Deep RL Arm Manipulation Project

Pritesh Gudge

Abstract—The goal to the Deep RL Arm project is to create a DQN agent and describe reward functions to teach a robotic arm to carry out two primary objectives. First,to have any part of the robot arm touch the a designated object on the ground, with at least a 90% accuracy. Second, to have only the gripper base of the robot arm touch the designated object, with at least a 80% accuracy.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, DeepRL.

1 Introduction

THE This paradigm of learning by trial-and-error, solely from rewards or punishments, is known as reinforcement learning (RL).The artificial agents are built to learn for themselves to achieve successful strategies that lead to the greatest long-term rewards. [1].

2 Background

When a robot needs to be deployed to achieve a certain objective, the controls of the robot can be trained using the reinforcement learning algorithms.

2.1 Deep Q-Networks

The agent must continually make value judgements so as to select good actions over bad. This knowledge is represented by a Q-network that estimates the total reward that an agent can expect to receive after taking a particular action. Deep Q-Networks (DQN) algorithm stores all of the agent's experiences and then randomly samples and replays these experiences to provide diverse and decorrelated training data [1].

3 Simulations

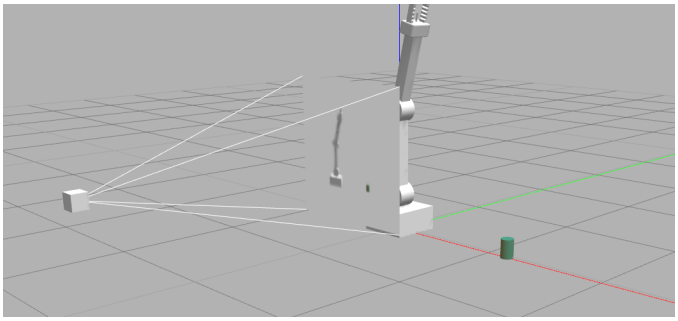

Fig. 1. Simulation setup

A simulation setup was provided by Udacity was used for the purpose of training the agent (Ref. 1). Camera and collision stream topics are published by the simulation setup which are used to setup the learning algorithm with appropriate reward and penalty function based on the actions taken by the agent. A DQN agent is used for the purpose and position-based control is used for the arm joints.

3.1 Arm Collision

The first part of the project was to achieve 90% accuracy for arm collision with the object in the simulation world.

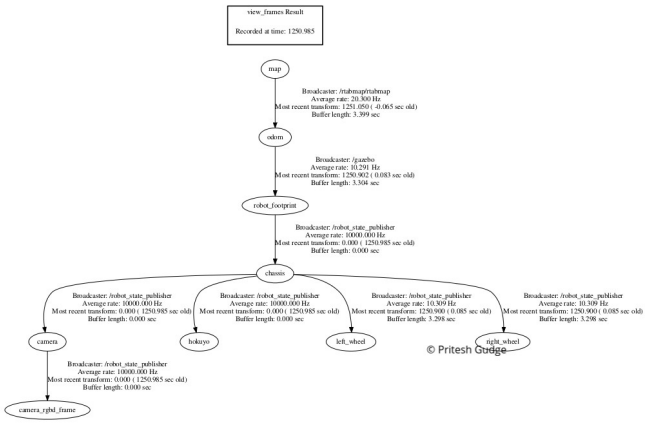

Fig. 2. TF Frame

3.2 Gripper Collision

The second part of the project was to achieve 80% for gripper collision with the object in the simulation world.

3.2.1 Gripper Collision

The Kitchen World model was mapped with RtabMap features "Kp/MaxFeatures" 400 and "Vis/MinInliers" at 15, and "Reg/Strategy" 0

The Custom World model was mapped with RtabMap features "Kp/MaxFeatures" 400 and "Vis/MinInliers" at 20, and "Reg/Strategy" 1

4 Results

4.1 Kitchen World

The kitchen world 2D and 3D maps are shown in figure5 and figure 6 respectively. In the 2D Map and the 3D map the general layout of the kitchen and the furniture structures are visible. A total of 198 loop closures were detected(Fig. 7).

