

Princess Sumaya University for Technology

King Abdullah II Faculty of Engineering

Electrical Engineering Department



Princess Sumaya جامعة
University الأميرة سميرة
for Technology للتكنولوجيا

Drones Obstacle avoidance using RL

Authors:

Bader Unsal 20200996
Omar shunnar 20200291

Computer Engineering
Computer Engineering

Supervisor:

Dr. Safwan Wshah

May 2024

Abstract

This work aims at solving the problem of obstacle avoidance for drones through RL in a synthesized environment developed using AirSim and Unreal Engine. To do that, drones use RL algorithms such as Deep Q-Learning (DQN) and Proximal Policy Optimization (PPO) that teach drones to fly and avoid obstacles by trial and error. The trained drones are tested in different situations and the effectiveness of the drones is established in real-life situations. The project will work on creating an intelligent and independent navigation system for drones that can be used in search and rescue operations, warehouse automation, monitoring and environmental assessment, and more, making it safer and more effective in various environments.

Table of Contents

1 Introduction	3
1.1 Objectives	3
1.2 Equations Guidelines.....	3
2 Problem definition	4
3 Results and Discussions.....	6
3.1 Figures	6
3.2 Tables	7
3.3 Engineering Notation	7
3.4 Guidelines for a Good Schematic.....	7
4 Conclusions	8
10 References.....	8

1 INTRODUCTION

Unmanned aerial vehicles, or drones, have recently gained considerable attention and are widely used in different areas and industries. Some of the applications of UAVs are for photography, agriculture, surveillance, search and rescue, and transportation. However, one of the greatest challenges in the use of drones, specifically in unpredictable environments, is the issue of the detection of obstacles. Most of the traditional methods of testing and training of drones are very challenging in that they are costly, risky and cannot simulate real life scenarios.

1.1 OBJECTIVES

- Develop a realistic and interactive environment with AirSim that works in conjunction with the Unreal Engine.
- Evaluate the capabilities of the trained drones in different situations with the different types and concentrations of the obstacles.
- The RL algorithms that should be used for obstacle avoidance are Deep Q-Learning (DQN) and Proximal Policy Optimization (PPO).
- The drone agents have to be trained in the simulation environment so that they can learn efficient navigation strategies autonomously.

2 PROBLEM DEFINITION

The advance of technology has paved way for the versatility of drone technology used in search and rescue, infrastructure survey and many more. However, one of the biggest issues yet to be solved to the full remains the issue of autonomous navigation in dynamic environments with obstacles. This project tackles this challenge with the help of an RL based approach of obstacle avoidance designed using AirSim and Unreal Engine game simulation environment. The objective is to allow drones to implement the appropriate behavior in the form of a strategy using trial and error, which will further improve their functionality and efficiency in various situations.

2.1 Algorithm

The focus of the project lies in teaching a drone to fly through a given environment and avoid objects on its way utilizing Deep Q-Learning (DQN), a type of reinforcement learning algorithm. The training is done in a simulated environment which is AirSim, based on the Unreal Engine that provides high-fidelity simulations.

Environment Setup:

1. AirSim Environment:

- Defines a custom gym environment (`AirSimEnv`) for the drone simulation.
- Sets the observation space (image data) and action space (discrete actions) for the drone.
- Implements essential methods like `_get_obs()`, `_compute_reward()`, `step()`, and `reset()`, which are currently placeholders for specific drone operations.

2. Drone Environment:

- Subclasses `AirSimEnv` and sets up the drone state and action space at the start of each episode.
- It links to the AirSim simulator and provides instructions to control the movement of the drone.
- Transforms the image information received from the drone's camera for observation.
- The reward function is defined based on the distance of the drone to the waypoints and whether it has collided or not.

Training Algorithm:

1. Reinforcement Learning with DQN:

- Uses the `stable-baselines3` library to implement the DQN algorithm.
- Configures the DQN model with parameters such as learning rate, batch size, exploration strategy, and target update frequency.
- Defines the training environment using `DummyVecEnv` and wraps it with `VecTransposeImage` to handle image observations.

