Dynamic Obstacles Avoidance in Coverage Path Planning Via Deep Reinforcement Learning

Raymond Chong
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Sutardja Dai Hall, Berkeley, CA 94704
`raymondchong@berkeley.edu`

Carl Chua
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA 94704
`carlchua@berkeley.edu`

Abstract

In the current state of the art for deep reinforcement learning, the primary focus is on maximizing an objective function. Both coverage path planning and dynamic obstacles avoidance present groundbreaking results using RL and traditional path planning algorithms. However, by combing the two objective into one reward function, it will be an interesting, yet difficult, experiment for the agent to learn from. We vary algorithms between A2C, SAC, PPO alongside adding several different observations states, memory property (LSTM), and Generalized Advantage Estimate (GAE) to help further fine tune the agent's learning experience. The results showcase that the agent is capable of handling both task as an integrated objective. The application for this project is: Given a farmland with clouds, we hope to investigate the optimal trajectory for an agent to navigate and cover all ground on the farmland where the clouds are moving obstacles that the agent needs to avoid that minimizes time, conduct coverage of the grid and avoid clouds as modeled with obstacles.

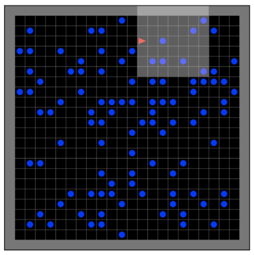
1 Extended Abstract

With the advancement in reinforcement learning, there has been many applications ranging from image/voice recognition, text translation to robotics. For example, there has been much literature and ongoing research focusing on path planning utilizing reinforcement learning technique. Within path planning, there are ongoing research focusing on reachability analysis, optimal control and global minimum cost for multi agent systems and many more. In many existing path planning algorithms (both graph based or learning approach), some of the main objectives were minimizing cost, shortest path, from starting point to destinations. Examples of traditional path planning algorithms, such as Dijkstra's, A* and Sample algorithm, are beneficial but often times they are over simplified to represent real life application. Hence, with a learning based approach, the agent can learn and adopt to dynamic environment to further better represent the changes that occurs in reality. However, within learning-based literature, few research focus on coverage path planning and more precisely **coverage path planning with dynamic moving obstacles avoidance** give an entire grid. It is fathomable that with the recent advances in machine learning, agents can also attempt to learn both coverage while avoiding dynamic moving obstacles, and this serves as the primary motivation for this project.

In this project, the agent attempted to learn to conduct coverage of the entire environment while avoiding moving obstacles via a mixed with deep reinforcement learning algorithms, namely varies actor-critic (SAC, A2C) and Proximal Policy Optimization (PPO). Additionally, the neural network architecture will start off with a CNN model on the Minigrid environment for path planning. Later, we added the LSTM architecture to introduce memory for the agent. As well, we introduce an additional observation state: previous frames look back (agent will have the capability of observing and keeping track of the 5 previous iterations of the obstacles). This should help the agent better understand the dynamic of the obstacles since all the obstacles are always flowing from left to right (similar to clouds).

The application for this project is: Given a farmland with clouds (and shadows), we hope to investigate the optimal trajectory for a vehicle (such as drone) to navigate and cover all ground on the farmland where the clouds are moving obstacles that the vehicle needs to avoid that minimizes time, conduct coverage of the grid and avoid clouds as modeled with obstacles.

The research project revolves around using the Minigrid environment Given an NxN gird environment, the agent has partially observable view of the environment MxM (the transparent white mask) where M <= N. The goal for the agent (red arrow) is to avoid the obstacle (blue circles) while covering every tile on the NxN grid.


View Every Tile

The full complete research consist of three stages.

The first stage is using a video based LSTM-CNN model to learn and prediction cloud prediction/movement. For this, I used algorithms based on Srivastava's Unsupervised Learning of Video Representations using LSTMs [1]. Using the image based prediction, we can more accurately predict and understand the dynamics of the cloud (growing bigger following the direction of the wind speed and etc...).

