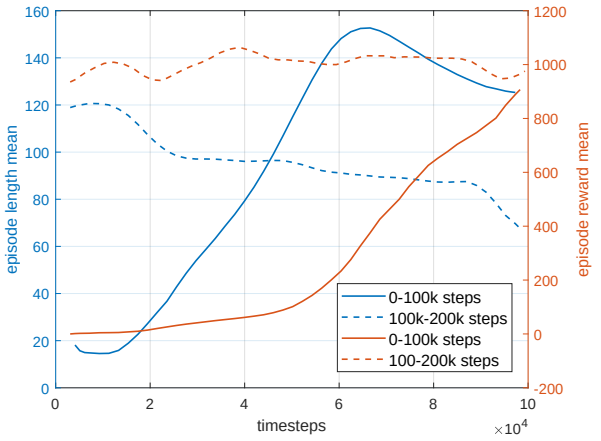


Figure 9b: Training data for complex obstacle map.

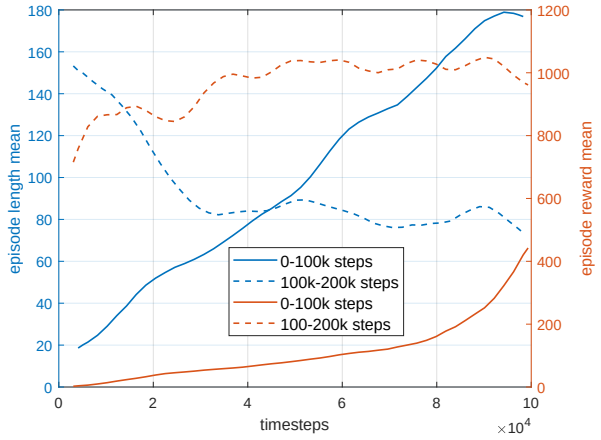


Figure 10b: Training data for complex wind-field map with updrafts.

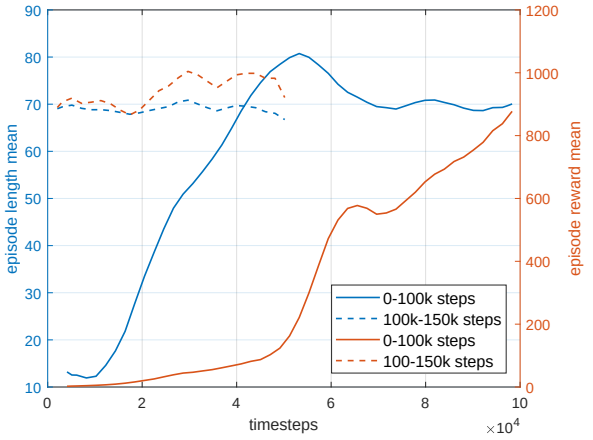


Figure 11b: Training data for complex wind-field map with downdrafts.

Simple Maps:

Figures 6a and 7a present the control tests for the simplest grid maps. There were no wind-fields active in these environments so the purpose of the tasks were to observe if the algorithm could successfully plan a path to the goal and by the optimal route -in this case the shortest path. Accordingly, only the goal, visit and obstacle reward parameters were employed. The results of Figure 6a identified that the planner could traverse the furthest distance in the grid ([1,1] to [49,49]) in order to reach the goal. Moreover, this was achieved with an obstacle reward of zero; the fact that this parameter is a terminal state was enough to discourage the agent from crashing into obstacles (the map boundary). The agent was also shown to initially reach the goal through a longer, sub-optimal, path but with further training the policy's deterministic solution was the straight-line shortest route. This is verified in Figure 6b and 7b as the path lengths decreased from a peak value until they stabilised to a length of 48.

On the other hand, Figure 7a exhibited the shortcomings of the algorithm as when the goal was moved to cell [49,1] the agent could no longer find it. That said, this result was still informative as there was insight into the method of exploration, which was shown in this figure to be predominantly random. This indicates the route taken did not follow any systematic plan of covering all the cells, reducing the probability of locating the extreme position of the goal. The end-point was therefore changed to a, larger, 2x2 region and the agent found it with the shortest path. The reward vs time-step plot in Figure 7b conveyed this with a rapid increase in

reward as the planner quickly adapted its actions to reach the goal. A final observation is that the optimal path obtained by the agent post training was unexpected when compared to the sub-optimal path during training, as the reward decreases.