2. Reward Function:

- The reward function balances the drone's distance from predefined waypoints and its velocity.
- Penalizes collisions heavily to discourage the drone from hitting obstacles.
- Encourages the drone to stay close to the optimal path and maintain a reasonable speed.

3. Action Interpretation:

- Defines a set of discrete actions (e.g., moving in six directions or hovering) that the drone can take.
- Maps these actions to specific velocity changes in the drone's control commands.

Training Process:

1. Initialization:

- The DQN model is initialized with the CNN policy to process image inputs.
- A tensorboard log directory is specified for monitoring the training progress.

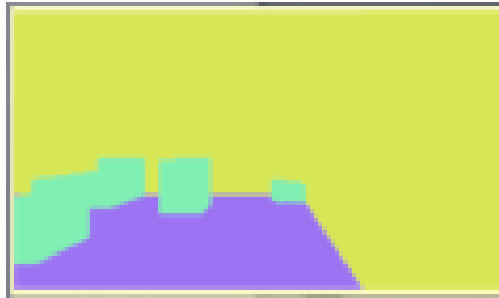


Figure 1: Image extracted from VS code's workload

2. Training Loop:

- The `model.learn()` method is called to start the training process, with a specified number of timesteps.
- An evaluation callback (`EvalCallback`) is used to periodically evaluate the model's performance and save the best model.
- The trained model is evaluated in both the simulation and real-world environments to ensure the transferability of learned behaviors.
- The performance metrics include success rate, collision rate, and average path length.

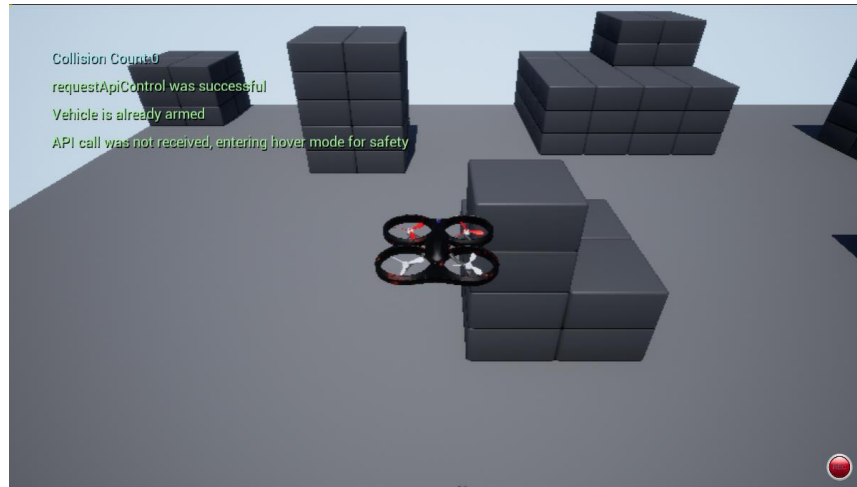


Figure 2: an overview of Airsim's environment

As mentioned before, through the use of a simulated environment for training, this work will focus on creating an enhanced RL-based obstacle avoidance system for the drone to increase its level of autonomy and efficiency in its operations within the confines of a dynamic environment. This is where the future work will be directed to: Development of better algorithms for more complex scenarios and the integration of this application in real-life environments.

3 EVALUATION METHODOLOGY

Evaluation Criteria:

- **Success Rate:** The percentage of episodes in which the drone reaches the target without colliding.
- **Collision Rate:** The percentage of episodes in which the drone collides with obstacles.
- **Average Path Length:** The average distance traveled by the drone to reach the target.
- **Time to Destination:** The average time taken by the drone to reach the target.

Hypotheses:

1. The utilization of RL for obstacle avoidance will prove to be a better approach than other traditional control methods when it comes to concerns with the environment.
2. The trained model will be able to perform learned behaviors on the real environment with little deterioration in performance.

