Project Proposal

Aman Tiwary
Aeronautics and Astronautics Department
amant10@uw.edu

Pengcheng Chen
Department of Mechanical Engineering
pengcc@uw.edu

Zhitao Yu
Department of Mechanical Engineering, Statistics, Applied Math
zhitaoyu@uw.edu

1 Introduction

1.1 Micromechanical Flying Insects (MFIs)

Micromechanical Flying insects (MFIs) also known as the "insect robot", are inspired by the mechanism of flapping wings of insects. Generally, a typical modern MFIs' main body structure is composed of two or more insect-inspired flapping wings, a piezoelectric ceramic actuator as the engine that drives the flapping wings, and a set of highly integrated micro-electro-mechanical system (MEMS) that provides the energy, sensor, and controller for the robot.(Figure. 1) In 2008, RJ Wood fabricated the first modern MFIs in Harvard[4]. Its' diminutive size and agility allow it to enter many environments that regular unmanned aerial vehicles (UAVs) cannot perform.

However, the tiny and delicate structure brings a new challenge to further development. Most of the automatic navigation algorithms for UAVs cannot directly transfer into the MFIs because the processing unit inside the MFIs is too small to have a similar performance as regular UAVs, thus, for most tasks of insect robots, we need to develop algorithms with low computational power.

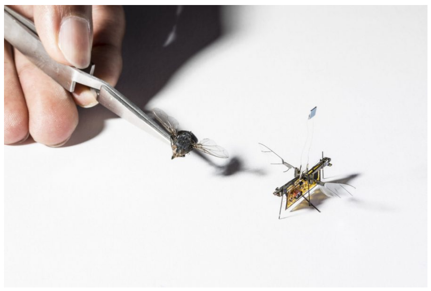


Figure 1: Profile of a typical micromechanical flying insect[1]

1.2 Tasks and Advancement

Corridor passing and obstacle avoidance are two primary tasks that are widely applied in autonomous UAVs, while they the still absent in MFIs research, our project is attempting to fill in the blanks and provide a basic autonomous cruising ability to the MFIs.

Our task is to navigate the MFIs through a narrow corridor with obstacles (shown in section 4) given only visual input. Visual navigation for MFIs is very challenging because in such a small scale, the size, weight, and power (SWaP) constraints do not appear to permit visual navigation techniques such as SLAM (Simultaneous Localization and Mapping). After all, they are likely to be too power-hungry.

In our project, we are going to try to use reinforcement learning algorithms to train a policy that needs low computational power to make the inference. Note that we don't care about the computational power for the training process because the training is offline, but once the training is done, we will have a policy for the robot to do the navigation, and we need this policy function to be simple.

And our work will be purely on simulation given the time constraint. We will use the quadcopter drone as our robot because we cannot find the MFIs developed in the simulation environments we selected so far, but our algorithm will aim for low computational power.

1.3 Expected Outcomes

In expectation, the outcomes will contain:

1) A low-computational-cost model-free policy that could guide the quadcopter through a corridor maze and avoid environmental obstacles (both conducted in the simulator).

2) A video demonstrating how the quadcopter passes the corridor and obstacles tasks.

3)* Transfer the policy from quadcopter into MFIs onboard and go through the same tasks (Might not be able to achieve due to the time limitation. We will continue after this course and have a publication.)

1.4 Challenges

In this project, the challenges would be varied. Primarily, there are four main challenges we might face with:

1) How to build a task environment through the simulator.

2) How to develop a neural network-based reinforcement learning algorithm that could pass through the corridor and obstacle tasks. Future work will focus on constraints, such as under a low sample rate (low FPS and poor image quality).

3) How to reduce the computation cost. Given the policy would be simple when the trained algorithm is applied to the UAVs.

4)* How to transfer the trained algorithm from the quadcopter to MFIs onboard. (Might not be able to achieve due to the time limitation)

2 Environment/platform

Airsim (Aerial Informatics and Robotics Simulation) is an open-source, cross-platform simulator for drones, ground vehicles such as cars and various other objects, built on unreal engine 4 (UE4). We will use the Airsim plugin and UE4 to demonstrate the path planning of the drone.