Fig. 3. Kitchen World

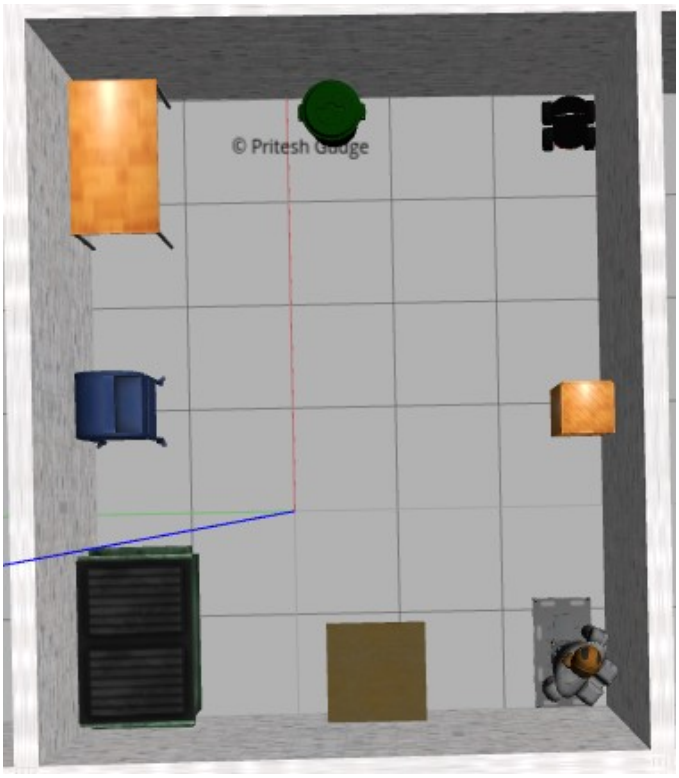

Fig. 4. Custom World

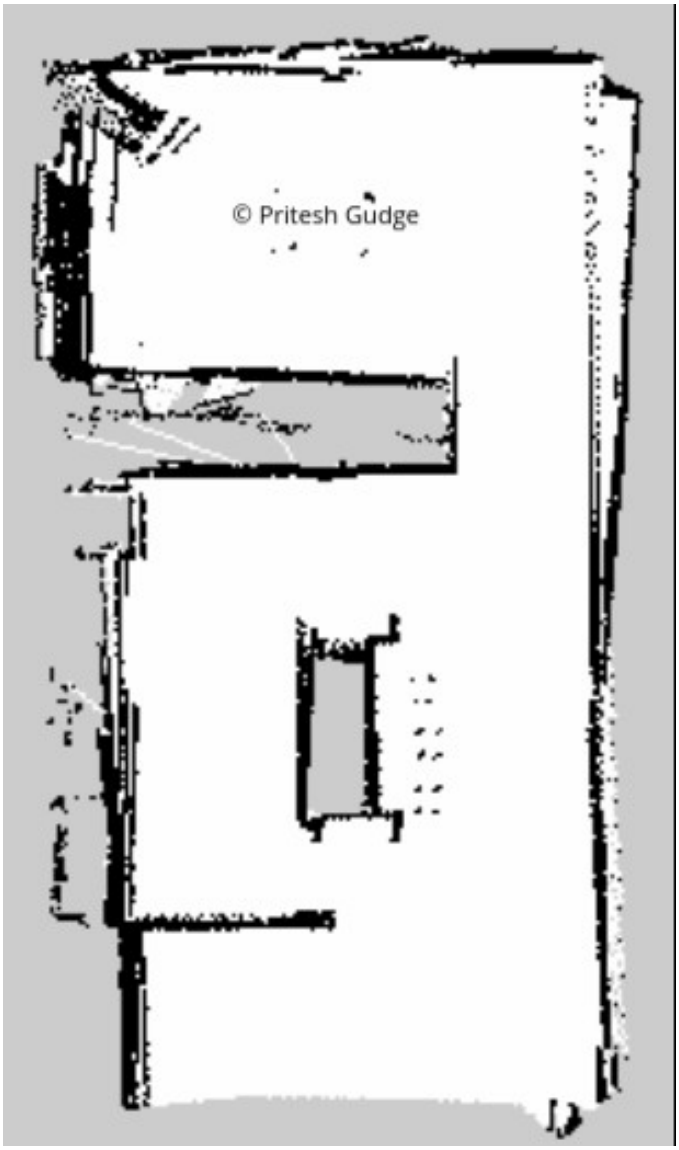

Fig. 5. Kitchen World 2D map

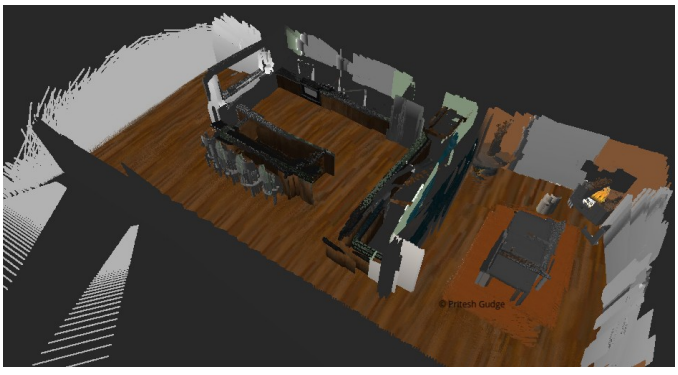

Fig. 6. Kitchen World 3D Map

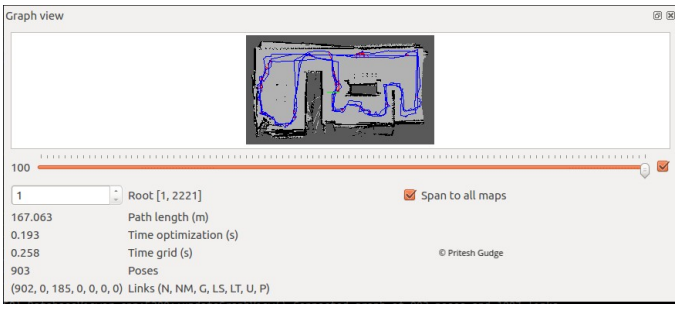

Fig. 7. Kitchen World Loop Closures

4.2 Custom World

The kitchen world 2D and 3D maps are shown in figure8 and figure 9 respectively. In the 2D Map and the 3D map the general layout of the floor and the furniture structures are visible. A total of 237 loop closures were found(Fig. 10).

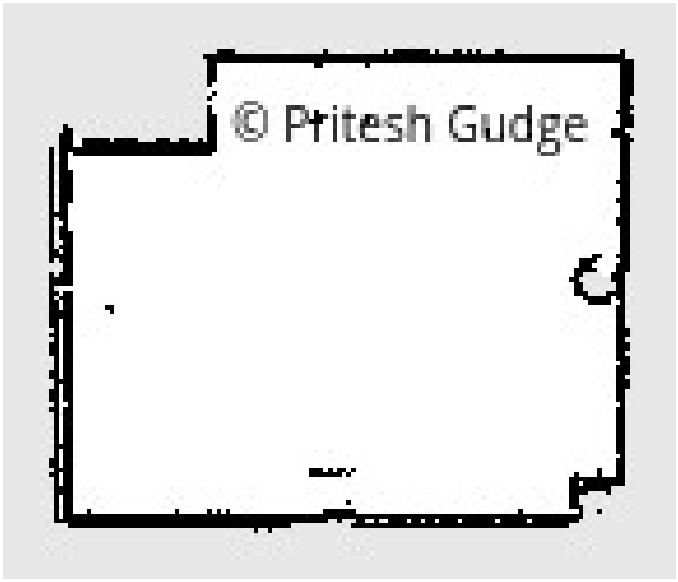

Fig. 8. Custom World 2D map

4.3 Technical Comparison

The kitchen_dining model performed significantly better then the student created cafe model. This was due to the richer and more complex features of the kitchen_dining model.

5 Discussion

The robot was teleoped (navigated via the keyboard) around the room. At some points the robot did not move forward. This appeared to be when it started to perform loop closure. The parameters were not changed from the default provided as satisfactory result was obtained with the same. However the 3D map quickly started to resemble the physical kitchen dining gazebo model. To improve loop detection rates some, on the spot circles were performed. Of particular note were the features in the main kitchen area. More SURF features were identified there as there was more variation in the surfaces.

