Applied Reinforcement Learning
SS 2019

Project Intermediate Report — Obstacle Avoidance

Akbar, Uzair	Gündogan Alperen	Ellouze, Rachid
3697290	3694565	3671114

June 06, 2019

Abstract

This report presents our progress for the TUM Applied Reinforcement Learning class project (Summer Semester 2019). We have successfully completed a simple implementation of the Q-learning algorithm for obstacle avoidance using LIDAR ranging data. The agent is tested in a Stage simulator environment and selected preliminary results are presented herein.

Contents

1 Introduction & Project Objectives **1**

2 Simulation Environment Setup **1**

3 The RL Agent **2**
- 3.1 State Space Definition 2
- 3.2 Action Space Definition 2
- 3.3 The Reward Function 2

4 Preliminary Results **3**
- 4.1 General Training Configuration 3
- 4.2 Experiments 3

5 Future Works **4**

1 Introduction & Project Objectives

We start off by quickly reiterating our problem statement and solution objectives. We intend to implement a solution for the obstacle avoidance problem for a turtlebot robot using a LIDAR sensor. The two metrics (which we shall try to quantify in the form of a reward function in section 3.3) to evaluate the quality of our solution are:

- The turtlebot covers maximum distance within an episode.
- The turtlebot avoids collisions.

2 Simulation Environment Setup

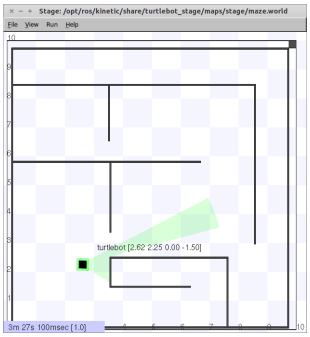


Figure 1: Turtlebot in the Stage simulator environment.

As discussed in our project proposal report, we chose to implement our Reinforcement Learning (RL) based obstacle avoidance solution first in a simulation environment, mainly for quick prototyping purposes. We shall later generalize to the real world using prospective approaches as listed in the proposal report. However, this report is primarily concerned with simulation-only implementation of our solution.

We used the Stage simulator to set-up or training environment as shown in Figure 1. Because the LIDAR sensor that is provided on the turtlebot does ranging estimates in a horizontal, 2D-plane, it was preferable to use a 2D simulator such as Stage so as to reduce unnecessary computational overhead.

The model of our environment, the reinforcement-agent and the Q-learning algorithm were implemented using the Python programming language. We used the rospy client API to interface our Python scripts with the Robot Operating System (ROS). The main motivation behind the use of Python is ease of implementation.

Upon starting the turtlebot Stage simulator, we run the `rl_agent` node*, which interacts with the simulator environment. Our RL agent subsequently starts receiving LIDAR ranging data at every time-step (set to $\delta t = 0.14$ sec.)

over 360° with a resolution of 1°. This data is pre-processed (details discussed in section 3.1) and is used to model our problem as an MDP to be solved by our RL-agent. The control signals of the RL-agent are published to the topic /cmd_vel_mux/input/teleop which moves the turtlebot in the simulation accordingly.

3 The RL Agent

We implemented our MDP as described in our project proposal report. However, we restate some of the details here because they have either been changed slightly or were not as clearly communicated in the project proposal report.

3.1 State Space Definition

Similar to our proposal, we define the state-space as $\mathcal{S} := \{s \mid s = [\, d^T\,,\, v\,,\, w\,]^T\}$, where d is the depth vector observed from the on-board LIDAR sensor. To leverage temporal information, v and w are linear and angular velocities of the robot respectively and are discretized as

$$v \in V := \{0.4, 0.2\,\mathrm{m\,s^{-1}}\}, \quad w \in W := \{\frac{\pi}{6}, \frac{\pi}{12}, 0, -\frac{\pi}{12}, -\frac{\pi}{6}\,\mathrm{rad\,s^{-1}}\}.$$

Similarly, we also discretize d into angular sectors as well as radially (details in section 4), to achieve a final vector of size of $5 \times 6 = 36$ (unless stated otherwise), enumeration over all possible values of which gives 5^6. Hence, in total, our state space consists of $5^6 * 2 * 5 = 156250$ discrete states.

3.2 Action Space Definition

The control space $\mathcal{A}$ shall determine the permissible linear and angular velocities for the turtlebot. We explored two possible discretizations of the action space:

1. $\mathcal{A}_1 := \Big\{a \;\Big|\; a = \big(v', w'\big),\ v' \in V,\ w' \in W\Big\}$, where v' and w' are next state linear and angular velocities of the robot respectively. Hence, $|\mathcal{A}_1| = 10$. This simply means that the current action determines what the velocity part of the next state should be.

2. A simpler/smaller action space where v' and w' are "decoupled," — $\mathcal{A}_2 := \{a \mid a \in \mathcal{A}_1\,,\ v' = v \text{ or } w' = w\}$, with $|\mathcal{A}_2| = 7$.

