Adnan Menderes University
Computer Engineering
Autonomous Drone Herd Project

Lecturer: Mahmut Sinecen

Student: Kemal Yildirim
Student Number: 161805025

Report Submission Date: 20.06.2019

Contents

Abstract

Unmanned Aerial Vehicles used in various fields, for instance, packet delivering, in military combat aircrafts, filming movies and concerts and so on. Building an autonomous control system for drones will make huge effect in all this fields and less human means less human error, and machines less likely make an error compared to humans.

Introduction

Problem Definition

Purpose of this project is design a drone herd contaning at least three drones with using Raspberry PI. However, project is simplified due to financial issues and complex herd design mechanics. So with simplifying, project content limited to making an autonomous drone simulation.

Plan of Development

First Atempt: Drone herd with N number of drones in 2D Map environment

After the limiting the project with simulation decision, my first atempt was creating a 2D drone herd simulation using based on this paper [1].
My goal was develop a basic web application using Angular.js and simulating that which drone will be sacrificed for destroying a target in that map and using neural networks for that decision making.

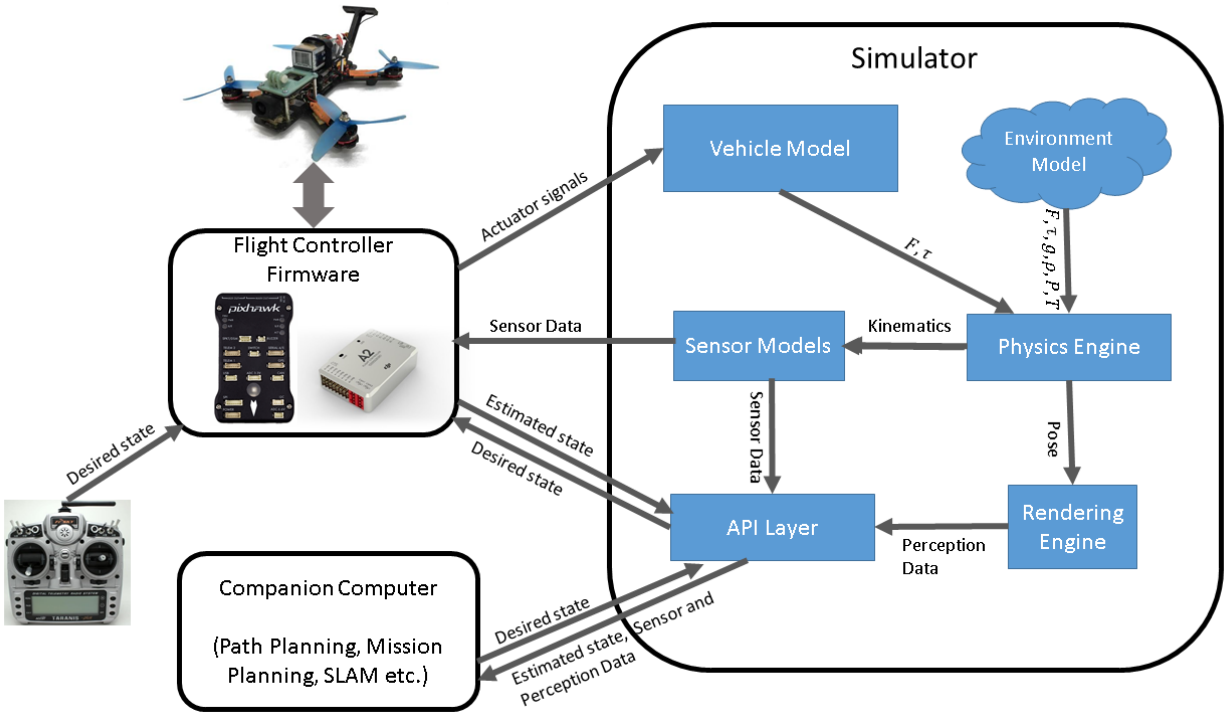
I had issues when I apply artificial bee algorithm to my simulation, algorithm worked fine with test apps(Rosenbrock, Rastrigin) but I could not apply it to my simulation.