The second stage is applying path planning algorithms and using the trained neural network from first stage. There has been several exploration analysis done on related path planning algorithms. For example, traditional algorithms such as A*, D* and other Sampling-based algorithms [2], [3]. However, there were limitation with these algorithms as it's not capable of having dynamical moving obstacles. As a result, we were in search for a learning based approach such as reinforcement learning.

The final stage is to combine the ML prediction with RL and path planning algorithm to develop the optimal trajectory for the drone's objective (coverage and collision avoidance in dynamic obstacle settings). From there, we will have to consider the dynamic of the drone and incorporate the speed of the wind and the velocity of the drone.

The focus on this report was on the latter part of the second stage. The obtained results demonstrate the RL agent's capabilities to learn and plan a reasonable (cost efficient) path that consider both dynamic obstacles avoidance and coverage in a reasonable time frame. Current work over winter break involved introducing adversarial environment for cloud movement to help robustify the RL agent by utilizing Emergent Complexity and Zero-shot Transfer via Unsupervised Environment Design (PAIRED), a curriculum learning methodology [4].

The key insight and takeaway from the project is that there is no reason reinforcement learning algorithms cannot learn coverage path planning and obstacles avoidance as independent tasks. However, the main challenge is combining both objective in one agent, and this becomes exponentially more difficult as the agent might be confused of understand and acknowledging both objectives at the same time.

2 Previous Work

2.1 Graphical Based Coverage Path Planning

In 2011, Marija Dakulovic, Sanja Horvatic, and Ivan Petrovic proposed a coverage D* algoithm for path planning of a floor-cleaning mobile robot. Following, in 2014, Zengyu Cai, Shuxia Li, Yong Gan, Ran Zhang and Qikun Zhang also proposed a coverage planning algorithm for cleaning robot. Both graphical techique showcase the ability to conduct coverage on deterministic path planning static obstacles with high coverage rate and low repetition rate using D* and A* search [2], [3].

2.2 Reinforcement Learning Based Coverage Path Planning

In 2020, researchers start to investigate applying RL to coverage planning problems. Dennis and Theile in their respective publication [5], [6] presented solving coverage path planning using DQN and DDQN. Their results have concluded that the algorithms is capable of conducting 95% of coverage in a limited time steps given to the agent. However, it is important to note that both paper assume obstacles are static. Hence, like previously mentioned, the problem becomes much more challenging with dynamic moving obstacles if the agent needs to avoid them while conducting coverage.

3 Experiment Setup

3.1 Environment

The MiniGrid [7] Gym Environment was used as a base, and the farmworld environment explored by the agent is built off of the Dynamic Obstacles environment featured in this repository. Our experiments were conducted on a 16x16 grid and a 24x24 grid. The environment also encounters blue obstacles, randomly spawning outside the environment, then moving rightwards to cross the world. These obstacles simulate clouds.

3.2 Action States, Observation States, Reward Function

The action states are {up, down, right} movement for the agent as it requires the agent to make a 180 degree turn on the same tile if it decides to change direction.

The observation states are {7x7 vision of the agent (which include the agent's position, obstacles within the vision, a queue of previous five iterations of the location of the obstacles, and a 2D array that represents the visited tile of the entire grid represented by 0s (not visited) and 1s (visited)}.

The formulation of our reward function we started off with a simple strategy:
= +3 new region, -10 encountering obstacle, -0.5 revisiting covered region.
However, we have investigated and experimented with varies values and decided with
= +3 new region, -20 encountering obstacle, 0 revisiting covered region.

3.3 Vision Coverage

In the vision coverage, the agent's observations are a 7x7 box directly in front of it, representing a drone's forward facing camera. The reward function gives a negative reward for collisions with obstacles, or if the agent re-sees previously seen tiles. Otherwise, the agent gains a reward for each newly seen tile. A discount factor is added so the agent does not ignore future rewards. A 'step' in this environment increases the timecount by 1, and includes moving forward, turning left, or turning right. Whenever the agent collides with an obstacle, the reward is defaulted to a large negative value (-20). This is done to significantly penalize obstacle collisions.