OpenAI Gym is an open-source python library for developing and comparing reinforcement learning algorithms by providing a standard API to communicate between learning algorithms and environments.

In this project, we will implement RL algorithms in AirSim(which runs on UE) using the OpenAI gym wrapper around the AirSim API. The environment setup is developed in Unreal Engine. Airsim provides the drone and the API to control it and also to record images. In order to use AirSim as a gym environment, we extend and implement the base method such as `step`, `_get_obs`, `_compute_reward` and `reset` specific to AirSim and the problem statement.

2.1 Software Architechture

The software architecture is shown in Figure2, each pathfinding environment contains 2 actors (DroneActor, TargetActor), they are used to detect the collision and whether the drone hit the target. The drone movement module contains 2 classes as well, they are relevant to the UE actors. To simplify the problem, the scoring protocol is declared as follows: If the drone collides with the wall, then the game is over, and denote 0 scores; if the drone successfully hits the target, then denote 1 score and stop the game.

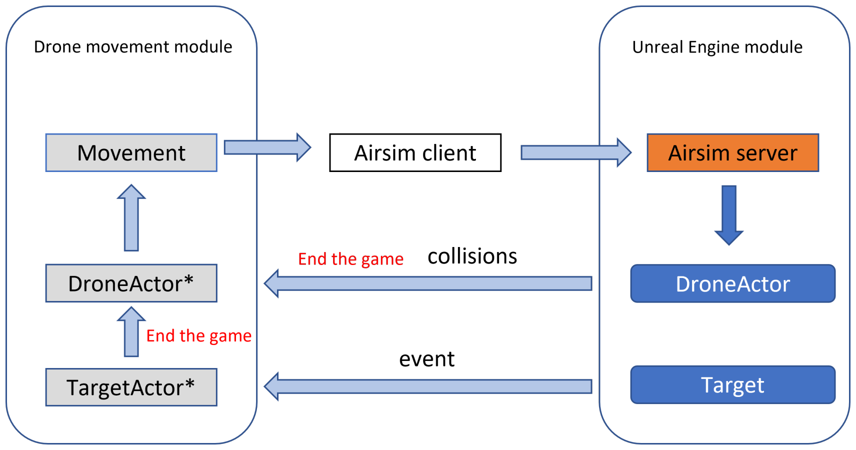


Figure 2: Software architecture

3 Methodology/algorithm

In this project, we would like to train RL algorithms. currently, there are two ideas:

1. Based on Deep Q Network(DQN), we will deep dive into how to reduce the computational cost so that the algorithm can be applied onboard on the MFIs.The current idea is that instead of using the deep neural network to fit the policy, we would like to use a shallow neural network for training to see whether it can converge.
2. PPO (Proximal Policy Optimization Algorithms) [3]. PPO is a state of art RL algorithm which belongs to the family of policy gradient methods. It can learn simple policies efficiently, which is aligned with the project target.

Note that in current MFIs, the limitation of the maximum load only allows the installation of a monocular camera, while developing an auto-navigation policy on monocular video is a tough task given the diminutive size. For implications, the camera input will be stereo in this work, which the depth and RGB information will both be provided. The monocular input-based version will be developed thereafter.

There would be 3 key metrics for success. 1) pass through all the training tasks (corridor task and obstacle task), 2) pass through the testing tasks 3) the computational cost (time) onboard for inference should be low enough for MFIs.

One of the challenges will be collecting the data for training the model. We know RL is a data-hungry algorithm. We need to have a clever way to collect data on the 3D environment we build.

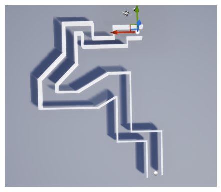
Generally, how to carefully balance the trade-off between computational cost and model accuracy is the hardest part we might confront. However, as a course project, we might not give full research on this issue, but definitely will do it in future work.