Second Atempt: Reinforcement Learning and Microsoft Airsim

While working on first atempt, I discovered Microsoft’s opensource Airsim. Airsim is an open source simulation developed for specifically autonomous vehicles and built on Unreal Engine and Unity [2].

Info about Microsoft Airsim

Architecture



API is written also in Python, so no need to extra effort while working with tensorflow and/or keras.
Based on this architecture, no need to work with Raspberry PI modules, electronical circuits, brushless motors and so on.

More can be found in Microsoft Research paper [3].

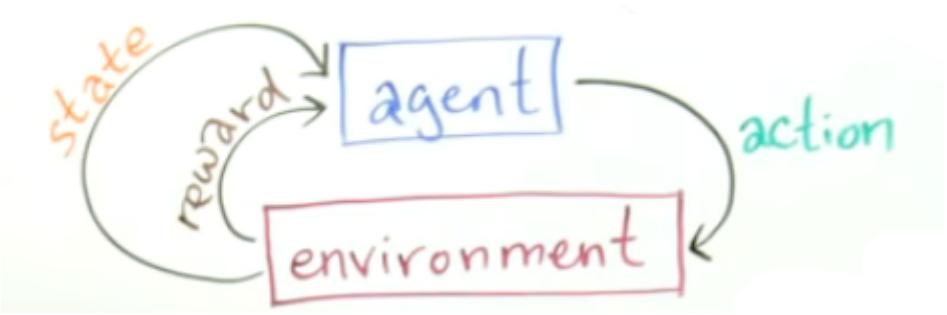
However, API was not much reliable; often landed_state informations, collision informations were not simultaneous to the simulation. This problem effected the learning step badly.

Reinforcement Learning

I have heard about reinforcement learning but I never had the chance applying in practice, so when I had this simulation chance I gave it a shot.

While searching, I encountered with Deep Q Learning method, reinforcement learning involved with deep neural networks, basically.

Deep Reinforcement Learning



Our agent(the drone) takes an action in our environment, then the environment feedbacks the agent with two information: state and reward. The agent tries to learn the overall optimal policy what action to take in any possible state.

Deep Q-Learning Network

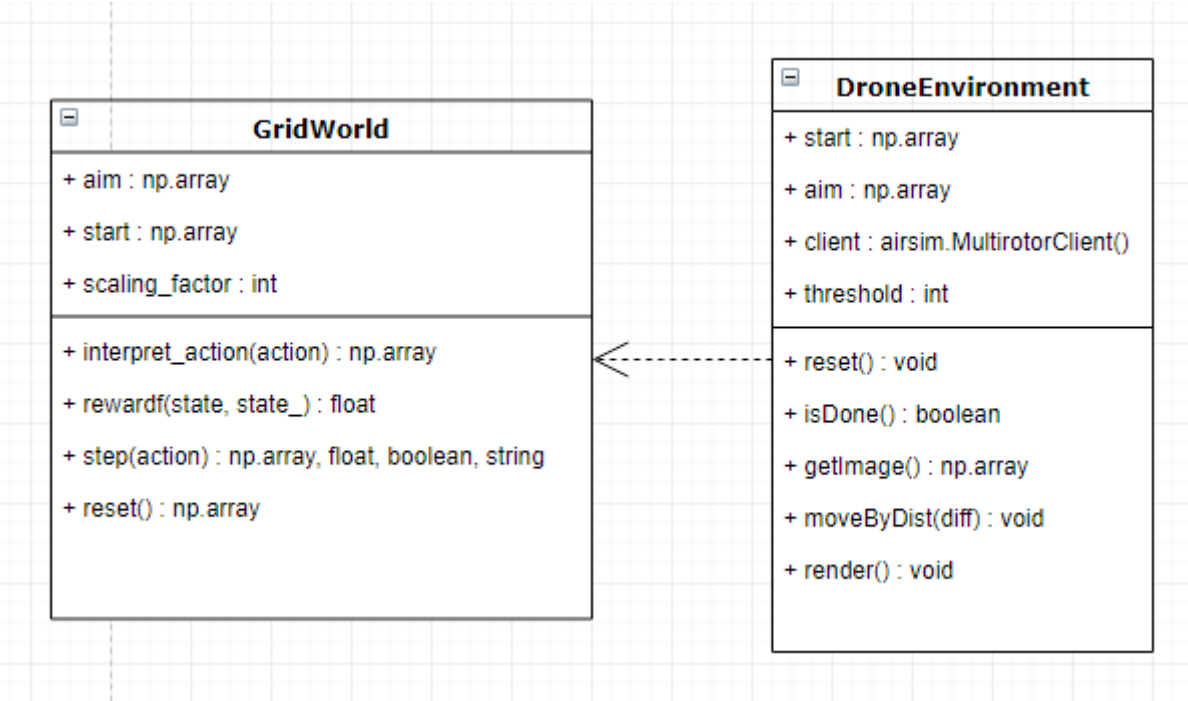
Two additional concepts included in DQN's; value function and Q value function. Value function is the excepted cumulative reward for being in a particular state. Q value function what is the maximum excepted reward for being in that particular state and given a particular action.

