山东大学

崇新学堂

2025－2026 学年第一学期

实 验 报 告

课程名称： 电子信息工程导论

实验名称： Robots in Hallways

专 业 班 级 崇新学堂

学 生 姓 名 高子轩，钱竹玉，吕思洁，徐亚骐

实 验 时 间 2025 年 12 月 4 日

Hallway World

In this section, we became familiar with the colors, probabilities, and other aspects of each room in the software-generated map when the small car moves through the corridor.

Check Yourself 1. Move the robot around in the perfect simulator

Our operating steps:

Keep going to the right, that is, input 1. The picture is shown below, and the running order is

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

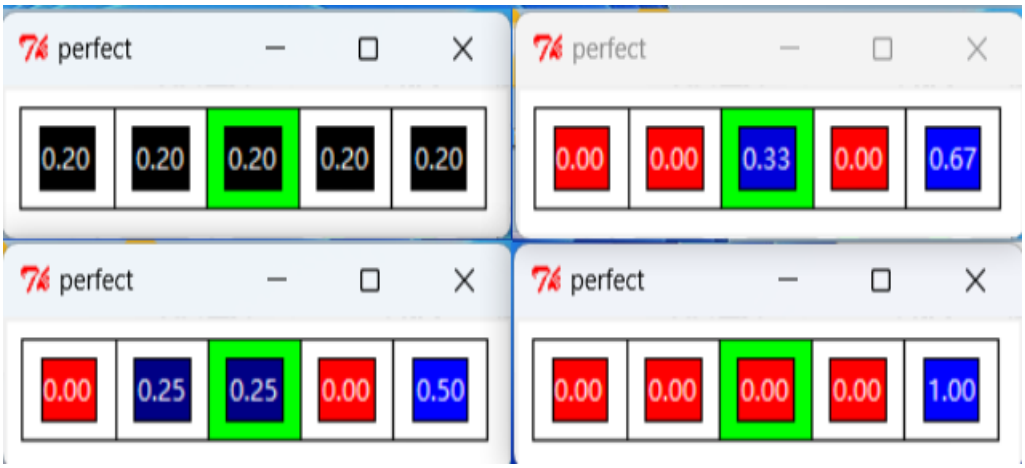

Figure 1 graphs under perfect conditions

Result Explanation:

Since the perfect simulator is completely accurate and without interference, the overall probability distribution will definitely shift to the right when moving to the right.

From a god's-eye view, we can see that the car initially is in the fourth room. It observes white, so it definitely did not start from the middle room, meaning that the probability for the fourth cell in the second image is 0.

After moving one step to the right and observing white again, this is still 0, and will not be elaborated further.

Check Yourself 2. Move the robot around in the noisy simulator.

Our first run steps:

Stay still, i.e., enter 0. The image is shown below, with the execution order as follows:

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

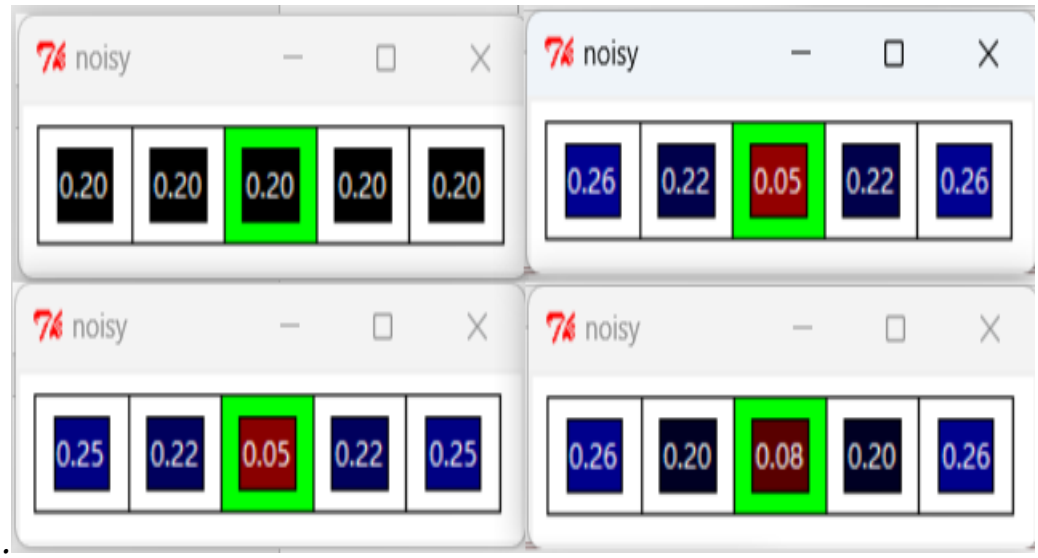

Figure 2 graphs under noisy conditions

Result interpretation:

Since we remained stationary, the car detected the white color, indicating a low probability of being in the green room. While inputting 0 might cause some deviation, this scenario didn't occur.