3.4 Training and Model Setup

The training setup involves 16 environments training in parallel, each running for 80,000 frames. This distributed training improves training efficiency. We also experimented with soft actor critic [8] in our a2c algorithm. Generalized Advantage Estimator [9] was incorporated into our model. This allowed us to make tuning adjustments on the bias-variance trade-off. Coverage is unique among RL problems where novel states almost always equate to desirable states - unless there is an obstacle in the way, visiting a new state is the desirable. In this vein, we made use of information entropy, and utilized the soft actor critic algorithm to improve training. This would hypothetically improve both training and performance. Aside from this, training is optimized with the use of Long Short-Term Memory (LSTM) Networks. Such networks contain feedback connections, allowing it to process time series data. They have been shown to be effective in acoustic modelling, outperforming standard DNNs and RNNs [10]. The goal in adding recurrence to our model was that the agent would learn obstacle pattern movements (that obstacles consistently move rightwards), and therefore learn to better dodge them.

Algorithm 1 A2C

```
 1: procedure A2C
 2:     Start with policy model $\pi_\theta$ and value model $V_\omega$
 3:     repeat:
 4:         Generate an episode $S_0, A_0, r_0, \ldots, S_{T-1}, A_{T-1}, r_{T-1}$ following $\pi_\theta(\cdot)$
 5:         for $t$ from $T-1$ to 0:
 6:             $V_{end} = 0 if (t + N \geq T)$ else $V_\omega(s_{t+N})$
 7:             $R_t = \gamma^N V_{end} + \sum_{k=0}^{N-1} \gamma^k \left(r_{t+k}\right.$ if $(t + k < T)$ else $\left.0\right)$
 8:         $L(\theta) = \frac{1}{T} \sum_{i=0}^{T-1} (R_t - V_\omega(S_t)) \log \pi_\theta(A_t|S_t)$
 9:         $L(\omega) = \frac{1}{T} \sum_{i=0}^{T-1} (R_t - V_\omega(S_t))^2$
10:         Optimize $\pi_\theta$ using $\nabla L(\theta)$
11:         Optimize $V_\omega$ using $\nabla L(\omega)$
12: end procedure
```

The image above represents where we are adding the LSTM layer.

3.5 Metrics

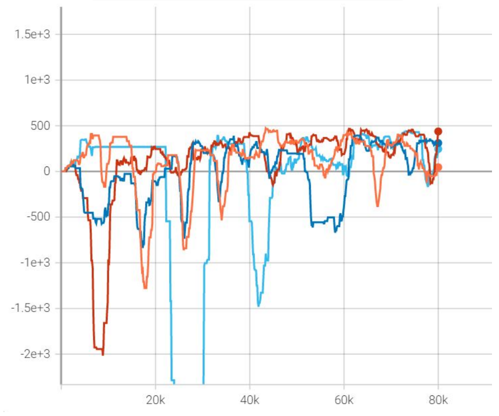
The performance of the agent was measured with its returns and timecounts to completion. The return is simply the total rewards accumulated by the agent until it covers the entire grid. The timecount is the number of steps the agent takes to cover the entire grid. These two variables serve as good performance metrics for the task - measuring how quickly the agent covers the grid and how efficient the agent is at coverage while also dodging obstacles.

4 Results

Below are the average time counts to completion and average returns per episode over the training run.

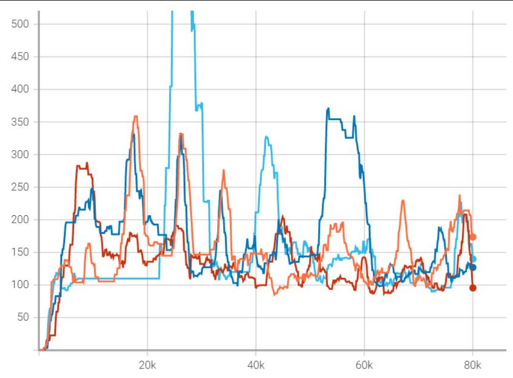
4.1 Coverage Results for varying Entropy Coefficients