Obstacle Maps:

In both obstacle grids, the agent struggled to find the goal, even when it was a larger size. The exploration of Figure 8a resembled more of a pattern to Figure 7a as there was now a clear 'zig-zag' path that aimed to cover more new positions. It was still imperfect, however, as it contained a high level of noise. Due to the lack of success in reaching the endpoints, the action space was reduced to four moves: North, South, East, and West. Applying this change, the algorithm in the simple map solved for the optimal route hence an action space of four was taken for the following, more complex environments.
For Figure 9a, this change alone did not solve the complex obstacle map. Although the agent found the goal, the route was sub-optimal. Therefore, in addition to the three basic path planning parameters the time-reward was introduced, and the agent was able to locate the shortest path (right map). This is illustrated in Figure 9b as the reward plateaus once the goal was found but the episode length continues to decrease for optimisation.

Wind Field Maps:

When the Gaussian filter was introduced with updrafts in Figure 10a, the agent was shown to take longer routes than the non-wind field equivalent. Furthermore, the direction of the agent suggested that it was attracted towards the updraft sources as well as the goal point. This was expressed by the high path length during training in Figure 10b, which was due to the agent's increased exploration around the updraft sources. To a similar effect, Figure 11a portrays the agent's avoidance of the downdrafts despite costing additional length to reach the goal. The path length in Figure 11b was lower here than the updraft and no-wind field case, suggesting that the negative rewards of the downdraft wind-field discouraged exploration of the environment. The shortest path displayed in this flow field shows that the path chosen by the algorithm achieved a lower flight cost than when ignoring the downdrafts.

4.2 Variable Wind Field Experiments

The variable wind field experiments involved testing the agent in an environment with random updraft type and locations every episode. This was completed on two grid layouts: the complex map and a realistic city map. Due to size and simplicity constraints of the grid map, the realistic city map could only be inspired by an urban area rather than an accurate representation. This area was taken to be a part of Canary Wharf in London, England, and further information on how this was achieved can be found in appendix B.

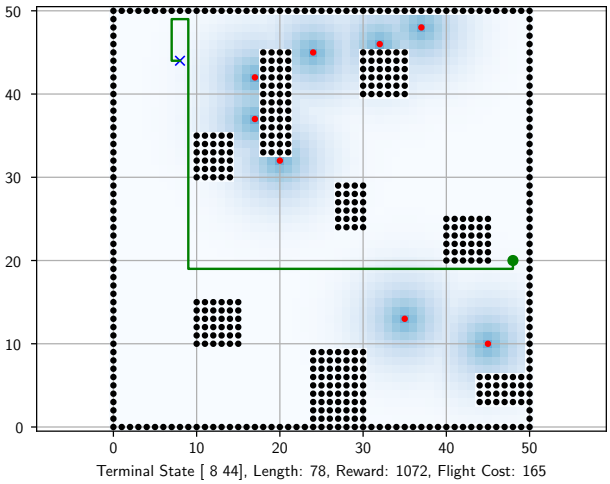


Figure 12: Complex variable-wind-field map.

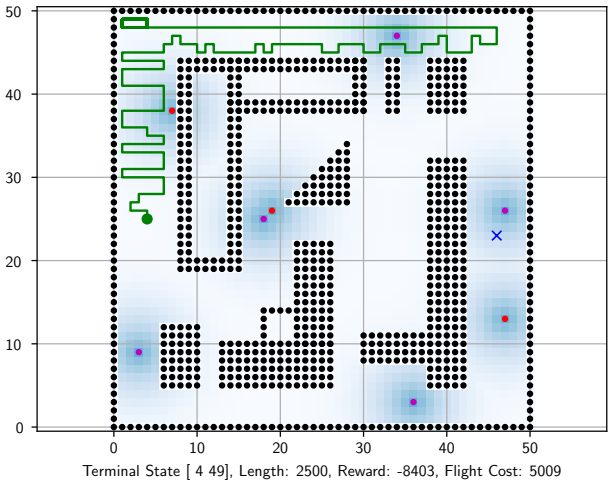
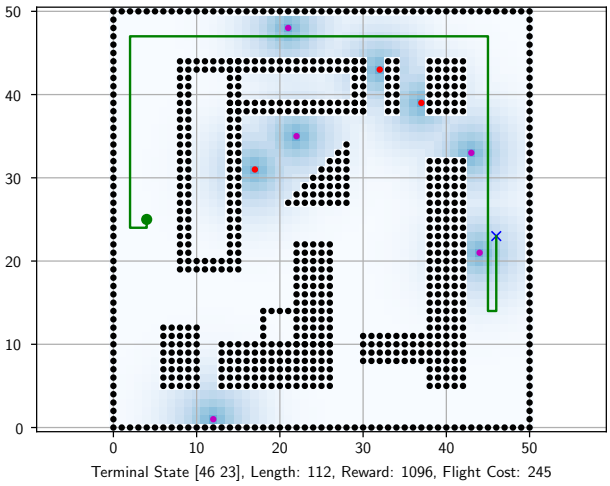


