Autonomous Drone Navigation for First Response

Victor Greenberg
Stanford University
vgreenbe@stanford.edu

Carlos Hernandez
Stanford University
charlsdh@stanford.edu

Extended Abstract

Despite the widespread adoption of unmanned aerial systems (UAS), modern drone operations remained limited by operator input and pre-scripted automation—such limitations required novel approaches to achieve full automation, the next milestone of drone technology. Our project explores how reinforcement learning (RL) can enable fully autonomous UAV teams to rapidly deploy in response to emergencies in urban environments. Our approach is rooted in autonomously navigating to target locations while avoiding obstacles, and coordinating timing and arrival. Our exploration serves as the first step toward a Hierarchical RL-driven Drone First Response (DFR) system that stands capable of full mission autonomy: deployment, pursuit, surveillance, and evidence capture. We began our research by employing both LiDAR sensor and simplified "nearest obstacle" observations to navigate between fixed locations. Although we found some success in pre-defined point-to-point navigation, our attempts towards generalized learning remained largely unsuccessful despite the use of expert examples; initial tests with curriculum learning (CL) appeared impractical but a subsequent alternative approach in CL proved useful. We further demonstrated the value of 3D visualization using Unreal Engine and AirSim, and discovered interesting personality differences between PPO and A2C and the effects of longer training timelines.

Main Findings:

- PPO worked well, achieving 93% success rates with fixed but challenging start and end locations.
- Reasonable (80%) success for navigation from a fixed location to an arbitrary one was partially achieved after significantly increasing the n_steps hyperparameter, longer training and a particular type of curriculum learning.
- Alternative algorithms (SAC, DDPG, TD3) performed very poorly, possibly due to suboptimal choice of hyperparameters.
- A2C achieved a lower success rate than PPO but produced higher quality trajectories sooner.
- Attempts at pre-training with "expert" trajectories using BC degraded performance (this included both manually specified and artificial potential field induced examples).
- Initial attempts at *curriculum learning* did not enhance generalization, however a different approach (starting with a simpler reward function) led to decent results (80% success rate)
- LiDAR with longer range 100m performed better than 12m but still (surprisingly) underperformed the "direction and distance to nearest obstacle" unrealistic observation.
- The LiDAR resolution (measurement every 15 degrees) performed as well as much higher resolution

Main Contributions:

- Use of simulation environments of different / appropriate complexity and realism, including a fully photo-realistic Unreal Engine world with AirSim. Unlike in reference (a) *we use a simple and computationally cheap simulation environment and when necessary test and fine-tune the models in AirSim.*
- Exploration of using manual as well as potential-field generated trajectories as suggested in reference [2]. Our results showed hazards in pre-training with expert examples, which can often lead to degradation in performance. This is likely due to differences in probability distributions between different examples and between examples and the learned policy.
- Validating reality transferability from the get go by testing on industry standard software-in-the-loop ArduPilot simulator and target edge hardware.

Autonomous Drone Navigation for First Response

Victor Greenberg
Stanford University
vgreenbe@stanford.edu

Carlos Hernandez
Stanford University
charlsdh@stanford.edu

Abstract

Despite the widespread adoption of unmanned aerial systems (UAS), modern drone operations remained limited by operator input and pre-scripted automation—such limitations required novel approaches to achieve full automation, the next milestone of drone technology. Our project explores how reinforcement learning (RL) can enable fully autonomous UAV teams to rapidly deploy in response to emergencies in urban environments. Our approach is rooted in autonomously navigating to target locations while avoiding obstacles, and coordinating timing and arrival. Our exploration serves as the first step toward a Hierarchical RL-driven Drone First Response (DFR) system that stands capable of full mission autonomy: deployment, pursuit, surveillance, and evidence capture. We began our research by employing both LiDAR sensor and simplified "nearest obstacle" observations to navigate between fixed locations. Although we found some success in pre-defined point-to-point navigation, our attempts towards generalized learning remained largely unsuccessful despite the use of expert examples; initial tests with curriculum learning (CL) appeared impractical but a subsequent alternative approach in CL proved useful. We further demonstrated the value of 3D visualization using Unreal Engine and AirSim, and discovered interesting personality differences between PPO and A2C and the effects of longer training timelines.

1. Introduction

The United States continues to make significant investments in UAS across multiple government sectors, reflecting the growing importance of drones in national defense, public safety, and law enforcement. This includes $10.95 billion budgeted in 2024 by the DoD and $91.51 billion by DHS. State and local law enforcement agencies have followed suit, adopting UAS technology for emergency response and surveillance, among other applications.

By enabling drones to operate independently in dynamic, high-risk environments, this approach reduces critical response time, minimizes danger to human personnel, and ensures faster aid to civilians—ultimately saving lives when every second matters.

This CS224R project lays the foundation for more ambitious and immediate follow-on initiatives that will apply RL to training a drone swarm capable of performing a range of drone operations with increasing complexity.

Our initial efforts strive to develop a drone as a first responder solution (Autonomous DFR) that enables a team of drones to collaboratively deploy in response to reported a burglary while safely and quickly navigating to the reported address. The key actions include: timing, approach and arrival coordination; halting robbery via interdiction; tracking retreating suspects; and video surveillance suitable for follow-on prosecution. In case of multiple suspects, the drones will split up and independently pursue while maintaining visual custody.

As one of the main building blocks, the extended project will explore ways to achieve reliable collaborative navigation between two arbitrary locations in urban environments. Collaboration will be needed in order to avoid collisions and coordinate approach directions. For the current phase of the project, however, we focus on the first step of safely navigating to the destination.

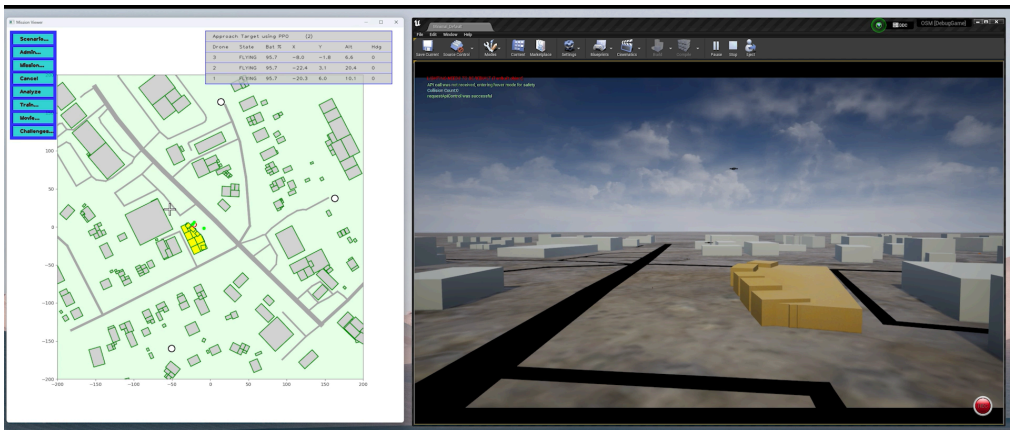
2. Key Components

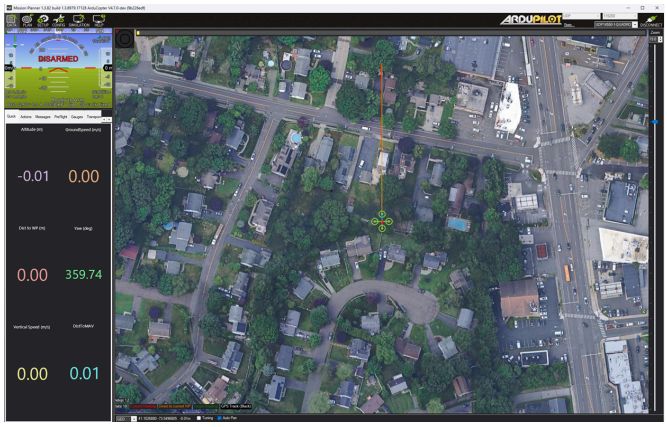
To avoid the difficulties often faced with reality transfer, we chose to build the project in a way that continuously validates the UAVs behavior using industry-standard simulators (ArduPilot SITL), with primary visualization using Unreal Engine with AirSim for visually and physically realistic simulation. Although not directly used in this project except to validate performance on edge hardware, a Pixhawk / Raspberry Pi powered drone stands as one of the intended agents to adopt our solution.