Average Returns per Episode



Orange: E=0, Dark Blue: E=0.01, Red: E=0.1, Light Blue: E=1

Average Time-Counts to Completion per Episode

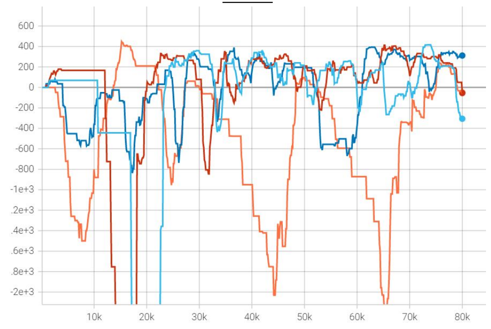


Orange: E=0, Dark Blue: E=0.01, Red: E=0.1, Light Blue: E=1

4.2 Coverage Results for varying GAE-λ Coefficients

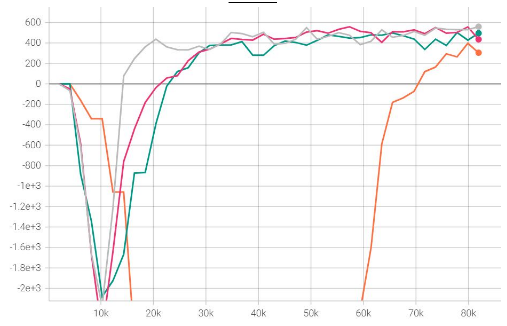
Average Returns per Episode

A2C



Light Blue: $\lambda = 0.5$, Red: $\lambda = 0.85$, Dark Blue: $\lambda = 0.95$, Orange: $\lambda = 1$

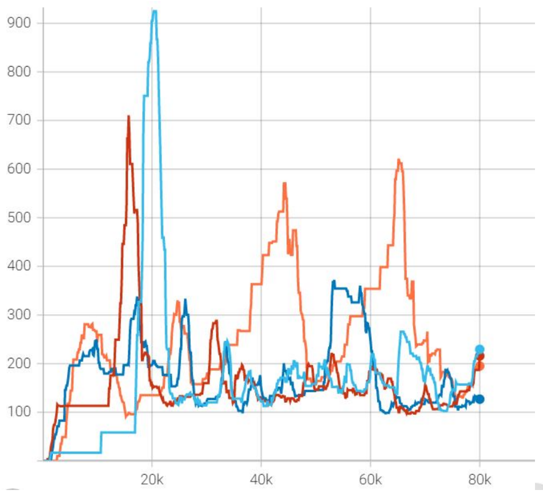
PPO



Grey: $\lambda = 0.5$, Purple: $\lambda = 0.85$, Turquoise: $\lambda = 0.95$, Orange: $\lambda = 1$

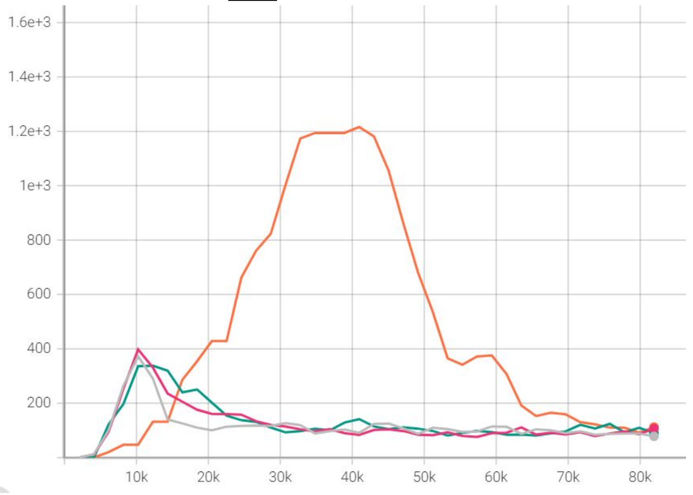
Average Time-Counts to Completion per Episode

A2C



Light Blue: $\lambda = 0.5$, Red: $\lambda = 0.85$, Dark Blue: $\lambda = 0.95$, Orange: $\lambda = 1$

