Learning a Wall Following Behavior using Sarsa(λ) Reinforcement Learning

John Hoare
Distributed Intelligence Laboratory, Department of Electrical Engineering and Computer Science
The University of Tennessee, Knoxville, TN 37996-3450
Email: jhoare@eecs.utk.edu

***Abstract*— In this work a simulated robot learns to follow a wall by using the reinforcement learning technique Sarsa(λ). The robot uses a 180 degree laser scanner mounted at 45 degrees to the left, which the robot uses to follow the left-hand side wall. The robot successfully learns a control that enables it to follow the wall on the left side. Presented is the discritation method for the state-space, as well as the situations presented to the robot in order to have it learn.**

I. Introduction

In this paper work is presented that enables a robot simulated in the stage environment to follow walls along its left side. The robot learns this behavior using the Sarsa(λ) reinforcement learning algorithm.

A. Problem Formulation

For this work, the robot is assumed to begin along-side a wall, and must continue to follow the wall. The robot will not start in free-space and "search" for the wall. The desired distance from a wall is denoted with δ_w which is set to 0.75 meters. Additionally, the robot is considered to be within this distance if it is within $\epsilon_w = 0.25$ meters of δ_w.

II. Approach

Reinforcement learning relies on learning the optimum action given a specific state. In this section a definition of how the current state is determined, and what the available actions are.

A. Robot

The robot is a pioneer 3dx robot, with a mounted sick lms200 laser range finder, mounted 45°to the left. The robot has other sensors also available to it, however only the laser range finder is used for this work.

B. States

The laser range finder provides far too much information directly to have a discrete state space. Because of this the laser is used to create a much smaller, discrete state space. While, the laser returns a range to an obstacle, this needs to be much more simple for a discrete state space. Because of this, we convert the range readings to discrete measurements:

$$\text{d-range}(r) = \begin{cases} \text{LOW} & \text{if } r < \delta_w - \epsilon_w \\ \text{GOOD} & \text{if } (\delta_w - \epsilon_w) \geq r \geq (\delta_w + \epsilon_w) \\ \text{HIGH} & \text{if } r > \delta_w + \epsilon_w \end{cases} \tag{1}$$

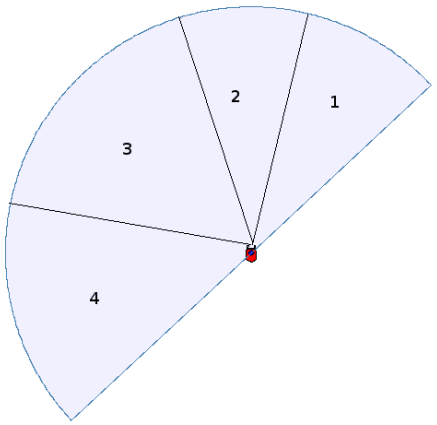

Fig. 1. The configuration of the robot and a graphical representation of how the laser is split up. Region 4 is the so-called "goal" region, that is the region that the robot tries to keep a distance of δ_w away.

The laser is cut into four regions total, first dividing the laser into thirds, and then sub-dividing the frontward-facing third again into half, where the minimal range value from each region is used as the representative laser range. From this, we create a four-tuple that represents the current state.

$$S_{\text{region}} = \min_{i \in \text{region}} \text{d-range}(r_i) \ \forall \ \text{region} \tag{2}$$

Because there are four regions, and three possible ranges (values) for each region, there is a total of $3^4 = 81$ states. The duration of a state is 5 player robot.Read() cycles, which occur at 20Hz. This results in a state lasting approximately $\frac{1}{4}$ of a second.

C. Actions

To keep the learning task more tractable, a minimal set of actions are defined. The robot has a turn left/right action as well as a drive forward action. The turn left and turn right actions both have a very minor forward component, to be able to move the robot out of oscillatory behavior. With 81 states, and 3 actions, the size of Q becomes 81×3.