For efficiency of training, we use a simplified geometry-based environment with building data downloaded from OpenStreetMap. We use PPO with simulated sensor inputs that provide the agent's position (GPS + Barometer) and direction, and distances from obstacles (i.e. LiDAR). We further use peer-to-peer data distribution service (DDS) communication between drones to provide respective positions in support of object detection information to help the drones avoid crashing into each other and coordinate their arrival time (which we want to be concurrent).

Below (top left) is a 2D visualization built using OpenStreetMap data and OpenCV. On the right is the same scene concurrently visualized in 3D in an Unreal Engine world (also generated from OpenStreetMap) and using Microsoft AirSim to simulate the drones. Among other things, the 3D world provides accurate physics (collisions, gravity, weather and lighting) and simulated cameras enabling virtual object recognition and (potentially) a high-dimensional visual observation space.

In the bottom left we see the same scene as seen in industry-standard MissionPlanner and ArduPilot, a parallel environment used to validate drone behavior using the MavLink standard protocol. Bottom right is our physical drone for testing on edge hardware (Pixhawk with Raspberry Pi) - see Appendix 4.





For the software stack we used the following packages:

1. PyTorch, Numpy (base platform)
2. Farama Gymnasium (for environment interface and SB3 integration)
3. Stable Baseline 3 (for RL algo implementations)
4. RTI Connext DDS (for peer to peer communication between drones and admin UI)
5. Shapely (for geometry calculations)
6. OSMNX (for OpenStreetMap data)
7. Geopy (for Geography transformations)
8. Dronekit (for MavLink API)
9. Ultralytics (for image detection in AirSim)
10. Airsim (for simulated drones)
11. Imitation (for behavior cloning)
12. OpenCV (for admin UI)

3. Related Work

- Commercial
  - Skydio DFR solutions
    - Skydio allegedly uses RL to train models that process real-time sensor data to make decisions in complex environments, however the exact details are not publicly available.
  - Shield AI
    - This is one of the leaders in the application of autonomy and AI to defense and is definitely using RL for drone missions. The details are not publicly available.
  - Flock Safety
    - Flock Safety specializes in public safety technology, focusing on automated license plate recognition, video surveillance, and gunshot detection systems. Their products use machine learning and computer vision to capture and analyze objective evidence, such as vehicle data, to aid in crime prevention and investigation. There is no indication of them using RL.
  - Brinc
    - Brink focuses on creating drones and tools to assist first responders, law enforcement, and public safety agencies. Their solutions integrate features like automated license plate recognition, real-time video feeds, and one-click drone deployment for rapid

response. Their ecosystem is designed to enhance situational awareness, de-escalate dangerous situations, and save lives. There is no indication of them using RL.

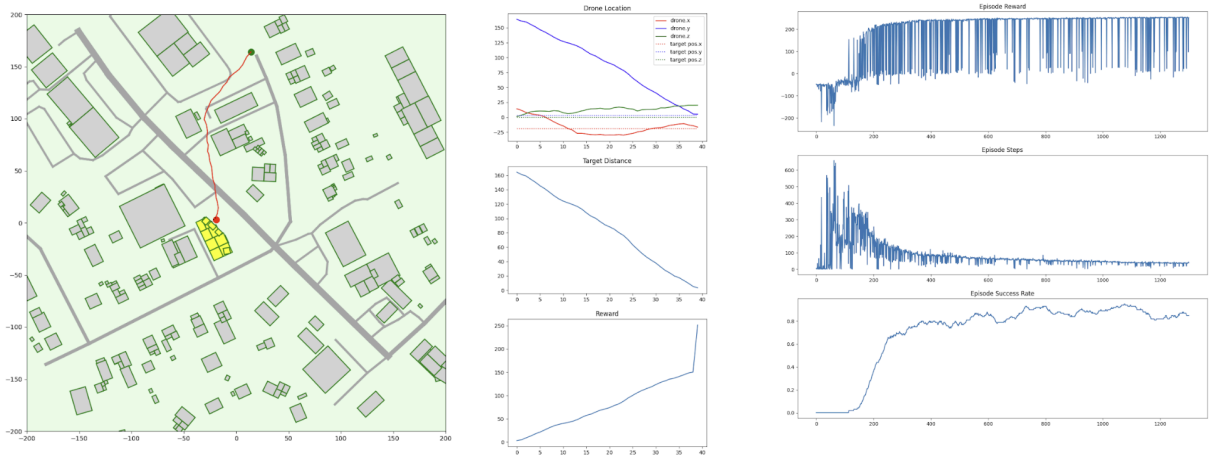
- Academic
  - [1] Parallel Reinforcement Learning Simulation for Visual Quadrotor Navigation
  - [2] Potential Fields Guided Deep Reinforcement Learning for Optimal Path Planning in a Warehouse
  - [3] Autonomous Unmanned Aerial Vehicle navigation using Reinforcement Learning: A systematic review

- Our main innovations
    - Use of simulation environments of different / appropriate complexity and realism, including a fully photo-realistic Unreal Engine world with AirSim. Unlike in reference [1] we use a simple and computationally cheap simulation environment and when necessary test and fine-tune the models in AirSim.
    - Exploration of using manual as well as potential-field generated trajectories as suggested in reference [2]. Our results showed hazards in pre-training with expert examples, which can often lead to degradation in performance. This is likely due to differences in probability distributions between different examples and between examples and the learned policy.
    - Validating reality transferability from the get go by testing on industry standard software-in-the-loop ArduPilot simulator and target edge hardware.

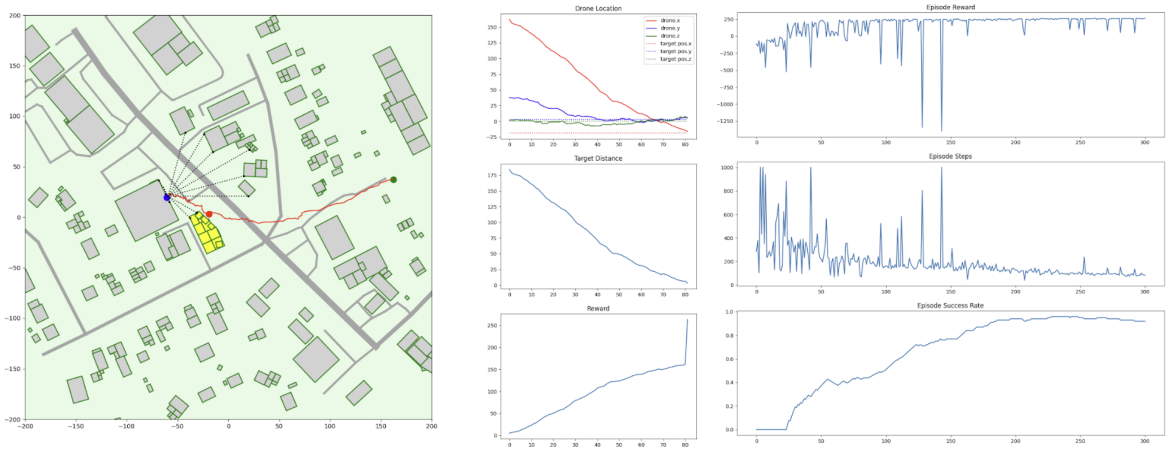
4. Results

4.1 Decent results for fixed start and end location

Here are the results for Dock 1 to the fixed burglary location, achieving an 85% success rate using the nearest obstacle observation.