Our second run steps:

Keep going to the right, that is, input 1. The picture is shown below, and the running order is

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

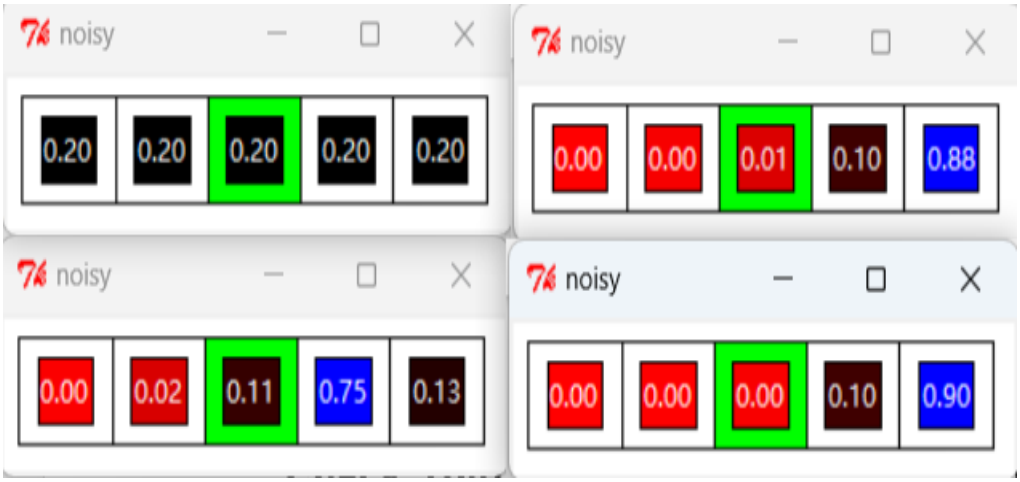

Figure 3 graphs under noisy conditions

Result Interpretation:

Moving to the right with probability is easy to understand. However, we found that no matter how many times we input 1 to control the car to move right, it will eventually display like the fourth picture. This is

because when the car moves, the probability of moving incorrectly to either side is 0.1, and the sum of the probability of moving right and correctly is 0.9, which results in the situation shown in the picture.

The observation model

In this section, we learned about the observation model and the related analysis process.

First, our known condition is

$$P(O_t = observedColor | S_t = s_t)$$

We need to calculate

$$P(S_t = s_t | O_t = observedColor)$$

You can obtain the result using Bayes' formula.

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

Wk.11.1.1

In this problem,we look at defining observation models.We define conditional distributions on the color that the robot will observe in a room,given the actual color of that room.

part1:

When the actual color is white or green, the probability of observing white and green is 0.5 each, while observations of other colors match the actual color.

```
def whiteEqGreenObsDist(actualColor):
    if actualColor in ['white','green']:
        return dist.DDist({'white':0.5,'green':0.5})
    else:
        return dist.DDist({actualColor:1.0})
```

Part2:

The white room is always observed as green, the green room is always observed as white; observations of other colors match their actual colors.

```
def whiteVsGreenObsDist(actualColor):
    if actualColor == 'white':
        return dist.DDist({'green':1.0})
    elif actualColor == 'green':
        return dist.DDist({'white':1.0})
    else:
        return dist.DDist({actualColor:1.0})
```

Part3:

It is observed that the probability of being the actual color is 0.8, with the remaining 0.2 probability evenly distributed among the other possible colors.

```
def noisyObs(actualColor):
    otherColors = []
    for color in possibleColors:
        if color != actualColor:
            otherColors.append(color)
    colorNum = len(otherColors)
    otherProb = 0.2 / colorNum
    distDic = {actualColor:0.8}
    for color in otherColors:
        distDic[color] = otherProb
    return dist.DDist(distDic)
```

Part4:

Build a complete observation model and pass in standardHallway and

noisyObs

```
noisyObsModel = makeObservationModel(standardHallway,noisyObs)
```

Check Yourself 3. Finish the exercise

Our answer:

1. $Pr(\text{obs} \mid R_0) = [1,0]$ $Pr(\text{obs} \mid R_1) = [0,1]$

Under a perfect observation model, you can accurately see the color in whichever room you are in.