PPO

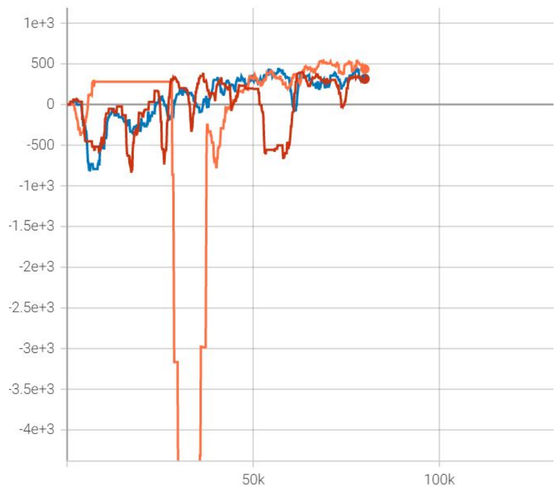


Grey: $\lambda = 0.5$, Purple: $\lambda = 0.85$, Turquoise: $\lambda = 0.95$, Orange: $\lambda = 1$

4.3 Coverage Results for varying Recurrence values

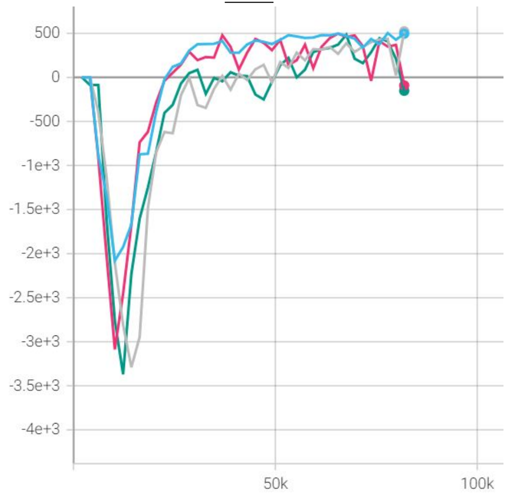
Average Returns per Episode

A2C



Red: R=1, Orange: R=4, Blue: R=8

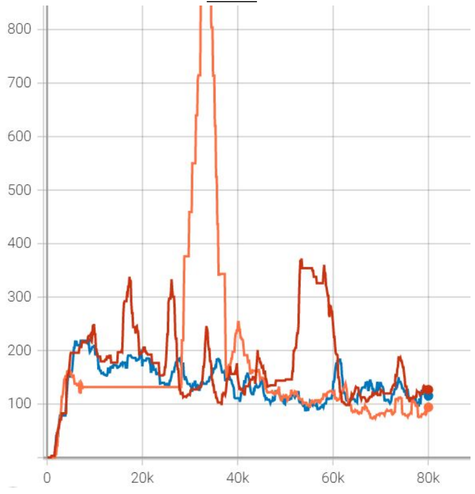
PPO



Light Blue: R=1, Purple: R=4, Green: R=8, Grey: R=16

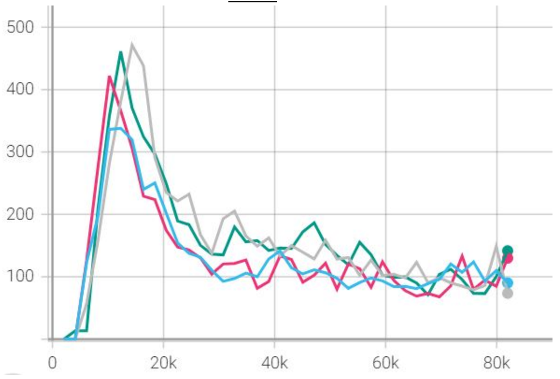
Average Time-Counts to Completion per Episode