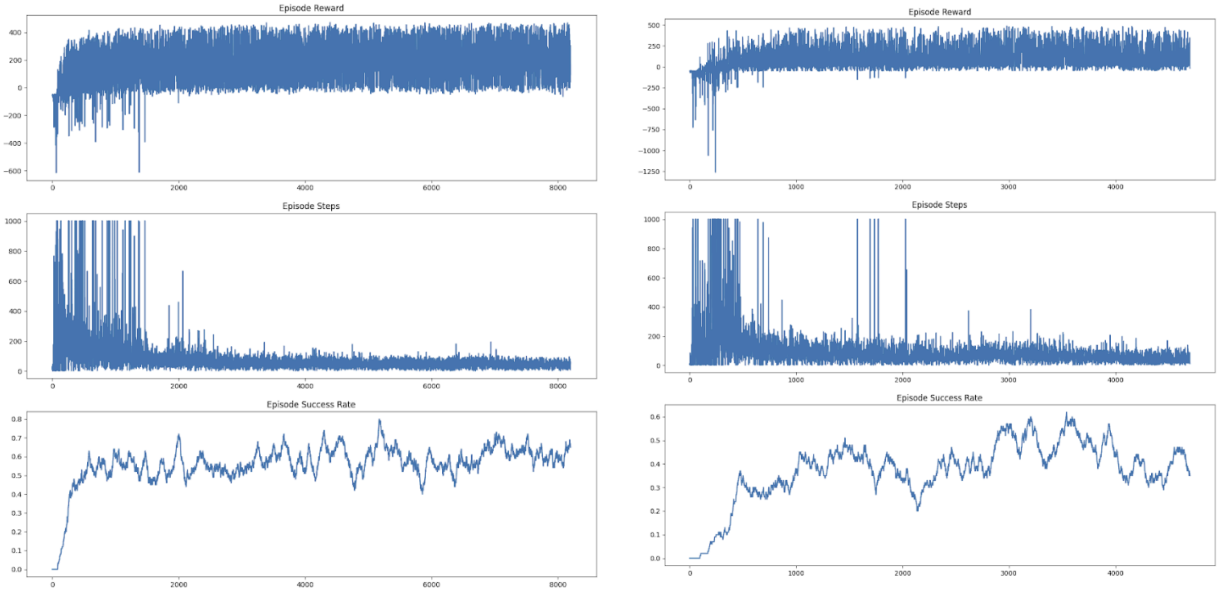


Similar results with (simulated) LiDAR based observations, here for Dock 2**:**

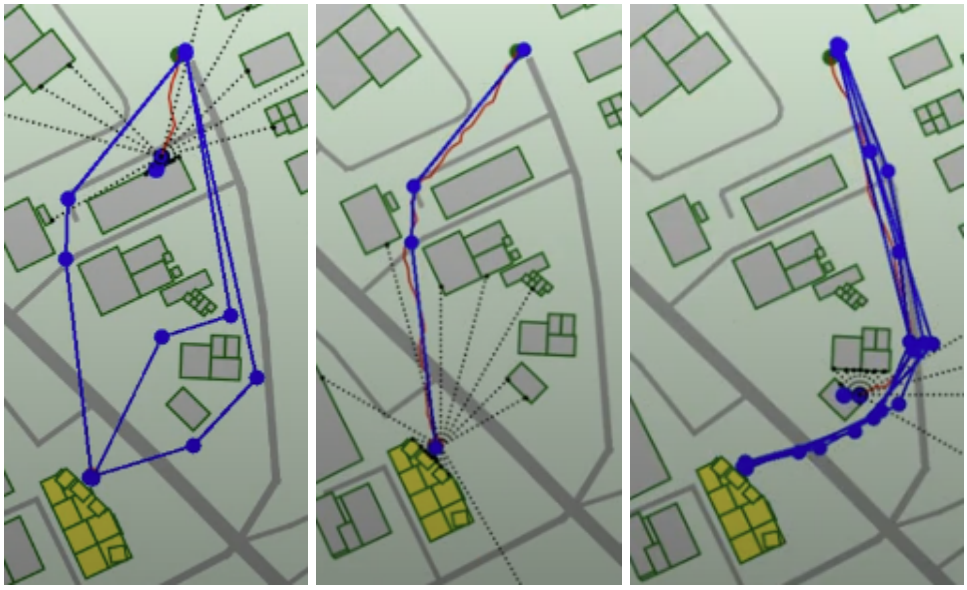


4.2 First attempts at Generalization

When training to navigate from dock 1 (left) or 2 (right) the results are not so good, achieving only 40-60% success rates.

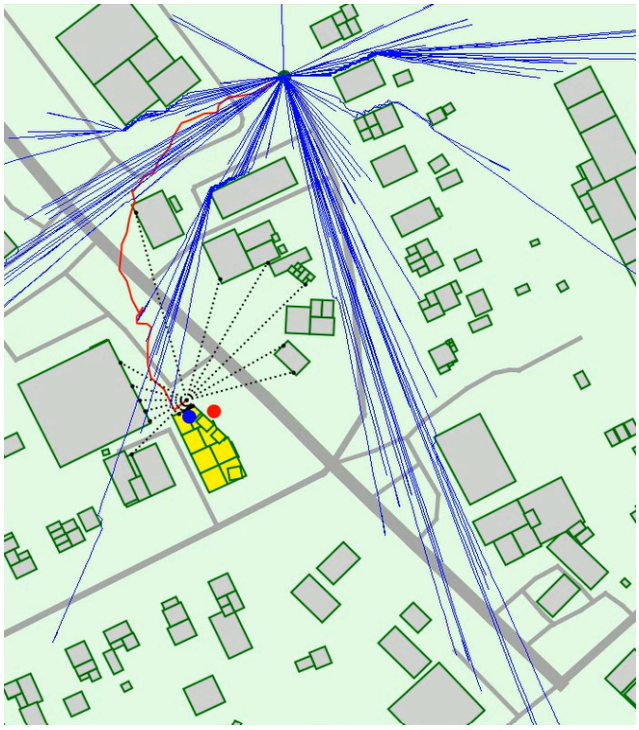


Pretraining PPO with Expert Examples (manual and artificial potential field based)