2. $Pr(\text{obs} \mid R_0) = [0.5,0.5]$ $Pr(\text{obs} \mid R_1) = [0.5,0.5]$

Under whiteEqGreenObsDist, the probability of seeing the wrong color is 0.5.

3. $Pr(\text{obs} \mid R_0) = [0,1]$ $Pr(\text{obs} \mid R_1) = [1,0]$

Under whiteVsGreenObsDist, the color will definitely be misperceived.

The state-transition model

In this section, we learned how to analyze state transition equations.

We know

$$P(S_t = s_t | O_t = observedColor)$$

You only need to multiply the probability distribution after movement based on the observations to get it.

$$P(S_{t+1} = s_{t+1} | S_t = s_t, A_t = a_t)$$

Wk.11.1.2

In this problem,we look at defining transition models

Part 1:

The corridor is connected end to end, so the robot is not blocked by boundaries but loops to the other end.

```
def ringDynamics(loc,act,hallwayLength):
    newLoc = loc + act
    return newLoc % hallwaylength
```

Part 2:

The robot has a 0.1 probability of sliding one square to the left of the nominal position, and a 0.9 probability of staying at the nominal position; if the robot is at the far left, it stays in place 100% of the time.

```
def leftSlipTrans(nominalLoc,hallwayLength):
    if nominalLoc == 0:
        return dist.DDist({nominalLoc:1.0})
    else:
        leftLoc = nominalLoc - 1
        return dist.DDist({nominalLoc:0.9,leftLoc:0.1})
```

Part 3:

The robot has a 0.8 probability of staying in its nominal position, a 0.1 probability of sliding 1 space to the right, and a 0.1 probability of sliding 1 space to the left; if sliding goes beyond the boundary, the probability is assigned to the corresponding boundary position (it cannot slide left at the left boundary or right at the right boundary).

```
def noisyTrans(nominalLoc,hallwayLength):
    pDic = {nominalLoc:0.8}
    maxLoc = hallwayLength - 1

    if nominalLoc > 0:
```

```
        leftLoc = nominalLoc - 1
        pDic[leftLoc] = 0.1
    else:
        pDic[nominalLoc]=0.9

    if nominalLoc < maxLoc:
        rightLoc = nominalLoc + 1
        pDic[rightLoc] = 0.1
    else:
        pDic[nominalLoc]=0.9

    return dist.DDist(pDic)
```

Part 4:

Build a complete model and input the lengths of standardDynamics, noisyTrans, and standardHallway

```
noisyTransModel = makeTransitionModel(standardDynamics,noisyTrans,5)
```

Checkoff 1. Comparison of four models

Our answer:

The black sensor is completely useless; whiteEqGreen is useful but fuzzy; whiteVsGreen is complete but with incorrect labels; the perfect sensor is both accurate and correct.

State estimation in the hallway world

Wk.11.1.4

Solution process:

1.Observations and movements are flawless.

Ⅰ.First, under the initial distribution, the probability in each room is the same, so

$$Bo(S) = Pr(S_0 = S) = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$$

II.Next, the robot observes that the first color is white, and we need to update the probability we know.

$$B_0(s) = Pr(O_0 = white | S_0 = s) = [1,0,1]$$

Using the law of total probability, we can calculate

$$Pr(O_0 = white) = \sum Pr(O_0 = white, S_0 = s) = \frac{2}{3}$$

Then deduce it using Bayes' theorem

$$B_0'(s) = Pr(s_0 = s | O_0 = white) = \frac{Pr(O_0 = white, S_0 = s)}{Pr(O_0 = white)} = \left[\frac{1}{2}, 0, \frac{1}{2}\right]$$

III.Now the robot moves one step to the right, which is easy to achieve.

$$B_1(s) = Pr\ (S_1 = s \mid O_0 = \text{white}, I_0 = 1) = [0, \frac{1}{2}, \frac{1}{2}]$$

IV.It is now observed to be green in color

$$Let \mathcal{H}_0 = \{O_0 = \text{white}, I_0 = 1\}$$

$$\begin{aligned} &Pr(O_1 = \text{green}, S_1 = s, \mathcal{H}_0) \\ &= Pr(O_1 = \text{green} \mid S_1 = s, \mathcal{H}_0) \cdot Pr(S_1 = s \mid \mathcal{H}_0) \\ &= \left[0, \frac{1}{4}, 0\right] \end{aligned}$$