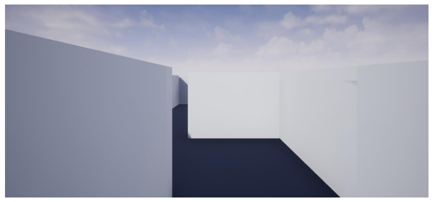
The potential risks will be the models we proposed fail to learn a good simple policy for us to show the demo, but once we have the instructor team approval of the models, we will do our best.

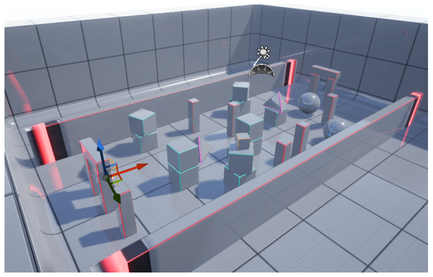
4 Demo

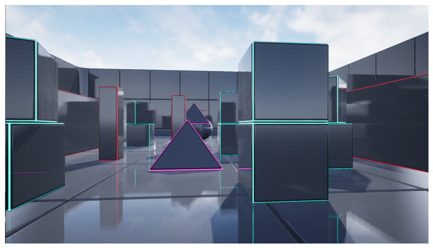
In this work, we are using Airsim and UE4 to build a 3D environment with the obstacles in the corridor and we will show a demo that the drone can navigate through that corridor using the policy trained from our reinforcement learning algorithm. The 3D corridor and its close view are shown in Figure. 3a. The respawn point(start point) of the drone is located in the upper right corner, and the target(endpoint) is located on the right downside. The corridor is a 3d animation of the 2D corridor of Figure 3e [2].

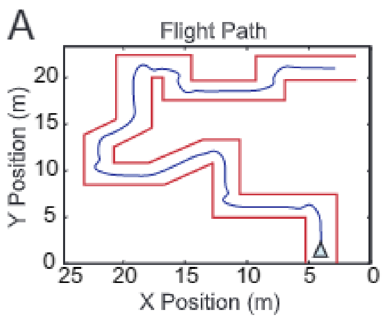
Similarly, we rendered a demo for the obstacle avoidance task, which is shown in Figure 3c, and the close view is shown in Figure 3d. The drone will respawn on the left side of the maze, and the target for this task is to navigate the drone to reach the door on the right side without collision.


(a) 3D corridor

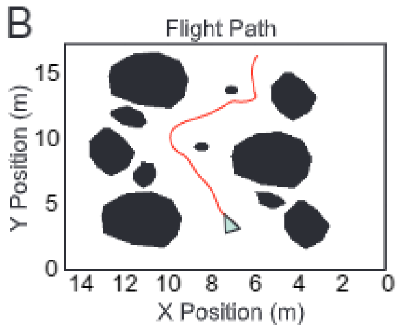

(b) 3D corridor close view


(c) 3D obstacle task


(d) 3D obstacle task close view



(e) 2D corridor following



(f) 2D obstacle avoidance

Figure 3: Illustration for our task

The plan to implement the demo. We would like to have a demo showing that the drone can navigate through a corridor with obstacles.

5 Timeline and Future work

1, developed the RL algorithm in either DQN or PPO (or both, after collecting the feedback from the instructor team).

2, set up the Airsim and UE4 3D environment and collect the data.

Before Dec 3rd, we should have finished training the model and have a policy worked for the demo.

Dec 5th: Presentation

Dec 12th: Final report.

One future work will be applying the low-computational-power algorithm on the insect-scale flying robot definitely, such as Robofly.

References

[1] Yogesh M. Chukewad, Johannes M. James, Avinash T. Singh, and Sawyer B. Fuller. Robofly: An insect-sized robot with simplified fabrication that is capable of flight, ground, and water surface locomotion. *IEEE Transactions on Robotics*, 37:2025–2040, 2021.

[2] J Sean Humbert, Richard M Murray, and Michael H Dickinson. A control-oriented analysis of bio-inspired visuomotor convergence. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 245–250. IEEE, 2005.

[3] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[4] Robert J. Wood. The first takeoff of a biologically inspired at-scale robotic insect. *IEEE Transactions on Robotics*, 24(2):341–347, 2008.