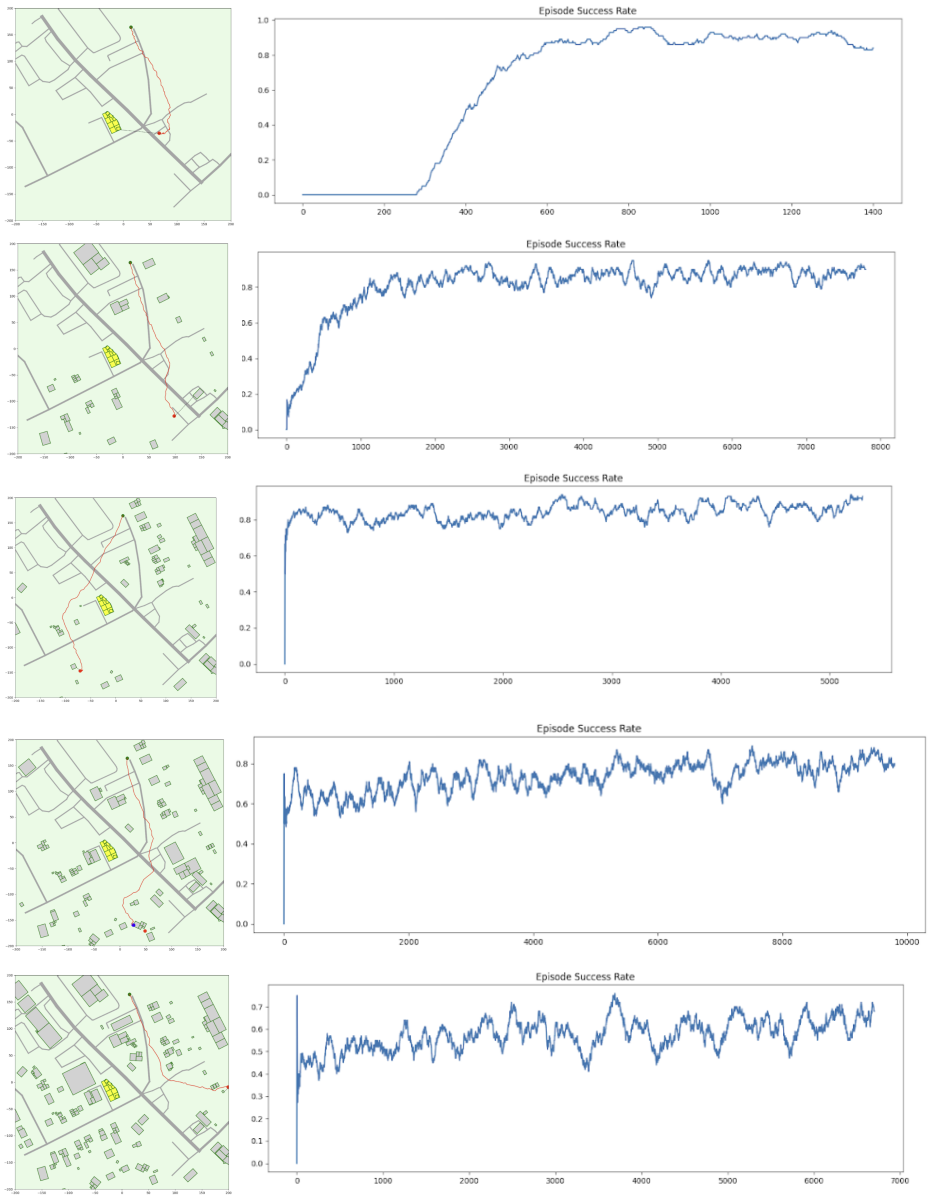
With three examples from apparently different distributions (left) the success rate collapsed to zero. It seems to be attempting some sort of average of the three, which is always unsuccessful. With a single example it quickly converged to a good solution with 91% success rate. With multiple similar examples again it seems to have gotten confused with a complete collapse in success rate. We suspect the issue is that what appears to the eye as similar is actually not that similar in the observation space.

A similar collapse to zero occurs with 100 artificial potential field generated examples.



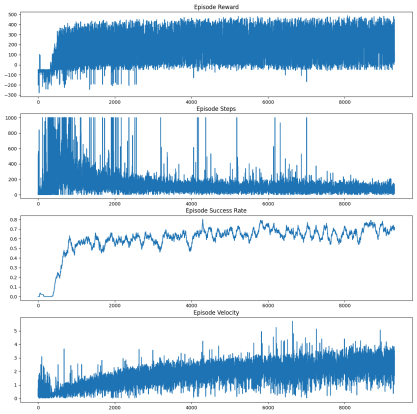
4.3 Curriculum Learning

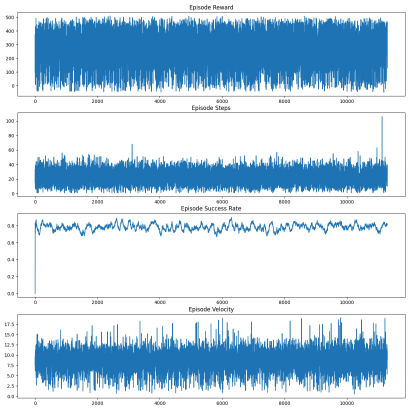
Our first attempt to use curriculum learning to crack the generalization problem attempted to simplify the environment by stepwise removing 100, 80, 60, 40 and 20% of the buildings. Unfortunately while the results are informative and demonstrate that curriculum learning works, the results obtained were no better than with the direct approach.



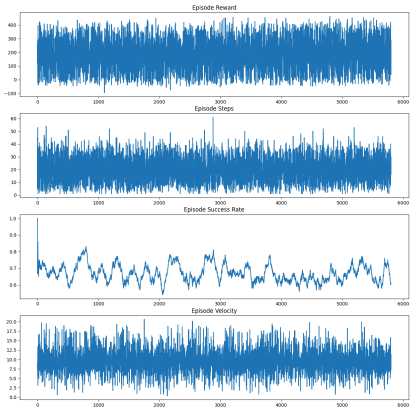
The inspiration for a solution eventually came from the a different experiment (see *Altitude Penalty Impact* below), in which we were able to obtain a good success rate for Dock 3 only by first pre-training *without* the altitude penalty (i.e. allowing the drones to fly as high as they wish), then fine-tuning with the altitude penalty added. Applying this technique to the generalized navigation problem we finally seem to have a break-through, achieving **80%** success rates.

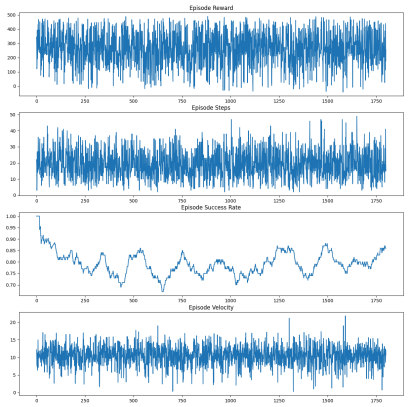
Without altitude penalty (initial on left, final on right after 82,000 episodes):





