e-YSIP 2019

Reinforcement Learning in Drones

Interns:
Karthik P.B
Keshav Kumar
Mentor:
Vikrant Fernandes
Duration of Internship:
22/05/2019-05/07/2019

2019, e-Yantra Publication

Reinforcement Learning in drones

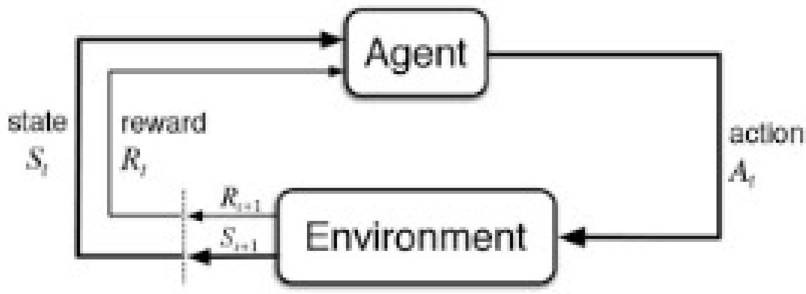
Objective:

Design and Develop a Reinforcement Learning (RL) module for Drone navigation and path planning by avoiding the obstacles placed in real world.

Abstract:

Reinforcement learning is a technique in which the agent is unaware of the environment it needs to act in and hence take some random actions and gets information about the environment using a reward system that is given by environment based on agent's actions.
An episode is defined as reaching terminal state or the agent is out of observation space. There are several techniques of reinforcement learning that are monte-carlo and temporal difference. Temporal difference can be implemented using SARSA(On Policy method) or Q-Learning(Off policy method). Here, in the project we have used Q- Learning.

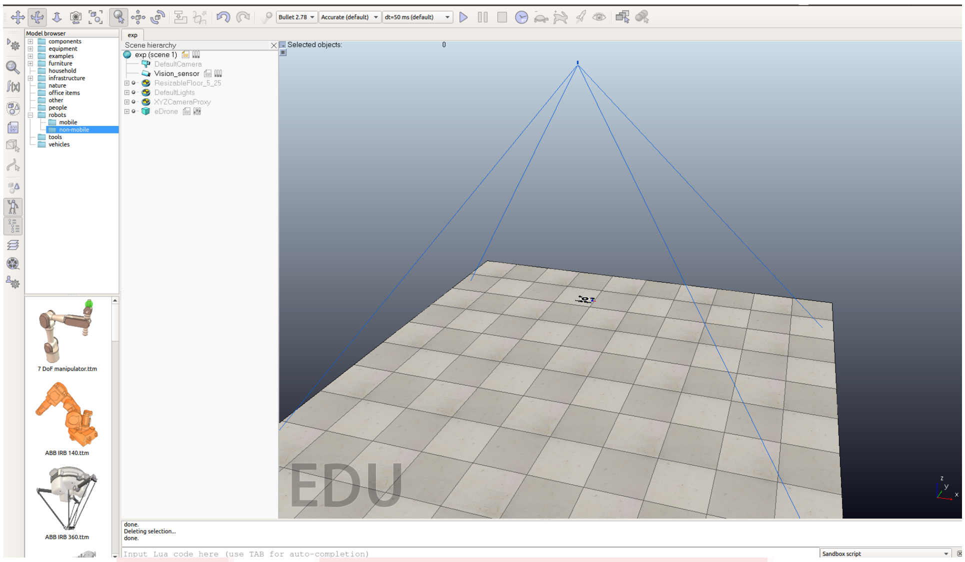
Q-learning is a model-free reinforcement learning algorithm which generates Q-table after it gets trained to act in any environment. The quality of our policy will improve upon training and will keep on improving.



Completion Status:

The deliverables of the project has been successfully completed.