Using the law of total probability

$$Pr(O_1 = green) = \sum_{i=1}^{n} Pr(O_1 = \text{green}, S_1 = s, \mathcal{H}_0) = \frac{1}{4}$$

Reuse Bayes' theorem

$$B_1'(s) = \frac{Pr(O_1 = \text{green}, S_1 = s, \mathcal{H}_0)}{Pr(O_1 = green)} = [0,1,0]$$

V.Now move one square to the right

$$B_2(s) = Pr(S_2 = s \mid O_0 = \text{white}, I_0 = 1, I_1 = 1) = [0,0,1]$$

2.The probability that the sensor correctly identifies the color is 0.8, and the probability of identifying other colors is0.05

Ⅰ.First, under the initial distribution, the probability in each room is the same, so

$$Bo(S) = Pr(S_0 = S) = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$$

Ⅱ.Let's calculate the probability of each observed color.

$$Pr(O_0 = black) = Pr(O_0 = red) = Pr(O_0 = blue) = \frac{1}{3} \cdot 0.05 \cdot 3 = \frac{0.15}{3}$$

$$Pr(O_0 = green) = \frac{2}{3} \cdot 0.05 + \frac{1}{3} \cdot 0.8 = \frac{0.9}{3}$$

$$Pr(O_0 = white) = \frac{2}{3} \cdot 0.8 + \frac{1}{3} \cdot 0.05 = \frac{1.65}{3}$$

Ⅲ.Next, an observation O0=white is introduced

$$Pr(S_0 = s, O_0 = white) = Pr(O_0 = white \mid S_0 = s) \cdot Pr(S_0 = s)$$
$$= \left[\frac{0.8}{3}, \frac{0.05}{3}, \frac{0.8}{3}\right]$$

Using Bayes' theorem

$$Pr(S_0 = s|O_0 = white) = \frac{Pr(S_0 = s, O_0 = \text{white})}{Pr(O_0 = white)} = \left[\frac{0.8}{1.65}, \frac{0.05}{1.65}, \frac{0.8}{1.65}\right]$$

IV. Move one space to the right now I_0=1

$$Pr(S_1 = s|O_0 = white, I_0 = 1) = \left[\frac{0}{1.65}, \frac{0.8}{1.65}, \frac{0.85}{1.65}\right] = \left[0, \frac{16}{33}, \frac{17}{33}\right]$$

V. Now let's calculate the probability of observing each color.

$$Pr(O_1 = black) = Pr(O_1 = red) = Pr(O_1 = blue) = 0.05$$

$$Pr(O_1 = green) = \frac{6}{33} \times 0.8 + \frac{17}{33} \times 0.05 = \frac{13.65}{33}$$

$$Pr(O_1 = white) = \frac{16}{33} \times 0.05 + \frac{17}{33} \times 0.8 = \frac{14.4}{33}$$

VI. Observed once again O_1=*white*

$$Pr(S_1 = s, O_1 = white) = Pr(O_0 = white \mid S_1 = s) \cdot Pr(S_1 = S) = [0, \frac{0.8}{33}, \frac{13.6}{33}]$$

The result can be obtained by using the Bayes formula

$$Pr(S_1 = s|O_1 = white) = \frac{Pr(S_1 = s, O_1 = \text{white})}{Pr(O_1 = white)} = \left[0, \frac{1}{18}, \frac{17}{18}\right]$$

VII. Move one more square to the right

$$Pr(S_2 = s|O_1 = white, I_1 = 1) = [0,0,1]$$

VIII. Move one more square to the right

$$Pr(S_3 = s|O_1 = white, I_1 = 1, I_2 = 1) = [0,0,1]$$

Wk.11.1.5

Solution process:

The action also has noise, with a 0.8 probability of being correct, and a 0.1 probability of moving left or right relative to the target position.