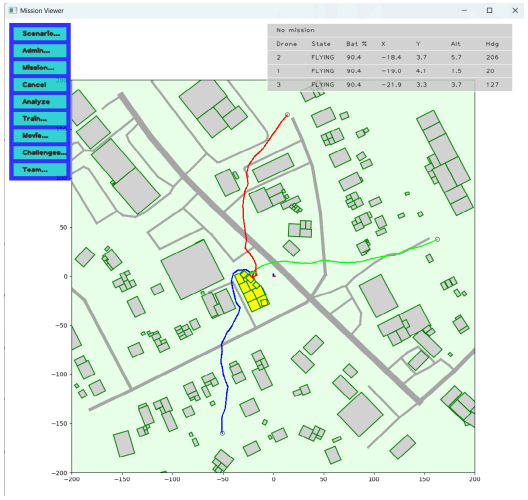
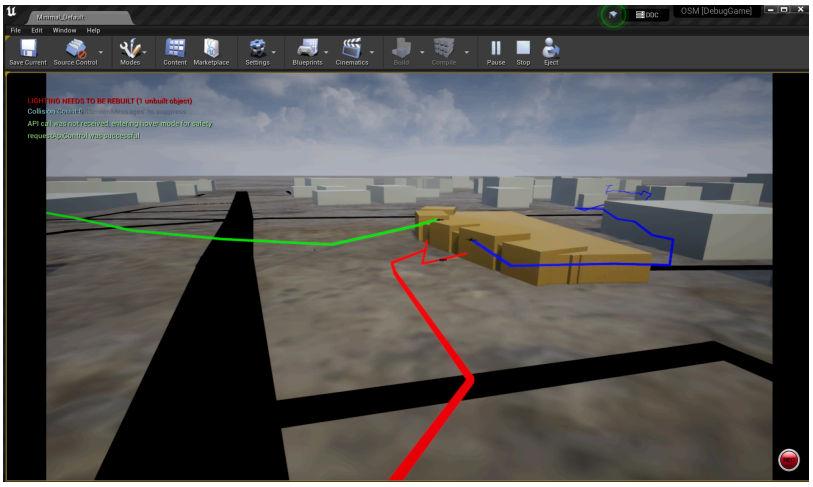
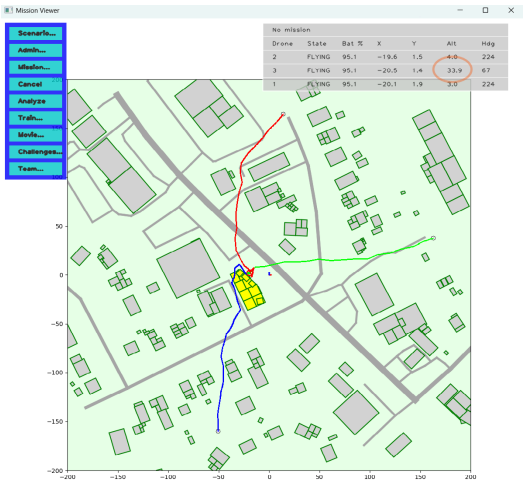
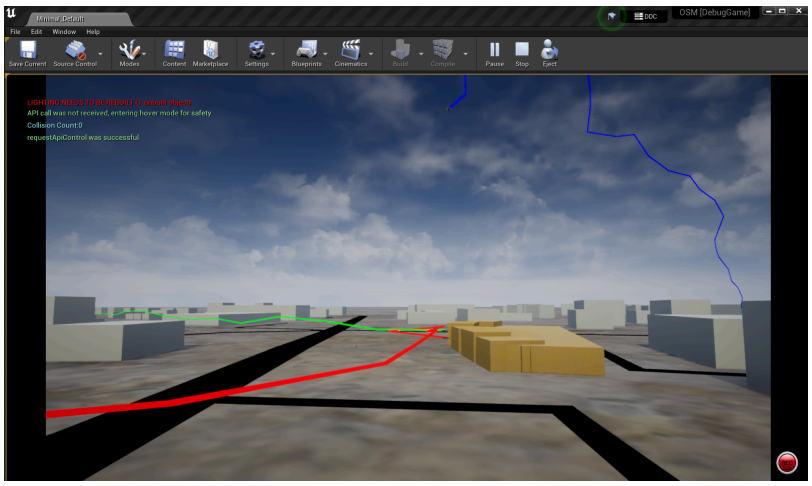
After adding a 0.3 per meter altitude penalty and continuing to train for another 82,000 episodes we finally get somewhere. Here are the initial (left) and final results:





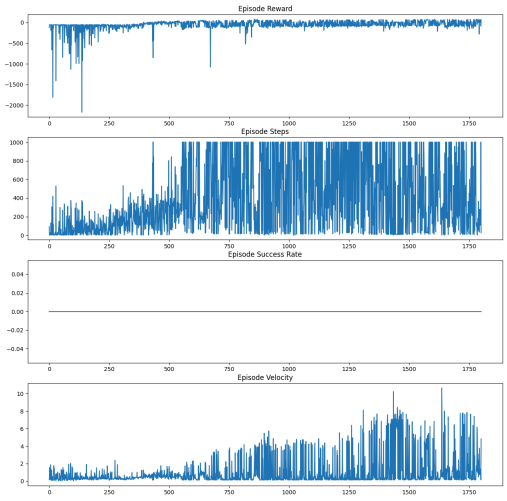
4.4 Altitude Penalty Impact

Here is a visual comparison of the 2D and 3D trajectory with an altitude penalty of 0.3 per meter (top) and without it for drone 3 (bottom, blue trace line for dock 3). (The red and green lines, for dock 1 and 2 are with the altitude penalty in both).

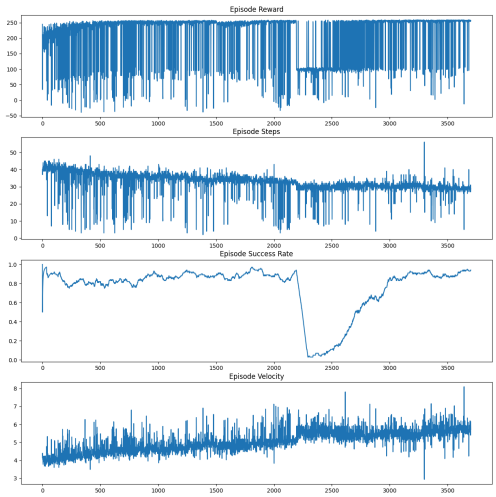





Incidentally, this clearly illustrates the value of a 3D visualization -- the two drastically different behaviours look identical in 2D!

Clearly we need the altitude penalty in order to get good trajectories. However, adding an altitude penalty of 0.3 / meter to the reward function surprisingly resulted in a collapse of learning for docking station number 3.

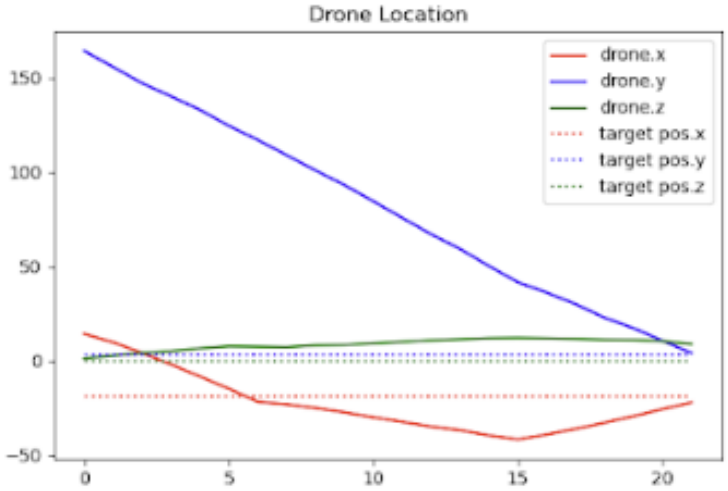


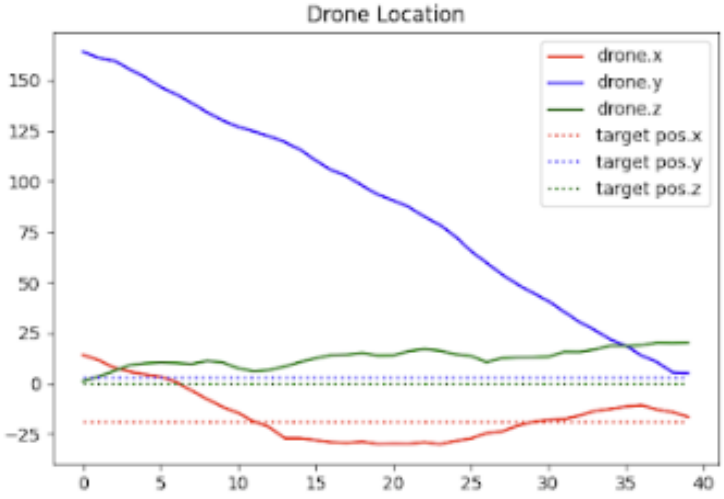
Since it was previously achieving success rates in the 90's before we added an altitude penalty (at the expense of flying too high) we attempt to use a different approach to curriculum learning by pretraining on the altitude-unconstrained reward model, then fine-tuning with the penalty re-introduced The result is below (with a curious temporary dip in success rate, quickly recovered). The 90% success rate is again achieved!