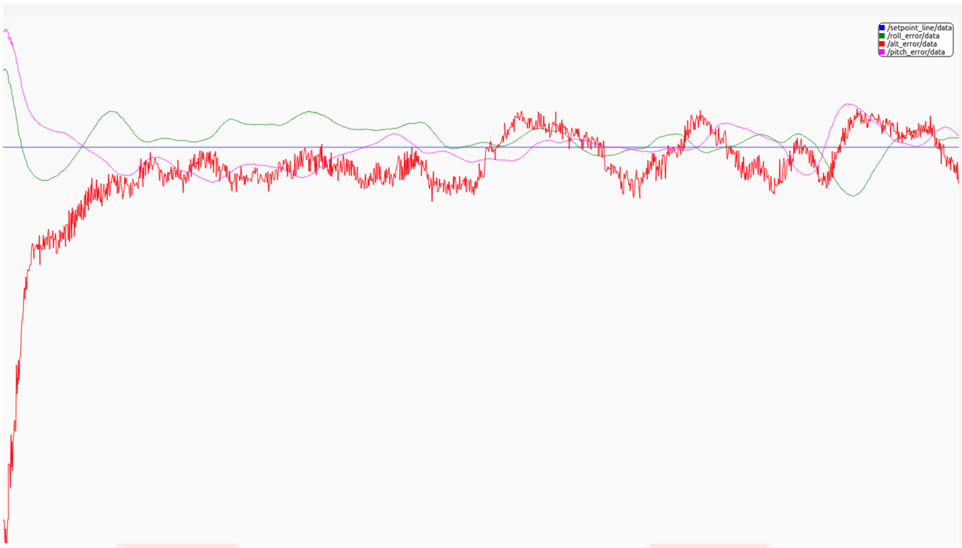
- **Simulation in V-REP.**
  V-rep scene developed considering the drone as our agent so that it could be trained and tested on the simulation first in order to make the agent acquainted with the environment.

- **Hardware Testing of PlutoDrone and Camera Calibration to remove fish-eye using check board.**

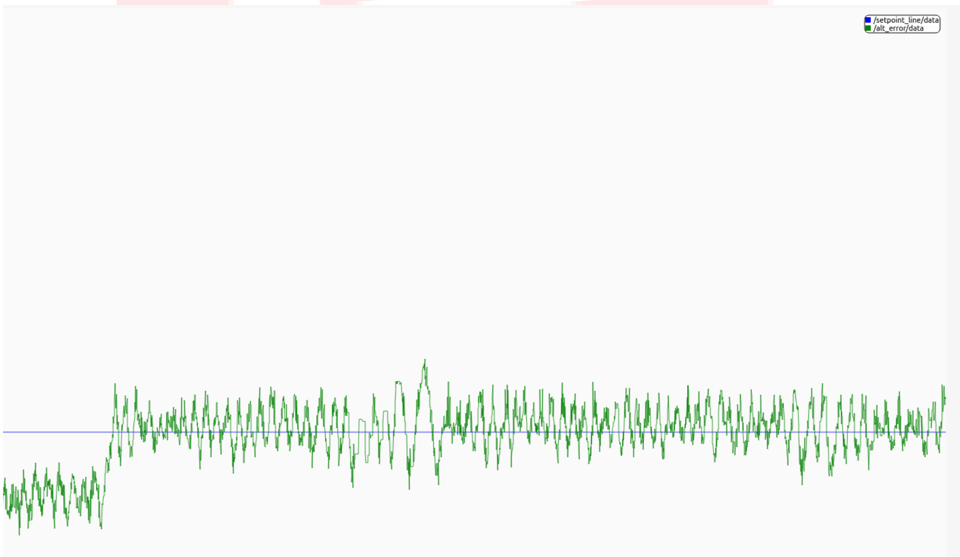
- **PID tuning of Drone has been done in order to do the comparative analysis of Reinforcement Learning and PID.**

  Graphs of the values given by the parameters(Kp,Kd,Ki) has been plotted for throttle,pitch and roll against the setpoint at which the drone needs to hold its position.



graph representing the variation of the current value of drone in respective axis w.r.t the setpoint(straight line)

- **Position hold of drone using the Q-Learning algorithm of RL.**



this graph represents the variation of current values of altitude w.r.t the setpoint

Q-learning

It is a model-free reinforcement learning algorithm which generates Q-table after it gets trained to act in any environment.

The quality of our policy will improve upon training and will keep on improving.

Q-table maps from different states to what actions to take using state action values [state, action]. We then update and store our q-values after an episode. This q-table becomes a reference table for our agent to select the best action based on the q-value.

Here are the 3 basic steps:

- Agent starts in a state takes an action and receives a reward.
- Agent selects action by referencing Q-table with highest value (max) OR by random (epsilon,).
- Update q-values to generate Q-table.