Q* is the optimal Q value function. Our goal is approximate the Q* with using Deep Q-Learning Networks.

More detailed lecture can be found in LiveLessons Youtube channel [4].

Implementation

1. Environment implementation



DroneEnvironment:

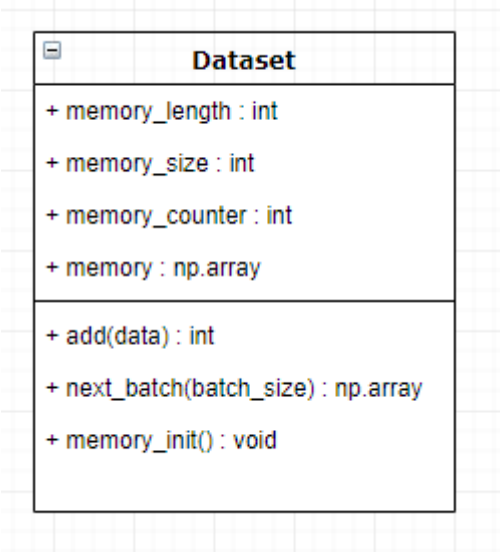
start, aim : Starting and ending vectors converted to np.array's.
client: client, connected to drone server, controls the drone.
threshold: goal threshold.
reset(): Set drone position to [0, 0, -5].
getImage(): Get depth vision from front camera.
isDone(): Checks if its done or not.
moveByDist(diff): moves with diff[x,y,z] velocities, in axis respectively for 1 sec.
render(): prints information about the environment at that instance.

GridWorld: Subclass of DroneEnvironment

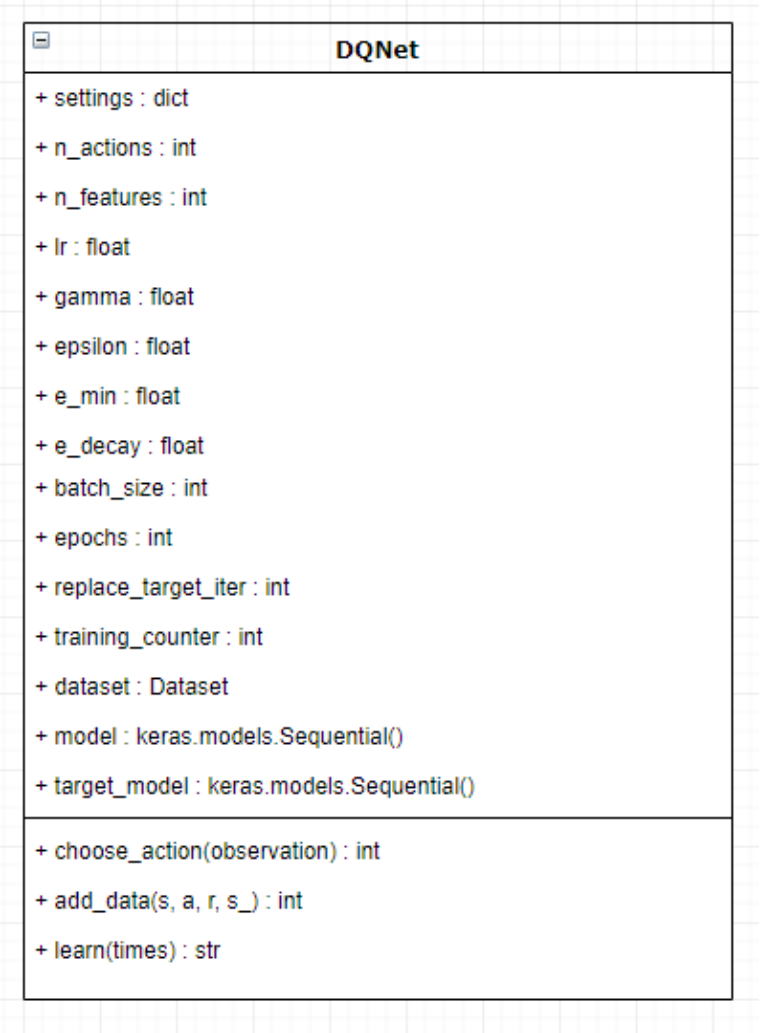
scaling_factor: action paramater.