D. Reward Function

The reward function is defined as a function of both the state and action. The state has a "reward" region, which is the region that the robot is to try to keep within the GOOD

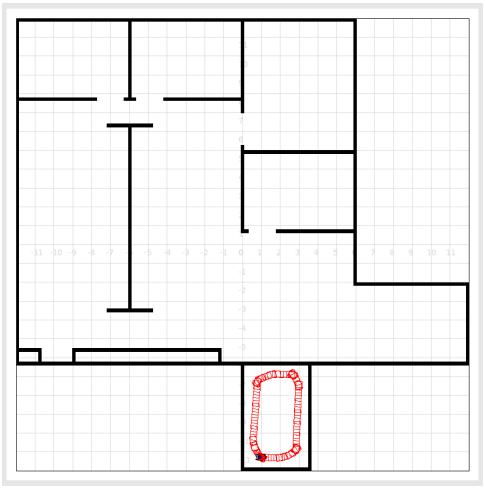

Fig. 2. The robot first learns to follow the wall in a small, simple environment. Results shown are after 600 epsiodes.

TABLE I
CONSTANT VALUES USED IN SARSA(λ)

Constant	Value
λ	0.9
γ	0.1
α	0.1

distance. The reward function gives the maximum reward, if the robot is within that state, and is executing the "move forward" action. The reward function is explicitly defined as:

$$\text{reward}(s,a) = \begin{cases} -10 & \text{if stalled} \\ -1 & \text{if } \exists \text{ LOW} \in s \\ & \text{if } s_{\text{reward}} = \text{GOOD} \\ 2 & \& \ a = \text{FORWARD} \\ 0.25 & \text{if } s_{\text{reward}} = \text{GOOD} \\ -1 & \text{if left the reward region} \end{cases} \quad (3)$$

E. Constants

Provided in Table I are the constants used in the learning activity. ϵ is different depending on which stage of training currently in (see next section) and decays by multiplying the last epsilon by 0.999 at the end of each episode.

F. Training Method

Training is done by starting the robot next to the wall about δ_w distance away, and an episode lasts as long as the robot travels without hitting a wall, or wandering away from the wall. At first, training is done in an epsilon-greedy fashion, where the robot is in a small, enclosed room shown in figure 4. The robot trains within the small room for approximately 4 hours, or 600 episodes. Once the robot is able to wall-follow in the small room the next stage of training begins. The robot is put into the normal environment, and trains for several more hours, using a strictly greedy approach. During this time, an epoch will last for much longer, as the robot is much more close to an optimum policy, and will make mistakes much more rarely. As time pasts, the robot learns to travel through the entire environment, avoiding obstacles. The robot experiences 48 more episodes over about 18 hours of training, to learn an effective policy. Of course the exact times and number of episodes will vary due to randomness, but the reported times are given as a guide.

III. EVALUATION

As the robot was trained as explained in the previous section, the state was stored at every 50 training episodes. From the states, we can plot the learned reward function, which can visualize how the robot is able to learn the optimum policy over time. The learned policy is also tested in a less structured, "hospital" environment, to see how well of a policy is learned for more complicated worlds.

IV. RESULTS

After 648 training episodes (about 22 hours), the robot is able to follow the wall without collisions as shown in Figure 5. While the robot goes through the hallway at the top of the map, it exhibits oscillatory behavior, but is however able to overcome this due to the slight forward action that is in the turn left/right methods. The learned value function is shown in figure 3. In the value function any state that is 0 is either no reward, or more likely an unvisited state. It is likely that there are many "useless" states that are not necessary when the robot starts next to the wall, that the robot does not encounter in the more simple environment, and as we can see from Figure 5, the robot is still very much able to preform well without knowledge of these states.

V. CONCLUSIONS AND FUTURE WORK

The robot has successfully learned to follow the wall by using the Sarsa(λ) reinforcement learning technique. This work has presented a method to iteratively train the robot to handle more and more complicated environments, in order to allow the robot to more slowly encounter new things. This method gives the robot a more likely chance to be able to succeed or to learn how to handle the new abnormality in the environment. The forward most region was divided into two (from the original 3 regions to 4 regions) to try to deal with the oscillatory behavior exhibited in the hallway. The oscillatory behavior is lessened by the increased state space but is still there, it is likely that further subdivisions may help, however it will also likely cause the learning time to go higher. Also, changing the reward function so the robot gets no reward while turning may also help stop this oscillatory behavior. In the hospital environment, the robot encounters many of the aforementioned "0" states, and the robot is unable to pick a good action. Future work would be to develop a way for the robot to encounter these states and be able to do exploration only at these points. Classical ϵ-greedy methods would not allow the robot to get far enough. A likely good solution, would be to have a step where the robot wanders around using its learned policy, and save the position of the robot a few seconds before failing (impacting a wall), and use this to generate initial positions for the robot to then follow an ϵ-greedy approach to learn how to handle these new instances.

