CS5478 Assignment4

Shicheng Chen A0215003A

April 2020

1 Problem 1

1.1 1

Let us first consider that w_l and w_r in same direction and $w_l < w_r$.

As shown in Figure 1, $R_1 + L = R_2, R_1\theta = w_l rt, R_2\theta = w_r rt$. Then $R_1 = \frac{w_l L}{w_r - w_l}, w_p = \frac{r(w_r - w_l)}{L}, V_p = \frac{r(w_l + w_r)}{2}, l_p = \frac{L(w_r + w_l)}{2(w_r - w_l)}$

For the second situation, w_l and w_r are in the opposite direction and $w_l < w_r$.

$R_1 + R_2 = L, R_1\theta = w_l rt, R_2\theta = w_r rt$. Then $R_1 = \frac{w_l L}{w_r + w_l}, w_p = \frac{r(w_r + w_l)}{L}, V_p = \frac{r(w_r - w_l)}{2}$.

If w_l, w_r have directions, we can combine these two situations together. $w_p = \frac{r(w_r - w_l)}{L}, V_p = \frac{r(w_l + w_r)}{2}$

1.2 2

For the rotation of the robot, $\triangle\theta = w_p \triangle t$, $l_p = \frac{L(w_r + w_l)}{2(w_r - w_l)}$.

For the transition of the robot, as shown in Fig 2, where $\theta_0 = \theta - \frac{\pi}{2}$, $\theta_1 = \theta_0 + \triangle\theta$

$$\theta^{'} = (\theta + \triangle\theta)\%(2\pi)$$

$$x^{'} = x - l_p \cos\theta_0 + l_p \cos\theta_1 = x - l_p(\cos(\theta - \frac{\pi}{2}) + \cos(\theta - \frac{\pi}{2} + w_p\triangle t))$$

$$y^{'} = y - l_p \sin\theta_0 + l_p \sin\theta_1 = y - l_p(\sin(\theta - \frac{\pi}{2}) + \sin(\theta - \frac{\pi}{2} + w_p\triangle t))$$

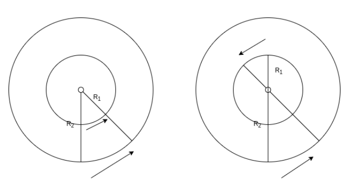

Figure 1

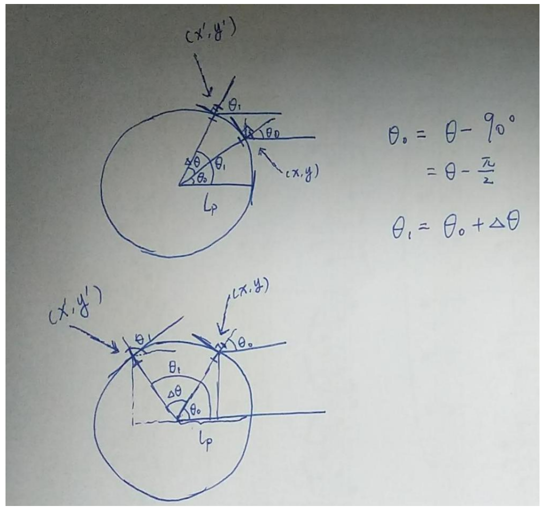

Figure 2

2 Problem 2

2.1 1

I choose ultrasonic sensors. Both LIDAR and ultrasonic sensors can measure the depth information. LIDAR achieves centimeter accuracy at a distance of over ten meters. However, we do not need such high accuracy for long distances since we use an indoor UVA. I prefer Ultrasonic because it is smaller, lighter, and much cheaper than LIDAR.

The multi-cameras system can also get 3D points from the 3D reconstruction. However, 3D reconstruction suffers from similarity ambiguity, so we also need extra devices to remove the ambiguity, such as accelerometers. Therefore, we do not choose cameras.