3.3 The Reward Function

Our task for the turtlebot is for it to go as fast as possible without any collisions. For this, we can employ a simple, action agnostic stage-wise reward function

$$r(s) = \begin{cases} v \cdot \cos(w) \cdot \delta t & \text{if episode running,} \\ -10 & \text{if collision.} \end{cases}$$

The cosine term penalizes meaningless rotation, and δt is the time-step for our turtlebot (set to 0.14 s).

4 Preliminary Results

4.1 General Training Configuration

The following setting should be taken as default in all of our experiments unless stated otherwise for comparitive study purposes.

- **State Space:** The LIDAR sensor delivers readings over 360° degrees (1° precision) of range measurements up to 6.5 meters. The original LIDAR space is then $[0°, 360°) \times [0, 6.5m[$. We have discretized the two dimensions separately. Range values are divided radially into 5 levels — $[0, 0.5[$, $[0.5, 1[$, $[1, 1.5[$, $[1.5, 2.5[$ and $[2.5, 6.5[$. The intuition behind higher resolution in near-range regions is that it is very critical to select an action when the robot is close to an obstacle. For angles, we limited ourselves to $[-60°, 60°]$ interval for now, which is divided 6 equal sectors.

- **Action Space:** Both, linear and angular velocities are not obtained directly from the environment. They are read directly from the previously taken action, as defined by action space $\mathcal{A}_2$. This presents a way to encode the previous states, as the MDP do not allow history dependent policies.

- **RL Algorithm:** We use simple Q-learning with eligibility traces and ϵ-greedy behaviour policy and enumerate over the states using a look-up table.

4.2 Experiments

In order to find (or validate) reasonable values for the above mentioned default configurations, we conducted the preliminary following experiments, all of which have been performed 5 times over 250 episodes and the results are aggregated and plotted in Figure 2 as mean and standard-deviation of the cumulative rewards of that respective episode.

- **State Space:** We compared the default LIDAR discretization strategy with lower radial discretization of only the 3 lower levels and compared the cumulative rewards per episode in Figure 2 Part (a). Similarly, we compared the sector-wise angular discretization with 3 and 10 equal radial segmentation as shown in Figure 2 Part (b).

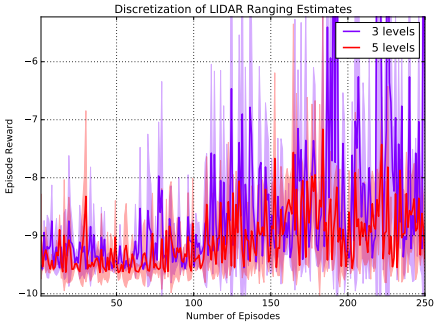
- **Action Space:** Both, $\mathcal{A}_1$ and $\mathcal{A}_2$ are compare in Figure 2 Part (d), and, as expected, $\mathcal{A}_2$ outperforms $\mathcal{A}_1$ on at least the rate of learning (the quality of solution for higher epochs remain to be seen).

- **Eligibility Traces:** For completeness purposes (and more for debugging purposes), we compared the results of eligibility traces implementation

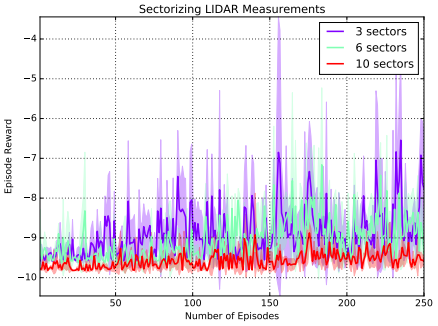
with that of no eligibility traces, and as expected got better performance for the former.

5 Future Works

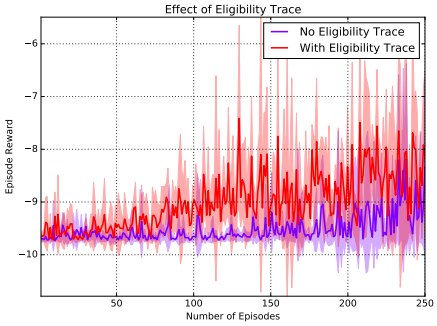
- Test the implemented algorithms in real environment and tune the paramters to improve the results.
- Implement better feature extraction method e.g. autoencoder.
- Implement SARSA with LVFA on ϵ - greedy policy.
- Implement better action space and reward signals.
- Implement and compare sof-tmax policy with ϵ-greedy.
- Determine the final approach and parameters to be used in the real turtlebot.



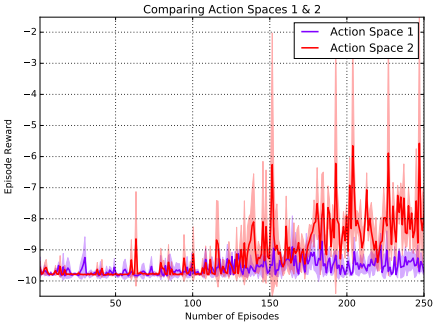
(a) Radial resolution of LIDAR readings.



(b) Angular resolution of LIDAR readings.



(c) Eligibility traces vs. no eligibility traces.



(d) Comparing action spaces $\mathcal{A}_1$ & $\mathcal{A}_2$.

Figure 2: Preliminary results for comparison of different training settings.