Figure 13a: Realistic variable-wind-field map with 3x Orographic reward.



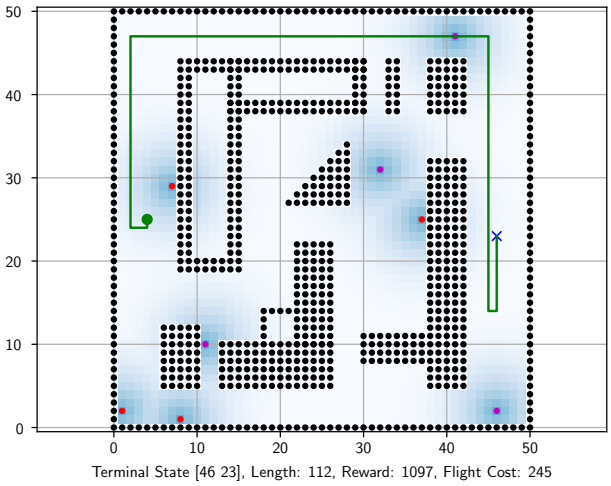


Figure 14a: Two episodes for the realistic variable-wind-field map, post-training.

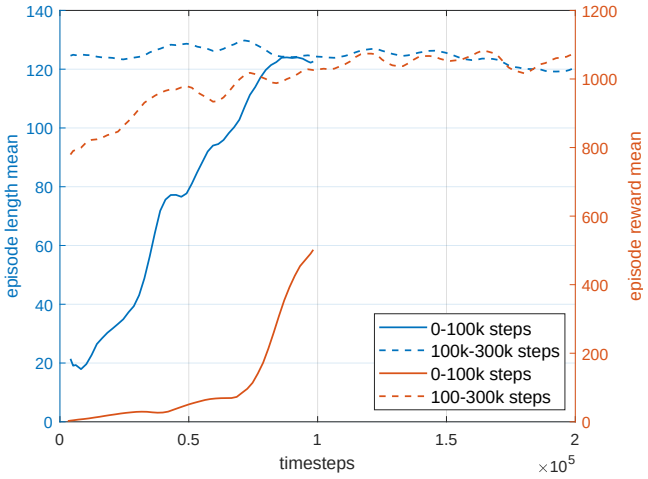


Figure 14b: training data for realistic variable-wind-field map.

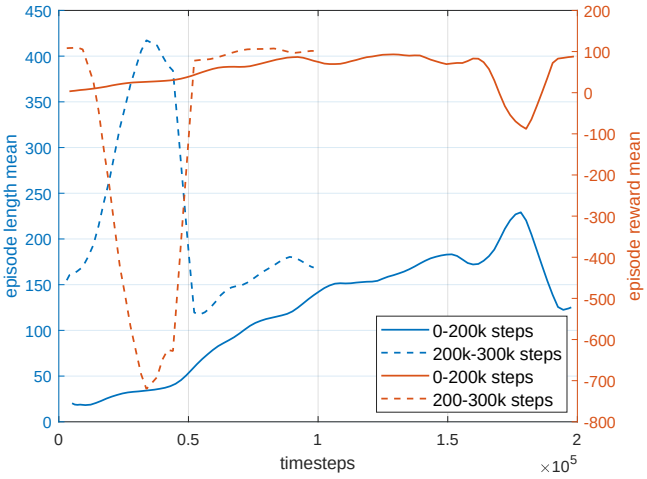


Figure 13b: training data for realistic variable-wind-field map with 3x Orographic reward.