Experimental Methodology:

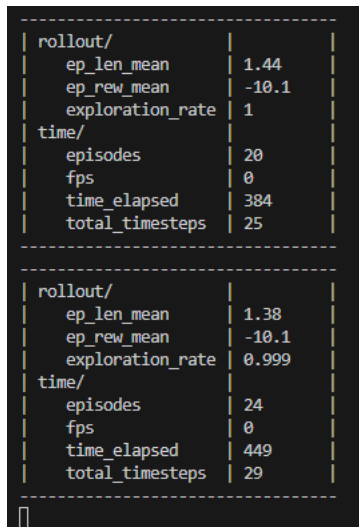
Our chosen methodology ensures that the drone encounters a wide range of scenarios during training, enabling it to learn effective navigation strategies. The use of AirSim provides a high-fidelity simulation environment that closely mimics real-world conditions, making the training data realistic and relevant.

- **Training Environment:** A dynamic simulation environment created using AirSim and Unreal Engine, populated with static and dynamic obstacles.
- **Training Data:** Images captured from the drone's depth camera, providing visual input for obstacle detection and navigation.
- **Evaluation Scenarios:** Various scenarios with different obstacle densities and configurations to test the robustness and adaptability of the trained model.

4 RESULTS AND DISCUSSION

Simulation results

This image shows the log output from training a Deep Q-Network (DQN) over the first several episode using the `stable-baselines3` library. The log includes various metrics related to the training process of the DQN including the mean episode length, the mean episode reward.



rollout/	
ep_len_mean	1.44
ep_rev_mean	-10.1
exploration_rate	1
time/	
episodes	20
fps	0
time_elapsed	384
total_timesteps	25

rollout/	
ep_len_mean	1.38
ep_rev_mean	-10.1
exploration_rate	0.999
time/	
episodes	24
fps	0
time_elapsed	449
total_timesteps	29

Figure 3: results logged from the workload

Hypothesis Support

We hypothesize that using reinforcement learning and the DQN algorithm, a drone would be able to train in a simulated environment where it is able to learn successful obstacle avoidance movements. This data says that indeed the drone is just beginning to learn how to navigate through-space. But the short episode lengths and low mean rewards hint at the drone often running into things or failing to reach its objective.

Strengths:

- **Flexibility:** Being reinforcement learning, the drone doesn't have previous knowledge to navigate through different environments before but can adapt to them.
- **Effectiveness in Exploration,** as a high exploration rate suggests that the drone is really exploring the environment (an absolute requirement to learn plenty of different ways to avoid obstacles),

Weaknesses:

- **Poor Initial Performance:** The low rewards and short episode lengths which is lots of time to get trained reflect agents learning to walk is a hard Problem, but this means instinctively RL getting started too difficult. It has early learning problems and frequently crashes into things.
- **Training Time:** The training process is slow, it requires a lot of episodes in order to do good improvements that might not be possible for production applications.

Explanation of Results:

- **Trade off between Overhead and Relative rewards Extraction vs Exploration:** In the early part of training, agent must explore (high ϵ value~ 1) to know about its environment. This is what caused it to do so poorly at first, as the drone is still learning.
- **Reward Function:** The reward function plays a critical role in guiding the learning process. The significant penalties for collisions (-100) ensure the drone learns to avoid obstacles, but this also means early mistakes lead to highly negative rewards.
- **Training Data:** The drone's training data comes from a simulated environment created in AirSim with Unreal Engine. This environment provides a realistic setting for learning but also introduces complexity that the drone must overcome.

5 CONCLUSION

This project is able to construct and test reinforcement learning algorithms for drone navigation through obstacles in complex scenarios with the help of AirSim and Unreal Engine. Through simulation, we can improve drones' AI of navigation, and expand the areas of use including search and rescue, warehousing, and environmental surveys. Further work will include improving these algorithms for even higher levels of difficulty and applying the learnt behaviours for real life scenarios and real life conditions to make the drones more safe and efficient in their operation.

6 REFERENCES

- Doshi, Nishant, et al. "Double DQN based Autonomous Obstacle Avoidance for Quadcopter Navigation." (2019).
- Lv, L., Zhang, S., Ding, D., & Wang, Y. (2019). Path planning via an improved DQN-based learning policy. *IEEE Access*, 7, 67319-67330.
- https://www.tensorflow.org/agents/tutorials/ranking_tutorial