3 Problem 3

3.1 1

First of all, we assume that there is no negative loop path in the graph. Otherwise, our solution will be negative infinity.

1. State s_i represents the $node_i$. $V(s_i)$ means the cost of the path from the goal to the $node_i$.

2. If there is a valid weight edge from $node_i$ to $node_j$, then there will be a valid action from $node_i$ to $node_j$.

3. $Reward(s, a)$ is only depend on s and $s^{'}$. We can get $s^{'}$ from s and a. If the weight of the edge from s to $s^{'}$ is r, $Reward(s, a)$ will be $-r$.

4. state-transition. If there is a valid action from $node_i$ to $node_j$, then the agent can transfer from $node_i$ to $node_j$.

For finite-horizon $V = E[\Sigma_{t=0}^{N} R(s_t, a)]$.
For infinite-horizon, count factor $\gamma < 1$ such as 0.99.

3.2 2

We start from $V(s_{start})$(start node). By $a = argmax_{a \in A}(R(s_t, a) + V(s_{t+1}))$, we can find the next node by choosing the optimal action. Following the above function, we can get the minimum-cost path to the goal.

4 Problem 4

4.1 1

We set $R(s, a) = \Sigma_{s^{,} \in S} T(s, a, s^{,}) R(s, a, s^{,})$, where $T(s, a, s^{,})$ is the probability of transition from state s to state $s^{,}$ given action a. We initialize $V_0(s) = 0$.

$$V_t(s) = max_{a \in A}(R(s, a) + \Sigma_{s^{'} \in S} T(s, a, s^{'}) V_{t-1}(s^{'}))$$

for $t = 1, 2, 3...N$.

4.2 2

Yes, we compress three dimensional data $R(s, a, s^{,})$ to two dimensional data $R(s, a)$ and loss the information in the third dimension.

5 Problem 5

5.1 1

1. $V_1(1) = V_1(4) = V_1(5) = V_1(3) = 0, V_1(2) = 10, V_1(6) = 10$

2. $V_2(4) = V_2(3) = 0, V_2(1) = V_2(5) = 8, V_2(2) = 10, V_2(6) = 14$

3. $V_3(3) = 0, V_3(4) = 6.4, V_3(1) = 8, V_3(5) = 11.2, V_3(2) = 10, V_3(6) = 15.6$

5.2 2

$V^*(s_6) = 0.5 \times 20 \times \Sigma_{i \in [0,\infty)}(0.8 * 0.5)^i = 16.6667$.

5.3 3

$\pi^*(s_1) = RIGHT, \pi^*(s_4) = RIGHT, \pi^*(s_2) = DOWN, \pi^*(s_5) = RIGHT, \pi^*(s_6) = UP, \pi^*(s_3) = STAY$

5.4 4

$p_0 = 0.08602150537634408$. If $V^*(s_6) = p_0 20\Sigma_{i\in[0,\infty)}(0.8(1-p_0)^i) = 6.4$. If the agent go to s_3 from s_6 via s_5, s_2, $V^*(s_6) = 6.4$. If the probability of success p less than p_0, the agent from s_6 will go to s_3 via s_5, s_2.

$V^*(s_6) = p_0 16\Sigma_{i\in[0,\infty)}((1-p_0)^i) = \frac{100a}{1+4a}$, $\frac{dV^*(s_6)}{dp_0} = \frac{100}{(1+4p_0)^2} > 0$, so $V^*(s_6)$ increases monotonically. From $V^*(s_6) = 6.4$, we can get $p_0 = 0.08602150537634408$.

6 Problem 6

6.1 1

$$T = \begin{pmatrix} 0.1 & 0.4 & 0.5 \\ 0.4 & 0 & 0.6 \\ 0 & 0.6 & 0.4 \end{pmatrix}$$

$$p_0 = [1, 0, 0]$$

$$p_3 = p_0 T^3 = [0.153, 0.362, 0.485]$$

6.2 2

$$p_1 = p_0 T = [0.1, 0.4, 0.5]$$

$$p = (s|z) = \frac{p(z|s)p(s)}{p(z)} = \eta[0.1, 0.4, 0.5] \otimes [0.2, 0.6, 0.2] = [0.055, 0.667, 0.277]$$

Therefore, $s_2 = 0.667$.