For all of the variable wind field experiments, the policy was trained over 300,000 time-steps. Figure 12 displayed one episode of the variable wind field mission in the complex map. At the end of training, the route determined optimum by the policy was not the true optimal; this specific episode should have followed the same path as the optimal route in Figure 9a.
For two different episodes of the realistic variable-wind-field maps presented in Figure 14a, although the agent reached the goal the path was not optimal. Furthermore, the final paths determined were the same in both episodes despite the changing wind-fields. An attempt to overcome this similarity was taken by increasing the weighting of orographic reward parameter, given in Figure 13a, however the agent was unable to reach the goal. Figure 13b illustrates this, as the reward did not increase much over the 300,000 steps. The dashed curves in Figure 13b show a simultaneous increase in episode length mean and decrease in episode reward mean. This identified that during training, the agent avoided the (obstacle) terminal states for the maximum episode duration -displayed in Figure 13a- however it retraced its visited cells causing the large negative reward displayed.

5 DISCUSSION

Evaluating the training data, the episode length and reward length vs time-step graphs illustrated a consistent characteristic behaviour. The large increase in the reward occurring in the first half of training identified that the agent was learning to reach the goal. Once this was achieved consistently the algorithm stopped learning; the reward plateau indicated that the policy settled to a strategy that it determined optimal for the environment. The length mean curve increased in the intermediate stages of the training which demonstrated exploration of the algorithm. This occurred over similar time-steps to the large increase in reward, suggesting that most of the learning occurred during high exploration. The length then decreased gradually during the plateau of the reward, highlighting how now that the planner was consistently finding the goal, it was slowly optimising its route taken.

5.1 Control Experiments

Simple Maps:

The exploration revealed in these tasks conveyed that the agent's method of interacting with the environment is inefficient compared with conventional algorithms that follow a set of rules. For example, the A* Algorithm uses its full knowledge of the map environment to search in the direction of lowest total cost [6]. The Reinforcement Learning algorithm was shown to take more random actions initially, except for when it was near obstacles as it was able to quickly learn to select actions that avoided crashing. In Figure 6a, the decrease in reward from the path during training to the end of training was unexpected, because the three basic reward parameters do not discourage the path planner from taking a longer route. In fact, to gain the highest reward, the agent should have visited all the cells once before reaching the goal -to get the maximum reward in this environment of 3498. An explanation for this observation, is that the PPO algorithm with MLP policy aimed to maximise the reward mean rather than the reward per episode; reaching the goal faster allowed the agent to achieve the high goal reward more often, so over a given number of time-steps the reward mean is maximised. This trait meets the requirement of the algorithm to find the optimal path as can be seen by the fact that the flight cost was lower for the post-training optimal path.

Obstacle Maps:

Reducing the action space in the obstacle environments enabled the agent to reach the goals as it reduced the number of options that the policy needed to (learn to) predict from. This can

be understood as reducing the randomness and noise in the policy's actions thus allowing the agent to explore more effectively. Despite simplifying the map this way, and further deviating from reality, the agent was shown to reach optimal performance for the environments.
In Figure 9a, the path planner's difficulty in finding the optimal path was expected, as this environment was deliberately designed to make the optimal routes require passing through the narrow gap between the two buildings closest to the goal. The fact that the algorithm succeeded with the time-reward term showed that introducing the real-world concept of a battery capacity in the agent's (SUAV controller's) decision-making was necessary for finding an optimised path in a complex urban area. The impact of this parameter on the RL algorithm can be understood by referring back to Section 3.6, where it was defined as a function of the path length (or equivalently time) that gets increasingly negative when the episode continues. Therefore, the parameter acts as a limiter for exploration, that is enforced with greater impact as the episodes reach greater lengths. With a weighting factor of -0.002 from Equation 5e the reward passes -1 per move after 500 steps. The comparison of the optimal and sub-optimal paths in Figure 9a highlight how this limit to exploration encourages the agent to find a shorter path.