A2C



Red: R=1, Orange: R=4, Blue: R=8

PPO



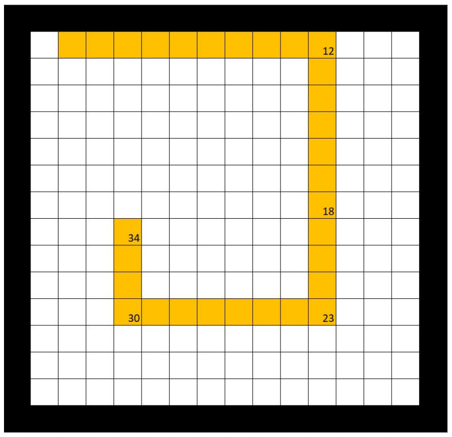
Light Blue: R=1, Purple: R=4, Green: R=8, Grey: R=16

5 Evaluation

The results were not as good as expected. Furthermore, when testing on a completely empty grid, devoid of obstacles, the agent did not perform the optimal action. This performance on a deterministic, non-stochastic environment was not favourable. Additionally, the parameters explored above (GAE-λ, Entropy Coefficients Soft Actor Critic, and LSTM) did not have a significant effect on performance with the tested values. Both PPO and A2C displayed very similar effectiveness on both of the performance metrics. Both algorithms were able to cover the grid completely in around 100 timecounts by the end of training, and both received average rewards of around 500 by the end of training.

However, as shown above, the agents do improve over time.

Unfortunately, due to the novelty of both the approach and the problem itself (the Gym Environment used in our experiment was custom built), no established benchmarks exist with which to compare the results.



However, the environment is cover-able in 34 steps (including turning) without obstacles. This path is shown in the figure above. From our results, the agent is capable of achieving finishing timecounts of around 100, which is relatively successful given the challenging nature of coverage problems.

6 Conclusion

In conclusion, we adopted multiple RL techniques to PPO and A2C algorithms and tuned multiple reward functions to train on two different coverage control problems. Our experiment saw relatively strong results on the two performance metrics we tracked - average reward and time counts to completion. Considering the current results, it is important to understand the challenges that is being faced by our RL agent: attempting to learn two task within one objection function. This problem is inherently challenging for the agent to understand and learn.

7 Broader Impact and Future Work

This research has wide applications in drone coverage problems, and therefore will likely be useful in the agricultural industry. Not much work has been done in the intersection between coverage control and Deep RL, and this paper serves as a good introductory dive into potential solutions and approaches in this area.

As previously mentioned in the extended abstract, the next step will be combining current work in adversarial environment such as PAIRED [4]. From there on, we will enter the final stage (in extended abstract) of this research project.

8 Code

```
https://github.com/carlchua/cs285_rl_files
https://github.com/carlchua/cs285_env.git
```

Please contact for access.

9 References

[1] Srivastava, Nitish, Elman Mansimov, and Ruslan Salakhudinov. "Unsupervised learning of video representations using lstms." International conference on machine learning. PMLR, 2015.

[2] Dakulović, Marija, Sanja Horvatić, and Ivan Petrović. "Complete coverage D* algorithm for path planning of a floor-cleaning mobile robot." IFAC Proceedings Volumes 44.1 (2011): 5950-5955.

[3] Cai, Zengyu, et al. "Research on complete coverage path planning algorithms based on A* algorithms." The Open Cybernetics Systemics Journal 8.1 (2014).

[4] Dennis, Michael, et al. "Emergent complexity and zero-shot transfer via unsupervised environment design." arXiv preprint arXiv:2012.02096 (2020).

[5] Theile, Mirco, et al. "UAV coverage path planning under varying power constraints using deep reinforcement learning." 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020.

[6] Zhang, Charles. "Area Coverage with Unmanned Aerial Vehicles Using Reinforcement Learning." (2020).

[7] M. Chevalier-Boisvert, L. Willems, and S. Pal. Minimalistic Gridworld Environment for OpenAI Gym, 2018. URL `https://github.com/maximecb/gym-minigrid`

[8] Levine et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor" ICML 2018, 2018. `https://arxiv.org/pdf/1801.01290.pdf`

[9] Schulman, Moritz et al. "High Dimension Continuous Control Using Generalized Advantage Estimation" ICLR 2016, 2016, `https://arxiv.org/pdf/1506.02438.pdf`

[10] Sak, Haşim, et al. “Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling.” Interspeech 2014, 2014, doi:10.21437/interspeech.2014-80.