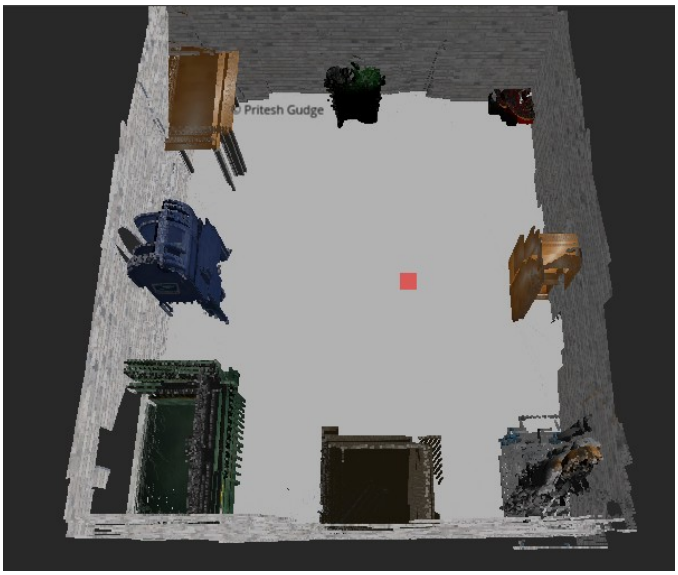

Fig. 9. Custom World 3D Map

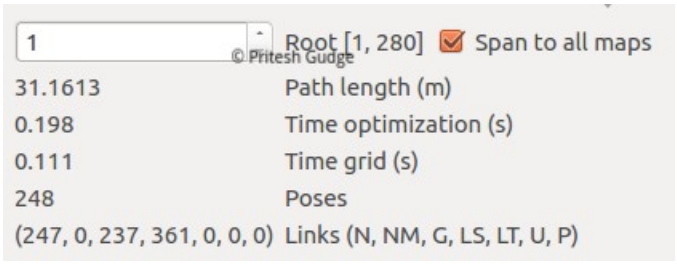

Fig. 10. Custom World Stats

For the custom layout Vis/MinInliers was increased from 15 to 20 and Reg/Strategy was changed to 1(Including ICP). As the size of the environment was small and to prevent repeated and incorrect loop closures the above values of parameters were chosen.

The custom layout gazebo model wall surfaces were tiled, repeatable pattern with lack of other discerning features sometimes caused the loop closure detection to map to an incorrect previous image. This then distorted the map. Additional features were added to achieve a successful map.

6 Conclusion / Future work

Mapping is important to help understand the world. There are a plethora of sensors and of interest is the about to arrive solid state lidars. As the price point of these sensors continues to drop it will open up opportunities to create richer and more realistic 3D maps at a cheaper price point. Being able to map an environment cost effectively to create a replicated virtual world will increasingly be important to allow for the training of deep learning models. We are actively looking to do this and then supplant the trained model back into a robot so it can navigate in the original environment that was mapped.

6.1 Modifications for Improvement

Examples:

- Adding more sensors to collect data e.g. 3D Lidar
- Modifying the sensor location and layout. e.g. Adding RGBD camera's to the sides of the robot, to collect data about the lateral surroundings.

6.2 Hardware Deployment

1) To be deployed on hardware, a computation unit and appropriate sensors will have to be installed on the robot chassis and the wheels. Proper calibration and verification needs to be done to limit the errors.
2) High CPU and RAM hardware is required to perform SLAM data collection from the multiple sensors, especially to detect loop closures online.

REFERENCES

[1] D. Silver, "Deep reinforcement learning." https://deepmind.com/blog/deep-reinforcement-learning/.
[2] Multiple, "Occupancy grid mapping." https://en.wikipedia.org/wiki/Occupancy_grid_mapping.
[3] C. Stachniss, "Grid slam." http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/pdf/slam13-gridfastslam.pdf.
[4] Multiple, "Graph slam." https://en.wikipedia.org/wiki/GraphSLAM.
[5] M. Labbe, "Rtab-map's ros-pkg." http://wiki.ros.org/rtabmap_ros.
[6] Multiple, "Graph slam." https://en.wikipedia.org/wiki/Bag-of-words_model_in_computer_vision.
[7] A. M. . A. K., "Introduction to surf." http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html.