Wind Field Maps:

The wind field was included in the control experiments in order to investigate how the path planning algorithm responded to orographic effects when traversing the grid. In taking the longer routes (than optimal Figure 9a path) to the goal, in Figures 10a and 11, the path planner is shown to be more than just a shortest path algorithm. This is because agent was aware of the wind field and by following the path of the updrafts or avoiding the path of the downdrafts, the policy was shown to understand the difference between energy harvesting and energy addition in the urban environment. Its ability to recognise the benefits of soaring was highlighted in the flight costs. The cost of the updraft paths were lower than the non-wind field shortest path, and for the downdraft paths the two maps demonstrated that the energy-based route reduces costs over the theoretical shortest path. Moreover, for the same path length of 70, the flight cost of Figure 11a was the same as Figure 9a. This conveyed that the agent discerned the downdraft flow field to be sub-optimal, hence chose a route that avoided it entirely.

5.2 Variable Wind Field Experiments

The unchanged route across the two episodes of Figure 14a suggested that the agent was unable to recognise the energy saving of orographic lift. The reason for this, was the wind field intensity represented in the orographic reward parameter was too low for the algorithm to be able to recognise the benefits of updrafts and drawbacks of downdrafts. This is supported by the control complex-wind-field map experiments in Figures 10a and 11a, as the agent was only able to recognise orographic effects because the sources were positioned in extreme, unrealistic, locations for the entirety of training. Moreover, in these static maps the algorithm still took over 100,000 time-steps to consistently follow or avoid the updrafts or downdrafts respectively. This indicates that for environments with a dynamic wind-field, a much longer training cycle was necessary to accurately assess the performance of the path planner. This conclusion is consistent with Bøhn *et al.'s* investigation into the attitude control of fixed-wing UAV's using PPO [10], as when their controller was trained in changing environments it took around two-million time-steps to converge to an optimal policy. The observations of Figure 13a conveyed the problem with trying to improve performance by increasing the wind field intensity; the agent became too focused on maximising reward by interacting with the orographic sources, that it failed to explore the full map and find the goal.
The sub-optimal routes produced in these variable experiments can also be attributed to the changing wind field, as although the agent failed to correctly interpret the orographic effects, they still added complexity to the learning process of the algorithm thus reducing performance.

Figure 14b emphasized this difficulty since the agent was only able to consistently find the goal after 200,000 time-steps -noticeably slower than the control experiments. Bøhn *et al.'s* study [10] similarly found that increased complexity resulted in a slower rate of improvement in the algorithm, and reducing the size of the observation space was necessary to increase learning.

6 LIMITATIONS AND FUTURE WORK

Section 3.1 described how the urban city environment was modelled in a two dimensional grid. Although the work of Shepard *et al.* [14] explored how orographic lift can be reduced to 2D, there was still a loss of accuracy in the altitude-averaging process. Therefore, the optimal path determined by the algorithm may not be true to the reality. Additionally, using this discretised environment limited the complexity to which urban structures could be recreated. This is because the grid divides objects into square-shaped cells which is not ideal for complex geometries, such as the buildings in Canary Wharf for the realistic city map. Future work could consider the use of triangular-shaped cells rather than squares, taking inspiration from 'UnrealEngine 5', where billions of triangles were used to model complex geometries in advanced game development [23]. The realistic environment was also restricted due to the action space. By reducing the possible moves to four, the agent often had multiple optimal paths to reach the goal; when traversing from the bottom-left corner to top right, any combination of North and East actions would result in the shortest path. Consequently, the arrangement of obstacles and endpoints had to be carefully chosen to prevent this, limiting the real city maps available to test the path planner's performance. Diagonal pathing would reduce the number of optimal paths in the map, yet the obstacle map results argued this led to a reduced path planner performance. Following research could investigate whether hyper-parameter tuning with a custom policy could successfully find optimised paths for the eight-action trials.
Although the results suggested longer training cycles could improve the policy, a limitation of Reinforcement Learning is that too much training can cause overfitting [11]. This is when the training data exactly fits the learned strategy, so the algorithm is unable to perform in an unseen environment. Thus, the alternative proposal made by Reddy *et al.* [17] to improve performance with a model-based approach could also apply to this research. Using a model would provide the agent with map knowledge in order to plan in the environment hence the RL algorithm would be more comparable with conventional techniques for future study.