4.5 Qualitative differences between PPO and A2C

Although the success rate we were able to achieve with A2C (64%) was much lower than with PPO (85%), the quality of the resulting trajectory is clearly preferable (top = A2C, bottom = PPO)



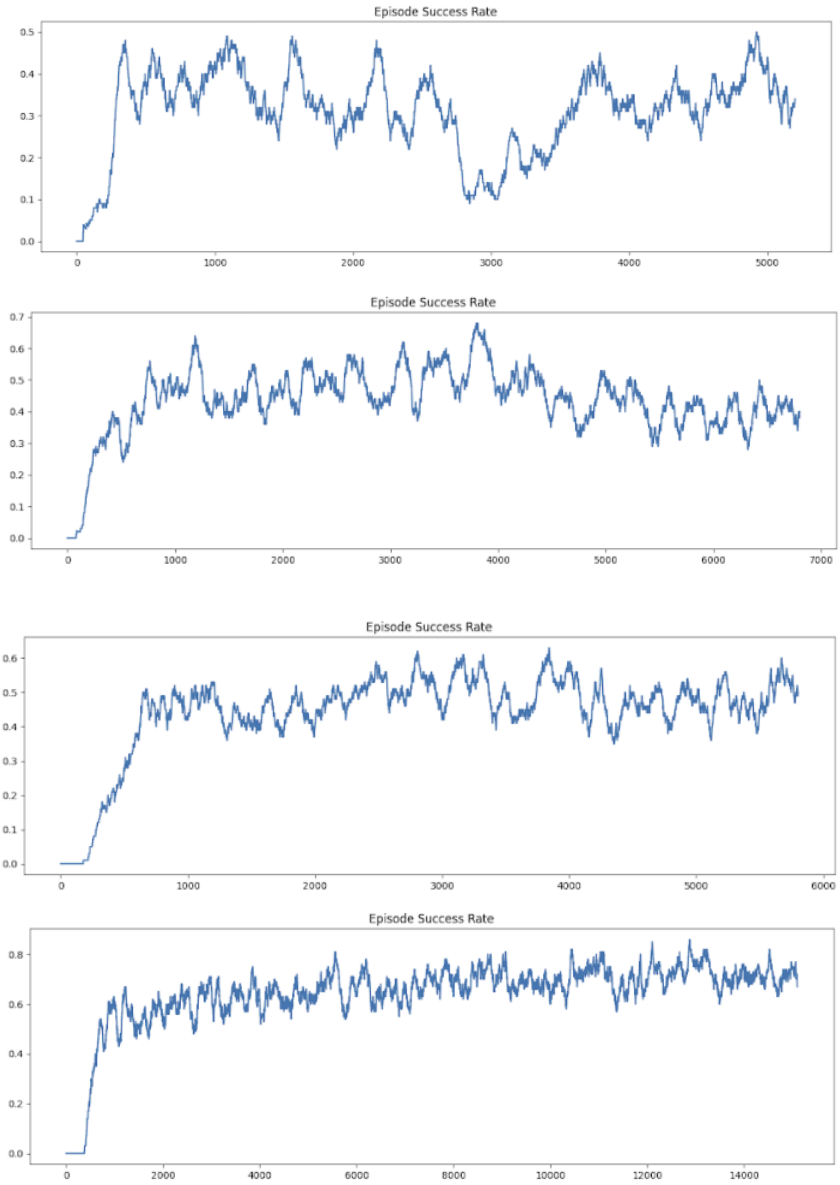


Note that the A2C trajectories are smooth and resemble what you would expect to see from an expert, while the PPO ones are slightly erratic with a very large number of unnecessary adjustments. This relative velocity (6 m/s) was also much better than with PPO (4 m/s).

Another interesting observation when comparing PPO to A2C is that A2C was very “decisive” and quickly started with large velocities, toning them down somewhat as training progressed, while PPO starts extremely timidly, with velocities of about 0.1 m/s (vs. about 100x for A2C). We suspect that the smoother trajectories achieved by A2C are a result of this more aggressive behavior. As discovered later (and discussed below) training much longer led to the PPO trajectories eventually matching the A2C ones in smoothness and speed.

4.6 Steps per update (n_steps) Hyper Parameter

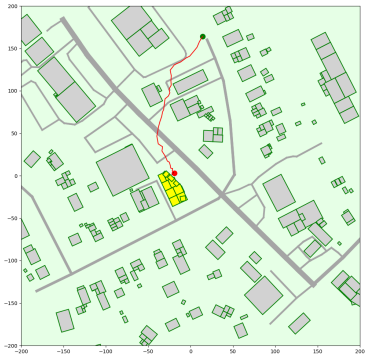
Impact of n_steps hyperparameter (1024, 2048, 4096, 16384) for Dock 1 to a random location

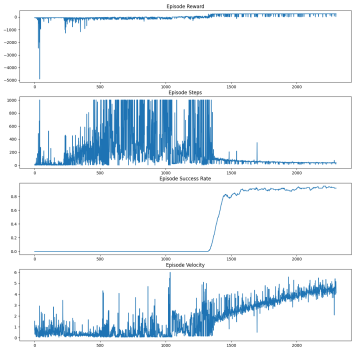


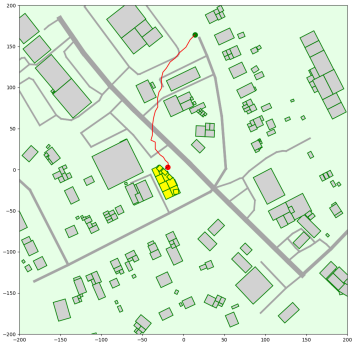
Increasing n_steps from its default of 2048 to 4096 and finally 16384 resulted in a much more stable learning and better success rates (34% to 50% to 67%).

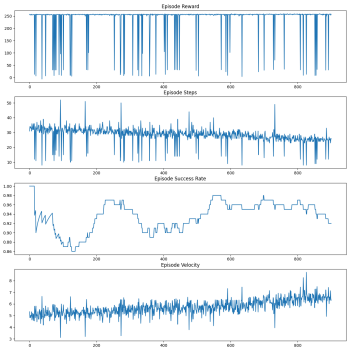
4.7 Longer Training

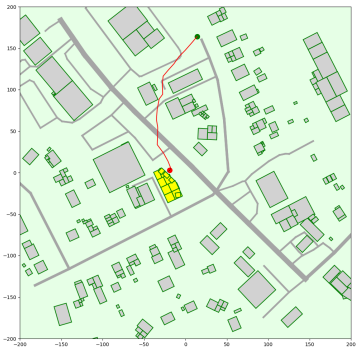
To see if we can get PPO to match the trajectory quality achieved by A2C we tried to train for more episodes. Previously we would stop when achieving a sufficiently high success rate. It turns out that an additional important stopping criterion to use is the average velocity achieved -- we want the drones to arrive at the destination as quickly as possible, and higher velocity tends to correspond to fewer random direction changes, i.e. smoother trajectories. As we continued training the velocity increased from 4 to 6 to 9 m/s and the trajectory started looking like the A2C one, but with a much higher success rate of 97% vs 64%.