Q-value is updated using bellman's equation:

$$Q_{new}(s_t, a_t) = (1 - \alpha)[Q_{old}(s_t, a_t)] + \alpha[R_t + argmax\,Q(s_{t+1}, a_t)]$$

act	d1410	d1430	d1490	u1600	d1380	d1420	d1390	u1530
42	0	0	0	0	0	0	0	0
43	0	0	0	0	0	0	0	0
24	-40.2237832598	-22.1384391695	-33.5278231335	-22.887311525	-84.6546743537	-164.100814697	-195.9558891216	-196.9551732011
25	-51.8976719625	-29.3262713771	-62.4626811067	6.8308899543	-58.954230805	-1.1387112271	-88.2946395139	-74.9648479386
26	19.6222978337	-9.2175201616	-10.6501295075	25.9216344586	-7.9113059766	33.0436401191	-14.7757448276	-7.6549558905
27	58.8539368904	66.1317137525	55.3702297504	72.3849845838	48.498213955	53.3829037459	60.5446605573	59.1864805542
21	0	0	0	0	0	0	0	0
22	293.5645821847	170.3878050473	88.8685301072	89.4632145869	161.0900376304	236.7545447935	235.5285340804	67.1308503009
23	-148.1320971127	-353.0628010288	-86.547073449	-78.1970950724	40.4930446841	35.5401389666	-353.9254553728	-343.9774597457
28	122.6084898438	115.1003711746	160.3910022637	111.0834968484	99.815886928	120.7860980329	117.6234869556	101.3520303356
29	345.2033361738	316.8462466723	328.3373787379	347.1286429029	320.2693531071	304.1814133124	348.6050646523	303.5044268614
40	0	0	0	0	0	0	0	0
41	0	0	0	0	0	709.4616238468	0	0
0	0	0	0	0	0	0	0	0
39	385.2000279627	633.893684034	487.199712303	0	584.8568372244	399.9999999266	639.9999999914	396.2221589005
38	139.7988347544	110.8216219921	137.3053328524	152.3090639992	117.6104858898	156.8680919169	153.3552776778	128.5736154652
33	63.8541050445	69.9851930897	50.0618723118	53.4657457483	64.5912428939	60.6423800207	60.5478866103	64.5026272731
32	141.6760545446	109.5286234689	154.0864841814	150.6761771986	122.7752646946	102.3025571634	129.0402382223	153.1083479009
31	279.282679873	337.3812395042	309.1597162562	283.7135775798	283.4753764064	337.9443513275	341.4355031724	341.0304008241
30	626.1863643588	603.7761621124	639.0712885837	626.5962693081	654.2348703231	654.7136868441	608.9471394869	684.8357336748
37	-241.4573636534	-219.0938042369	-226.9252899867	-227.5173870581	-221.9527647485	-231.897071839	-225.6787649393	-228.1410273934
36	-107.4799570389	-166.3861332397	-110.518157622	-128.2722253684	-164.2045644438	-160.821750139	86.9276457886	-162.7529276926
35	-35.6703505015	-53.1645260653	-55.210615968	-34.261240511	-49.4398567031	-66.3501346652	-53.8080498765	-21.9388919659
34	6.7964495768	16.3971488313	8.9834300022	15.6315752215	16.6696550035	3.6818155465	7.4284549663	18.8599608446

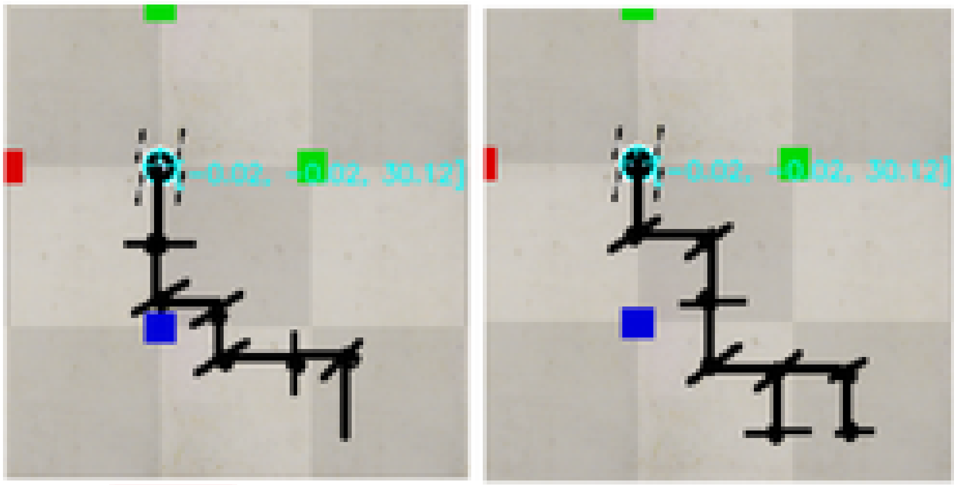
In Q-Learning, unlike SARSA, based on experiences the agent gets greedy gradually i.e. it always goes for immediate maximum reward.To maintain exploration and exploitation strategy, we use epsilon greedy policy. Random action is important because it allows the agent to explore and discover new states that otherwise may not be selected during the exploitation process.