Beyond the grid, the approximation of an orographic wind field with a Gaussian filter introduced uncertainty. Though literature supported the use of GPs for estimating wind fields, the same cannot be assumed for grid mapping. Even with 2500 cells a large amount of information is lost in the discretisation process, and Lawrance *et al.* [19] concluded that the number of grid positions required to provide reasonable resolution would be too computationally expensive. An alternative approach could employ a smoothing function such as the 'common squared exponential radial basis function' used by Lawrance to more accurately approximate the GP. In the generation of the updrafts and downdrafts, sources were taken to be random in the environment, however, they should only be found close to buildings as this lift is created through wind-building interactions [15]. Future work use the research by Shepard *et al.* [14] displayed with Figure 3 to model orographic lift parallel to the face of buildings, positive on one side and negative on the opposite. A final limitation, was that the wind intensity at each cell in the grid was only determined by the closest orographic source. This diverges from reality, as there could be regions of overlapping updrafts and downdrafts hence further investigations should consider the effect of the entire wind field on each cell.
Although the map was simplified and the flow field did not accurately represent the real world, the control experiments still demonstrated that the agent could recognise and optimally respond to the presence of obstacles, a goal and static energy saving regions within an environment.

7 CONCLUSION

In this study, to investigate the ability of a path planning algorithm for utilizing orographic soaring, in order to predict the path of minimum flight costs in an urban environment, a Reinforcement Learning approach was proposed. This was achieved by simplifying urban areas into two dimensional grids, then modelling the orographic wind field with a Gaussian filter. The RL path planner was designed using Open AI gym's PPO method and the agent was made to learn in the environment through interaction, over a number of time-steps. The key findings on RL algorithms, obtained from the results were: the PPO method aimed to maximise the reward mean across training instead of the reward per episode -meaning the algorithm learns for optimization by default- and the exploration of the environment with an RL approach follows a highly random, noisy, path which is less efficient than conventional algorithms. Overall, this investigation revealed that, while an RL approach could minimise flight costs in static wind-field maps, in dynamic wind-field urban environments the algorithm was unable to make use of orographic soaring to achieve this goal.

REFERENCES

[1] Ali Shuja Siddiqui, Mohammad Ali, Jinnah University, Pakistan Karachi, Sameer Qazi, and Asim Imdad Wagan. UAV based real time video surveillance over 4G LTE. *ieeexplore.ieee.org*.

[2] CA Thiels, JM Aho, SP Zietlow, DH Jenkins Air medical journal, and undefined 2015. Use of unmanned aerial vehicles for medical product transport. *Elsevier*, 34:104–108, 2015.

[3] Rodrigo L. Mota, Luiz F. Felizardo, Elcio H. Shiguemori, Alexandre B. Ramos, and Felix Mora-Camino. Expanding small UAV capabilities with ANN: A case study for urban areas observation. *2013 IEEE 2nd International Conference on Image Information Processing, IEEE ICIIP 2013*, pages 516–520, 2013.

[4] Emmanuel Avian, N Gavrilovic, A Mohamed, M Marino, S Watkins, J-m Moschetta, and E Be nard. Avian-inspired energy-harvesting from atmospheric phenomena for small UAVs. *iopscience.iop.org*, (1):1–20, 2018.

[5] Scott A. Bortoff. Path planning for UAVs. *Proceedings of the American Control Conference*, 1:364–368, 2000.

[6] C Williamson. *Bio-inspired path planning for unmanned air vehicles in urban environments*. PhD thesis, University of Bristol, Bristol, 7 2020.

[7] Brian Dunbar. Autonomous Soaring — NASA. *NASA*, 8 2017.

[8] Max Harvey. Soaring UAV Path Planning Through Restricted Urban Airspace. Technical report, University of Bristol, Bristol, 4 2021.

[9] Nicholas R J Lawrance. *Autonomous Soaring Flight for Unmanned Aerial Vehicles*. PhD thesis, University of Sydney, 2011.

[10] Eivind Bøhn, Erlend M Coates, Signe Moe, and Tor Arne Johansen. Deep Reinforcement Learning Attitude Control of Fixed-Wing UAVs Using Proximal Policy Optimization. *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2019.