interpret_action(action): Sets the movement direction for 3 axis, 6 directions(x-, x+, y-, y+, z-, z+).
rewardf(state, state_): future reward function by previous state(state) and current state(state_).
step(action): Chooses an action and takes it, then returns state, reward, action ,done and additional informations.
reset(): resets drone and returns the state.

2. Agent Implementation



Firstly, we have a dataset class which creates and two dimensional array and has a function that return random size of batches from that array. We use this dataset to remember our actions, states and rewards and for a random intervals agent remembers this data and uses it instead of randomly exploring.



DQNet agent based on the theory given in [4].

Main Script

```
import numpy as np
import keras
import airsim
import time
import droneEnvironment
from dqn_net_keras import DQNet
from dqn_net_keras import Dataset
import tensorflow as tf
```

Dependencies

```
settings = {}

settings["learning_rate"] = 0.003
settings["reward_decay"] = 0.9
settings["memory_length"] = 81920
settings["batch_size"] = 32
settings["epochs"] = 1
settings["replace_target_iter"] = 50
settings["model"] = test_model
settings["n_actions"] = 7
settings["n_features"] = 6
```

Hyperparameters

After these, in while True;

*Resets the environment in conditions collision, out of range
*If succeed lands the agent.

1Problems Encountered

When we think first purpose of this project and the simplified version, the major problems are:

1- Buying a designed drone is very expensive, building your own drone is very expensive as well. However, buying a designed drone could limit your low-level access to it, so if you really want a real drone I suggest building one with Raspberry PI.

2- If want to built a drone herd, their communication will be very difficult.

3- Considering this report's date, Airsim drone server if not very reliable as I said earlier. It could be also an environment issue but I encountered very often that when the drone is crashed into an object API do not receive that it is crushed, same problem for landed_state information.

4- In my project, my deep RL network's states not including the image which I get from DroneEnvironment.getImage(). This function returns the np.array which it gets from the front camera and turns it into a depth image. This could be huge level up because if an object is present in the camera these areas are black, else white but, I could not make the application into the deep neural network.

5- Drone does not move very smoothly, basically it moves step by step, rather than moving on a vector until it encounters an object. I think this is happening because my environment design is bad.

References

[1] : http://fbd.beun.edu.tr/index.php/zkufbd/article/view/698
[2] : https://github.com/microsoft/AirSim/
[3] : https://www.microsoft.com/en-us/research/wp-content/uploads/2017/02/aerial-informatics-robotics.pdf
[4] : https://www.youtube.com/watch?v=OYhFoMySoVs

Hands-On Machine Learning with Scikit-Learn and Tensorflow – Aurélien Géron

Extra Resources

Real drone design : https://www.instructables.com/id/The-Pi-Quadcopter/
Real drone design 2 : https://www.instructables.com/id/Autonomous-Drone/
MIT’s drone design simulator for whether if your drone flies or not : https://github.com/mit-gfx/multicopter_design
Artificial bee algorithm implementation : https://github.com/rwuilbercq/Hive
A hello world 2D map application : https://codepen.io/MarcMalignan/pen/jABfk
A long reading about RL : https://skymind.ai/wiki/deep-reinforcement-learning