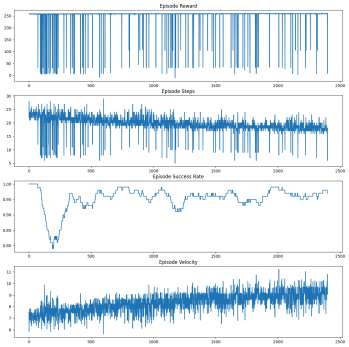












4.8 Reward Shaping

The two reward functions we attempted are shown in Appendix 3.

The second reward function we attempted used exp (- distance_to_target * 0.05) * 10 instead of rewarding getting closer and penalizing getting further from the destination. The theoretical justification for this was that we want to avoid relying on information from the previous state. The coefficients chosen to produce reasonably high gradients throughout the trajectory with a maximum of +9 for 2m (target distance for success being 2.5m).

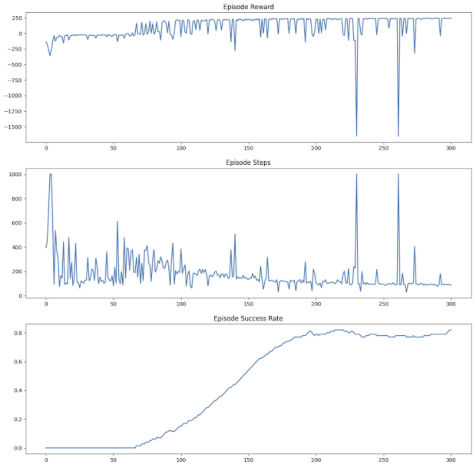
While overall success rate with this reward function was slightly lower the learning progress appeared to be much more fixated on high reward states (locations closer to the target).

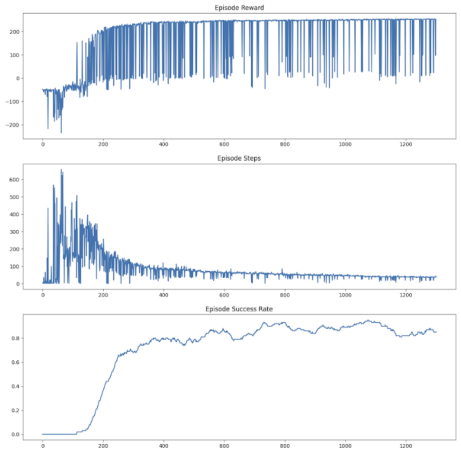
With V1 it seemed to explore more varied trajectories initially, perhaps because it was rewarded for **moving** closer rather than **being** closer to the target. We suspect this is behind the quicker progress but lower success rate with V2. An alternative explanation is that the trajectory it seems to find with V1 (far right) is less risky or easier to fine tune. More investigation is needed.

4.9 Observation Space variations

Training with the LiDAR (left) produced a more steady learning pattern. One guess as to the explanation is that the inputs are a lot more continuous, whereas with the nearest obstacle approach only, the identity of the obstacle jumps discontinuously.

Further investigation is needed.





5. Conclusions and Key Findings

- PPO worked well, achieving 93% success rates within with fixed but challenging start and end locations
- Reasonable (80%) success for navigation from a fixed location to an arbitrary one was partially achieved after significantly increasing the n_steps hyperparameter, longer training and a particular type of curriculum learning
- Alternative algorithms (SAC, DDPG, TD3) performed very poorly, possibly due to suboptimal choice of hyperparameters.
- A2C achieved a lower success rate than PPO but produced higher quality trajectories sooner
- Attempts at pre training with “expert” trajectories using BC degraded performance (this included both manually specified and artificial potential field induced examples)
- Initial attempts at *curriculum learning* did not enhance generalization, however a different approach (starting with a simpler reward function) led to decent results (80% success rate)
- LiDAR with longer range 100m performed better than 12m but still (surprisingly) underperformed the “direction and distance to nearest obstacle” unrealistic observation.
- The LiDAR resolution (measurement every 15 degrees) performed as well as much higher resolution

6. Learnings and Limitations

- Getting to truly generalizable navigation is difficult. Much more experimentation is needed to achieve adequate performance. We also plan to explore changes to the observation model to make all observations relative to the current drone heading, the suspicion being that absolute readings impede generalization - we want to be able to learn something like if the obstacle is in front of you, try going left or right.

- Visualizing performance as a simulation in Unreal Engine revealed many deficiencies that might otherwise have gone unnoticed, such as drones gaining excessive height to avoid risk of crashing into the ground

7. Future Directions

1. Implement a Hierarchical RL based end-to-end solution a discussed in Problem Definition
2. Achieve generalizability of the learned model to an arbitrary start and end location and different urban environments
3. Compare success of training with different (simulated) sensors including cameras, LiDAR, sonar
4. Explore navigation in GPS and communication denied and degraded environments
5. Explore navigation in unknown environments using SLAM
6. Transition to physical testing with Pixhawk and Raspberry Pi and Jetson Nano Oren platforms to validate sim-to-real transfer.
7. Expand to swarm-based DFR operations using MARL theoretical frameworks for distributed learning and execution
8. Engage with defense contractors and emergency response agencies for domain-specific applications

Appendix 1: Team Contributions

Victor

- Setup framework(s) for the project
- Produce initial implementation using PPO
- Explored using behavior cloning and curriculum learning to improve generalization
- Performed Literature search

Carlos

- Explored using SAC, DDPG, TD3, A2C
- Wrote initial draft of final report
- Proposed extensions and applications to defense context
- Performed Literature search

Appendix 2: Results details

Experiment	Script	Success Rate (%)	Data	Notes
Dock 1 to burglary	./run_nearest.sh 1	85	loc	
Dock 2 to burglary	./run_nearest.sh 2	94	loc	
Dock 3 to burglary	./run_nearest.sh 3	91	loc	
Dock 1 to burglary lidar 100m	./run_lidar.sh 1 100	0, 82	loc	
Dock 2 to burglary lidar 100m	./run_lidar.sh 2 100	92	loc	
Dock 3 to burglary lidar 100m	./run_lidar.sh 3 100	0, 0	loc	
Dock 1 to burglary lidar 12m	./run_lidar.sh 1 12	0	loc	
Dock 2 to burglary lidar 12m	./run_lidar.sh 2 12	2	loc	
Dock 1 to random	./run_nearest_random.sh 1	40	loc	
	./run_nearest_random.sh 1 –n_steps=1024	34	loc	
	./run_nearest_random.sh 1 –n_steps=4096	50	loc	
	./run_nearest_random.sh 1 –n_steps=8192	61	loc	
	./run_nearest_random.sh 1 –n_steps=16384	67	loc	
Dock 2 to random	./run_nearest_random.sh 2	35	loc	
	./run_nearest_random.sh 2 –n_steps=16384	64	loc	
Dock 3 to random	./run_nearest_random.sh 3	26	loc	
	./run_nearest_random.sh 3 –n_steps=16384	48	loc	
Center to random				
Dock 1 to random, lidar	./run_lidar_random.sh 1 100	0	loc	
	./run_lidar_random.sh 1 100 –n_steps=16384	7	loc	
Dock 1 to burglary + BC	./run_nearest.sh 1 –examples=1	84	loc	
Dock 1 to burglary + BC, lidar	./run_lidar.sh 1 100 –examples=1	0	loc	