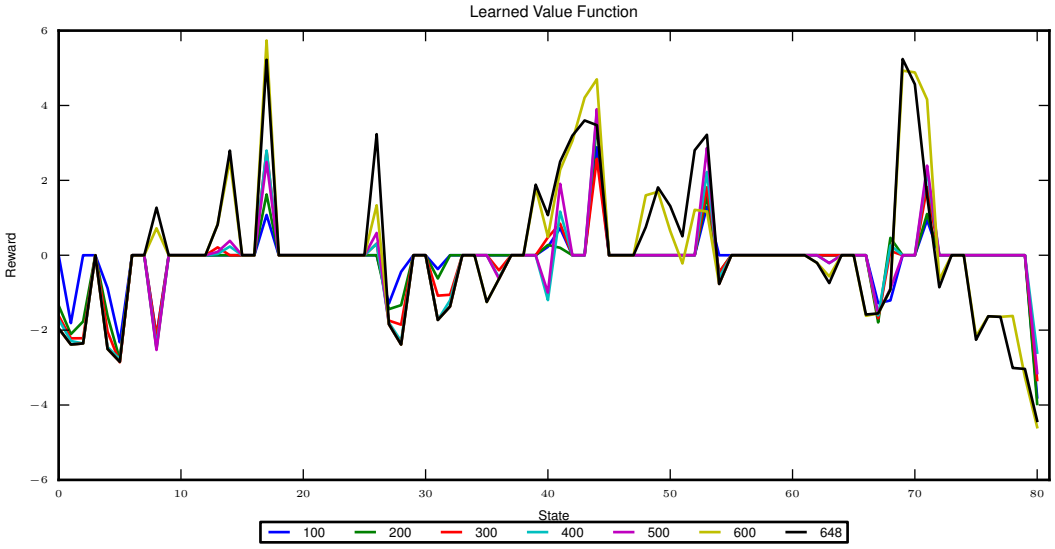

Fig. 3. The learned value function, after different levels of episodes.

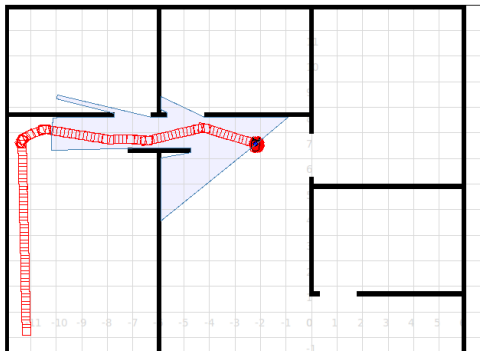

Fig. 4. The robot moving in the full environment, after it has only been experimenting within the smaller environment. The robot is unable to sucessfully follow the wall through the full envionment, and fails to enter the two top rooms. Results shown are after 600 epsiodes.

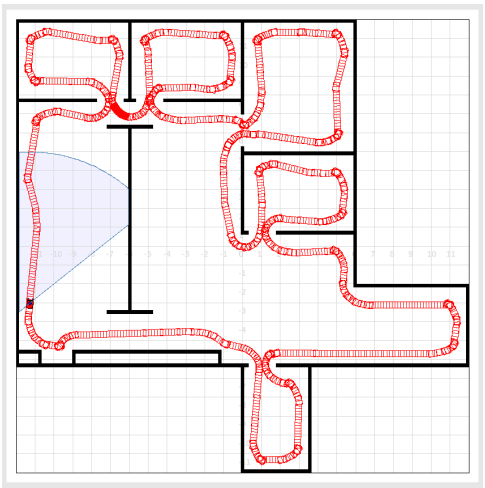

Fig. 5. The robot moving in the full environment, after it has learned its final policy. The robot is able to successfully navigate through the environment, entering any rooms it goes by. In the narrow hallway above, the robot participates oscillatory behavior, which causes it to move very slowly through this area.

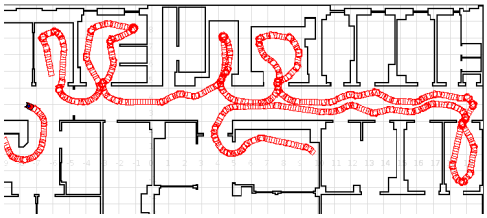

Fig. 6. The robot moving in the more complex, "hospital" enviornment, after it has learned its final policy from the other environment. The robot is able to usually wall-follow successfully, however it us unable to enter the small doors, and will sometimes collide with the walls in some of the more complicated rooms.