Note: You can balance exploration/exploitation using epsilon and setting the value of how often you want to explore vs exploit.

- **Path planning and Navigation of drone on coordinates given by RL while avoiding the real world obstacles.**

  The regular trajectory planning algorithms just depend on the intial position, final position, intial velocity and final velocity.

  But in reinforcement learning, the agent takes random actions and visits most of the points in space and then finds obstacles and try to avoid them in next episodes by which it learns to go to a target point. The obstacles are placed in the real world environment which the agent(Drone) has to navigate and plan the path avoiding them.



  This figure above depicts that the drone was put at a particular starting coordinate let's say [-0.02,-0.02,30.12] it planned the path keeping obstacles in picture and navigating on that path by avoiding the obstacles placed in the real world and reached the desired setpoint.

1.1 Hardware Used

- Plutox Drone Drona-Aviation

1.2 Software used

1. Operating System

   - Version: Ubuntu 16.04 (LTS)
   - Download Link: http://releases.ubuntu.com/16.04/
   - Installation Instructions: Download the installation file from the link provided and follow these instruction to create a bootable USB. Later, follow these instruction instructions to install Ubuntu on you laptop.

   Note: This project can only be created using a Linux system, as ROS is only supported on linux. Therefore, this report will only contain instructions and steps for Linux systems hereon.

2. Robot Operating System (ROS)

   - Version: Kinetic Kame
   - Installation Instructions: Follow these instruction to install ROS Kinetic on both the laptop Follow these instruction to create a ROS workspace.

3. V-REP
   It is a Simulation Software used to verify results before applying to the real world.The robot simulator V-REP, with integrated development environment, is based on a distributed control architecture: each object/model can be individually controlled via an embedded script, a plugin, a ROS or BlueZero node, a remote API client, or a custom solution. This makes V-REP very versatile and ideal for multi-robot applications. Controllers can be written in C/C++, Python, Java, Lua, Matlab or Octave.

   - Installation instruction V-REP installation

1.3 Software and Code

https://github.com/eyantra-eysip/RL-Drone-2019
This project repository containts various pakages,scripts,launch files,Graphs etc

- Whycon
  this package contains all the required documentation along with the installation guide and how to put it into use for our project
- v-rep ros interface
  this package will help in the installation of the ROS package required to enable communication between your ROS nodes and V-REP simulator.
- plutodrone
  this package helps us control the pluto drone via service and topic.
- Scripts
  this folder in the repository contains the scripts of algorithms which we used in order to implement them in V-REP scene to train our model. Apart from that it contains the python script for the PID algorithm used to hold the drone so that it could navigate on the path it has planned based on Reinforcement Learning Algorithm.
- launch
  This provides convenient way to start up multiple nodes and a master, as well as other initialization requirements such as setting parameters.
- graphs
  these are the graphs plotted between the values given by the cordinate Axis of the whycon marker (pitch,roll,throttle,yaw) placed over the drone and the desired cordinate it has to reach.this also show does the comparative analysis of the PID and Reinforcement Learning.
- .ttt files
  these files are to be loaded as a scene in V-REP to run on the simulation plateform.

1.4 Use and Demo

1. Path Planning and Navigation https://www.youtube.com/watch?v=vB7WC2BcLSs
2. Position hold Reinforcement Learning https://www.youtube.com/watch?v=vF-H1EUbfEM
3. Demo

1.5 Future Work

– implementation of DQN and its training for better results.

– Oscillations to be reduced while holding the position using RL.

1.6 Bug report and Challenges

– The Hardware implementation of this project was tidious in itself.

– The challenge was to get the optimum values of PID parameters(Kp,Kd,Ki) in order to stablize the drone because any external factor e.g. Whether Change e.t.c may affect it.

– Due to Oscillation in holding the drone in one axis it was difficult to hold it in some other axis because of the ramdom action taken by RL algorithm.

1.7 References

1. David Silver- Reinforcement Learning video lectures
2. Research paper
3. Reinforcement Learning Strategy for UAV
4. OpenAI GYM
5. ROS tutorials
6. V-REP tutorials
7. Improving the Beginners PID Introduction
8. PID Control - A brief introduction
9. Hardware Demo of a Digital PID Controller