	./run_lidar.sh 1 100 –examples=2	91	loc	one example seems to improve over multiple examples from different distributions
	./run_lidar.sh 1 100 –examples=3	0	loc	Multiple examples from similar distribution - success rate collapses
	./run_lidar.sh 1 100 –examples=4	0	loc	100 artificial potential field examples
Dock 1 to random, curriculum	./run_curriculum.sh	68	loc	100:84, 80:90, 60:93, 40:80, 20:68
Dock 1 to burglary, SAC	./run_nearest.sh 1 –algo=SAC	0	loc	
Dock 1 to burglary TD3	./run_nearest.sh 1 –algo=TD3	0	loc	
Dock 1 to burglary DDPG	./run_nearest.sh 1 –algo=DDPG			
Dock 1 to burglary A2C	./run_nearest.sh 1 --algo=A2C --net_size=512	64	loc	Spiked to %60+; worth further exploration
Dock 2 to burglary A2C	./run_nearest.sh 2 --algo=A2C	84	loc	
Dock 3 to burglary A2C	./run_nearest.sh 3 --algo=A2C	79	loc	
Dock 3 to burglary A2C	./run_nearest.sh 3 --algo=A2C --net_size=512	13		
Dock 1 to burglary A2C, lidar	./run_lidar.sh 1 100 --algo=A2C --net_size=512	0		
Dock 1 to random A2C	./run_nearest_random.sh 1 --algo=A2C	4	loc	Spiked to ~%35
Dock 1 to burglary with exp reward	./run_nearest.sh 1 --reward_version=2	63	loc	
	./run_nearest.sh 1 --reward_version=2 --n_steps=16384	0		
	./run_nearest_random.sh 1 --reward_version=2 --n_steps=16384	9		
Dock 1 to equi-distant random				
Dock 1 to random in fixed dir				
Dock 1 to nearest, no altitude penalty	./run_nearest.sh 1 --altitude_penalty=0	91	loc	
Dock 1 to nearest, no altitude penalty	./run_nearest.sh 1 --altitude_penalty=0.5	92	loc	

Appendix 3: Observation, Action and Reward structure

Observation Space

Item	Min	Max
Drone Heading	0	360
Heading to Target	0	360
Heading to Nearest Obstacle	0	360
Distance to Target	0	D * 2
Distance to Nearest Obstacle	0	D * 2
Altitude	0	5 (?)
Crashed?	0	1
VX	-20	20
VY	-20	20
VZ	-20	20
LIDAR DISTANCES *	0	1000

Note:

- Observation Space optionally includes 360 / lidar_angular_resolution lidar measurements (1000 means nothing detected in that direction)
- D is the distance from the specified address for urban world creation (so world is 2*D x 2*D)

Action Space

Item	Min	Max
VX	-20	20
VY	-20	20
VZ	-20	20

Reward Function V1

Item	Value
Each meter closer to target	1
Each meter further from target	-1
Each step	-0.2
Each meter above max altitude	-1
Each meter below min altitude	-1
Crashed	-50
Got to target	100

Reward Function V2

Item	Value
Distance to target	exp(- d * 0.05) * 10
Each meter above max altitude	-1
Each meter below min altitude	-1
Crashed	-50
Got to target	100

Appendix 4: Hardware

Alienware Laptop, Intel Core Ultra 9, NVIDIA RTX 4090	
DJI Mini 3	
Raspberry Pi 4 Model B	
Pixhawk Drone from Drone Dojo kit	

Appendix 5: Annotated References

1. Decentralized Control of Quadrotor Swarms with End-to-end Deep Reinforcement Learning
    - "We demonstrate the possibility of learning drone swarm controllers that are zero-shot transferable to real quadrotors via large-scale multi-agent end-to-end reinforcement learning."
    - In our project we continuously shuttle back and forth between simple simulation, rich simulation and physical hardware to increase the likelihood of zero-shot transfer
2. Parallel Reinforcement Learning Simulation for Visual Quadrotor Navigation
    - Simulation framework, built on AirSim, which provides efficient parallel training
    - *Training of models directly on AirSim can be very expensive. We used a simple and computationally cheap simulation environment and when necessary fine-tuned the models in AirSim.*
3. Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms
    - Review the theoretical results of MARL algorithms mainly within two representative frameworks, Markov/stochastic games and extensive-form games
    - Fully cooperative, fully competitive, and a mix of the two
    - Learning in extensive-form games (strategic interactions in game theory, where players make sequential decisions over time, and the game's structure is modeled as a game tree. They are particularly relevant for understanding complex, multi-step decision-making processes)
    - Decentralized MARL with networked agents
    - MARL in the mean-field regime (Instead of modeling every individual agent's interactions, the collective behavior is approximated by averaging the effects of all agents into a mean-field—a single, representative distribution of the agents' states, actions, or policies)
    - *This provided some inspiration for our project but mostly remains to be explored in follow up work.*
4. Deep Reinforcement Learning for Vision-Based Navigation of UAVs in Avoiding Stationary and Mobile Obstacles
    - From the abstract: "this paper describes a methodology for developing a drone system that operates autonomously without the need for human intervention. This study applies reinforcement learning algorithms to train a drone to avoid obstacles autonomously in discrete and continuous action spaces based solely on image data. The novelty of this study lies in its comprehensive assessment of the advantages, limitations, and future research directions of obstacle detection and avoidance for drones, using different reinforcement learning techniques. This study compares three different reinforcement learning strategies—namely, Deep Q-Networks (**DQN**), Proximal Policy Optimization (**PPO**), and Soft Actor-Critic (**SAC**)—that can assist in avoiding obstacles, both stationary and moving; however, these strategies have been more successful in drones. The experiment has been carried out in a virtual environment made available by **AirSim**. Using Unreal Engine 4, the various training and testing scenarios were created for understanding and analyzing the behavior of RL algorithms for drones. According to the training results, **SAC outperformed the other two algorithms. PPO was the least successful among the algorithms, indicating that on-policy algorithms are ineffective in extensive 3D environments with dynamic actors**. **DQN and SAC, two off-policy algorithms, produced encouraging outcomes**. However, due to its constrained discrete action space, DQN may not be as advantageous as SAC in narrow pathways and twists. Concerning further findings, when it comes to autonomous drones, **off-policy algorithms, such as DQN and SAC, perform more effectively than on-policy algorithms**, such as PPO."
    - *We tried to replicate this result using PPO and SAC but using low-fidelity sensor data rather than images. PPO was successful, but SAC was not.*

5. Potential Fields Guided Deep Reinforcement Learning for Optimal Path Planning in a Warehouse
    - Potential fields are employed to guide to collect better quality training data to improve data efficiency
    - *For our project: potential fields are notoriously a very unreliable method for navigation - often falling victim to local minima and stationary points. However, if we throw away the unsuccessful trajectories, can the successful ones be used as a proxy for expert demonstrations? Our results suggest that the answer is no.*
6. The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games
    - From the abstract: “Proximal Policy Optimization (PPO) is a ubiquitous on-policy reinforcement learning algorithm but is significantly less utilized than off-policy learning algorithms in multi-agent settings. This is often due to the belief that PPO is significantly less sample efficient than off-policy methods in multi-agent systems. In this work, we carefully study the performance of PPO in cooperative multi-agent settings. We show that PPO-based multi-agent algorithms achieve surprisingly strong performance in four popular multi-agent testbeds: the particle-world environments, the StarCraft multi-agent challenge, Google Research Football, and the Hanabi challenge, with minimal hyperparameter tuning and without any domain-specific algorithmic modifications or architectures. Importantly, compared to competitive off-policy methods, PPO often achieves competitive or superior results in both final returns and sample efficiency. Finally, through ablation studies, we analyze implementation and hyperparameter factors that are critical to PPO's empirical performance, and give concrete practical suggestions regarding these factors. Our results show that when using these practices, simple PPO-based methods can be a strong baseline in cooperative multi-agent reinforcement learning.”
    - *This is a bit surprising and contradicts paper number 3 above. Our project arrived at a similar conclusion.*