[11] Richard S Sutton and Andrew G Barto. Reinforcement Learning: An Introduction. *MIT Press*, (2018).

[12] J Schulman, F Wolski, P Dhariwal, A Radford arXiv preprint arXiv ..., and undefined 2017. Proximal policy optimization algorithms. *arxiv.org*.

[13] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba Openai. Openai gym. *arxiv.org*.

[14] Emily L.C. Shepard, Cara Williamson, and Shane P. Windsor. Fine-scale flight strategies of gulls in urban airflows indicate risk and reward in city living. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 371(1704), 9 2016.

[15] Ana Guerra-Langan, Sergio Araujo-Estrada, and Shane Windsor. Unmanned aerial vehicle control costs mirror bird behaviour when soaring close to buildings:. *https://doi.org/10.1177/1756829320941005*, 12, 8 2020.

[16] CJ Williamson, A Spelt, SP Windsor AIAA Journal, and undefined 2021. Bird velocity optimization as inspiration for unmanned aerial vehicles in urban environments. *arc.aiaa.org*, 59(7):2503–2516, 2021.

[17] Gautam Reddy, Jerome Wong-Ng, Antonio Celani, Terrence J. Sejnowski, and Massimo Vergassola. Glider soaring via reinforcement learning in the field. *Nature 2018 562:7726*, 562(7726):236–239, 9 2018.

[18] Hannah J. Williams, Andrew J. King, Olivier Duriez, Luca Börger, and Emily L.C. Shepard. Social eavesdropping allows for a more risky gliding strategy by thermal-soaring birds. *Journal of the Royal Society Interface*, 15(148), 11 2018.

[19] Nicholas R.J. Lawrance and Salah Sukkarieh. Simultaneous exploration and exploitation of a wind field for a small gliding UAV. *AIAA Guidance, Navigation, and Control Conference*, (8032), 2010.

[20] Normal cumulative distribution function - MATLAB normcdf - MathWorks United Kingdom.

[21] States, Observation and Action Spaces in Reinforcement Learning — by #Cban2020 — The Startup — Medium.

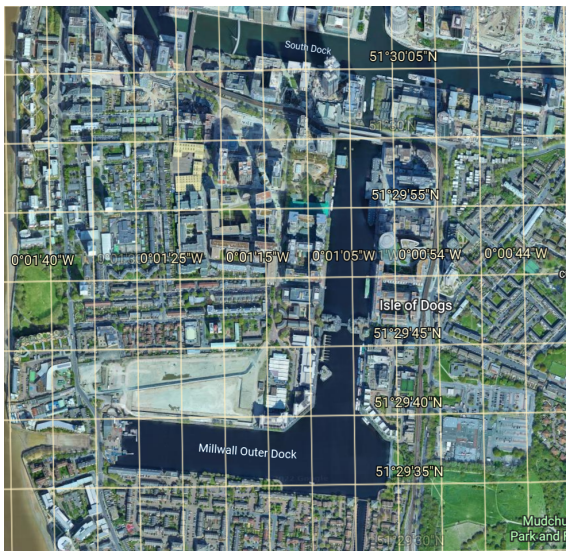
[22] S Ravichandiran. *Hands-on reinforcement learning with Python: master reinforcement and deep reinforcement learning using OpenAI gym and TensorFlow*. Packt Publishing Ltd, 2018.

[23] Nanite Virtualized Geometry in Unreal Engine — Unreal Engine Documentation.

A GITHUB REPOSITORY FOR RL PATH PLANNING ALGORITHM

https://github.com/PrithviAbhishetty/rp3

B REALISTIC MAP INSPIRATION



Google Earth 2D view of Canary Wharf, London, England.

Close up view of Canary Wharf.

When attempting to model the urban area of Canary Wharf as a whole, it was deduced impractical for the discrete grid. This is because, given the 50x50 cell positions, the buildings would've had to be modelled as one or two cell obstacles, which would fail to capture the complexities of urban environments. Consequently, a smaller section of Canary Wharf was considered, that allowed the arrangement of buildings to be captured. The close-up-view presents tall buildings condensed in one area, thus, it was appropriate to design a grid layout with a similar compact arrangement of buildings. Furthermore, the 'L' shaped and triangular-shaped buildings selected for the final realistic grid map were inspired by the buildings in this figure.