7 Problem 7

7.1 1

$\frac{1}{2}$ for two particles representing two distinct locations. $\frac{1}{2}$ for representing the same location.

7.2 2

0.5^N for two particles representing two distinct locations after N steps. $1 - 0.5^N$ for representing the same location.

7.3 3

When two particles stay in the same location, they cannot go to two distinct locations again. However, two particles with two distinct locations always have chances to come to the same location.

7.4 4

Yes, this way can relieve the situation. However, the variance of the weights increases at every step, even for a large number of particles. After running a very long time, all particles will still converge to a single point.

7.5 5

Every period of time, instead of using sequential importance resampling(SIR), we can sample uniformly without any existing prior knowledge, which means that we reset our system.

8 Problem 8

8.1 1

I choose the particle filter. For a 2D environment, the size of the state space is moderate. I prefer to use the particle filter since it is more efficient and robust than the histogram filter. I do not choose Kalman filter since particle filter is more robust than Kalman filter for complex, multi-modal distributions.

8.2 2

I choose the Kalman filter. Kalman filter is good at tracking, and enemy aircraft can be represented by a linear function with Gaussian noise. We need a real-time tracking system to track and attack the aircraft, so we need the filter to be very fast, and the Kalman filter is suitable than particle filter and histogram filter.

9 Problem 9

9.1 1

1. State, the probability of tiger in the left door is b, in the other door is 1-b. We can set b to 0.5 for the initial state.
2. Actions: open the left door($a = 0$), open the right door($a = 1$), or just listen($a = 2$).
3. Observations: only when we choose the action, listening, we can get an observation. Otherwise, there is no observation.

4. Observation function, we can set $z = 0$ if the listening action result tells the tiger is behind the left door. We set $z = 1$ for the other door.

5. state-transition function. If we open a door, we will get a reward, and the game will reset. If we choose to listen and the observation for the first state is $z_0 = 0$, we can use Bayes' rule $p(s_0|z_0 = 0) = \eta p(z_0 = 0|s_0) * p(s_0) = 0.85$. If we choose listen again and the observation is $z = 0$, $p(s_1|z_1 = 0) = \eta p(z_1 = 0|s_1) * p(s_1) = 0.969$.

6. reward function: the reward for opening a door which leads to a tiger, opening the other door, or just listening are -100, 10, -1 respectively. $R(b, a = 0) = -100b + 10(1 - b)$,$R(b, a = 1) = -100(1 - b) + 10b$,$R(b, a = 2) = -1$, where b is the probability of a tiger hiding behind the left door.

10 Problem 10

10.1 1

I choose HMM. We have no observations in this problem, and we cannot get a reward function(otherwise, we can infer the observation from the reward function). Both POMDP and MDP need a reward function. HMM and POMDP can have initial state distribution, but MDP does not have this ability. For HMM, we can only make use of a probabilistic motion model but do not offer extra observation information.

10.2 2

Actions U, D, R, and L represent going up, down, right, and left, respectively. $U \times 3$ means going up for three times continuously. Our action list is $U \times 3, L \times 2, D \times 3, R \times 4, L \times 2$. After these actions, the robot will reach the T and stay there.

Let us dive into the action list. First we execute actions $U \times 3, L \times 2$. If the robot is in C, the robot will be at T. If the robot is in A or B, the robot will not move because of obstacles. Then we execute actions $D \times 3, R \times 4$, the robot will stay in the middle of the B and C no matter where the robot starts. Lastly, we execute actions $L \times 2$, the robot will reach the goal and stay there.