I.First, under the initial distribution, the probability in each room is the same, so

$$Bo(S) = Pr(S_0 = S) = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$$

II. Initially observed as white, same as above.

$$B_0'(s) = \left[\frac{16}{33}, \frac{1}{33}, \frac{16}{33}\right]$$

III. Now take a step to the right $I_0 = 1$

Start from s=0

$$\left[\frac{16 \times 0.1}{33}, \frac{16 \times 0.8}{33}, \frac{16 \times 0.1}{33}\right]$$

Start from s=1

$$\left[\frac{1 \times 0}{33}, \frac{1 \times 0.1}{33}, \frac{1 \times 0.9}{33}\right]$$

Start from s=3

$$\left[\frac{16 \times 0}{33}, \frac{16 \times 0.1}{33}, \frac{16 \times 0.9}{33}\right]$$

Corresponding to the sum of s

$$B_1(s) = \left[\frac{1.6}{33}, \frac{14.5}{33}, \frac{16.9}{33}\right]$$

IV. Seeing white again $O_1 = white$

$$Pr(O_1 = white, B_1(s)) = \left[\frac{1.6 \times 0.8}{33}, \frac{14.5 \times 0.05}{33}, \frac{16.9 \times 0.05}{33}\right] = \left[\frac{1.28}{33}, \frac{0.725}{33}, \frac{13.52}{33}\right]$$

Calculate using the law of total probability

$$Pr(O_1 = white) = \sum_{i=1}^{n} Pr(O_1 = white, B_1(s)) = \frac{15.525}{33}$$

Calculate using Bayes' theorem

$$B_1'(s) = \frac{Pr(O_1 = white, B_1(s))}{Pr(O_1 = white)} = [\frac{256}{3105}, \frac{145}{3105}, \frac{2704}{3105}]$$

V. Move one more square to the right I1=1

Start from s=0

$$\left[\frac{256 \times 0.1}{3125}, \frac{256 \times 0.8}{3125}, \frac{256 \times 0.1}{3125}\right]$$

Start from s=1

$$\left[0, \frac{145 \times 0.1}{3125}, \frac{145 \times 0.9}{3125}\right]$$

Start from s=2

$$\left[0, \frac{2704 \times 0.1}{3125}, \frac{2704 \times 0.9}{3125}\right]$$

Corresponding to the sum of s

$$B_2(s) = [\frac{256}{31050}, \frac{4897}{31050}, \frac{25897}{31050}]$$

Preparing to localize

Wk.11.1.6

When calculating the position of the robot in the global odometer coordinate system, the coordinates $(x_S,\ y_S)$ of the sensor center relative to

the robot center, as well as the angle thetaS between the beam direction and the robot's heading, are obtained through the sonar sensor. Using the transformation formula from coordinate system A to B:

$$a_x = x_B + cos\theta_B \cdot bx - sin\theta_B \cdot b_y$$

$$a_y = x_B + sin\theta_B \cdot bx + cos\theta_B \cdot b_y$$

We can first convert the sonar coordinate system to the robot coordinate system, and then to the global coordinate system. This can be achieved through the following code:

```
def sonarHit(distance, sonarPose, robotPose):

    hit_x_sensor = distance

    hit_y_sensor = 0

# Step 1: Sonar coordinate system → Robot coordinate system

    hit_x_robot = sonarPose.x + math.cos(sonarPose.theta) * hit_x_sensor -
math.sin(sonarPose.theta) * hit_y_sensor

    hit_y_robot = sonarPose.y + math.sin(sonarPose.theta) * hit_x_sensor +
math.cos(sonarPose.theta) * hit_y_sensor

# Step 2: Robot Coordinate System → Global Coordinate System

    hit_x_global = robotPose.x + math.cos(robotPose.theta) * hit_x_robot -
math.sin(robotPose.theta) * hit_y_robot

    hit_y_global = robotPose.y + math.sin(robotPose.theta) * hit_x_robot +
math.cos(robotPose.theta) * hit_y_robot

    return util.Point(hit_x_global, hit_y_global)
```

Wk.11.1.7

By using the ideal sonar readings under the calculated robot posture to determine the robot's position, we can utilize the functions discreteSonar and idealReadings: discreteSonar is used to discretize the continuous sonar readings into a finite number of integer bins to handle measurement noise; idealReadings, based on the known map and the possible positions of the robot, calculates the ideal intersection distances between the sonar sensor and the wall at each position, and discretizes these readings, thereby providing the basis for the observation model for the state estimation algorithm, enabling the robot to update its probability distribution of position by comparing the actual observations and the ideal readings.

For the discreteSonar function：

```
def discreteSonar(sonarReading):

    # Calculate the bin width

    bin_width = sonarMax / numObservations

    # If the reading exceeds the maximum value, it will be truncated and the last bin
will be returned.

    if sonarReading >= sonarMax:

        return numObservations - 1

    # Calculate the bin index

    bin_index = int(sonarReading / bin_width)
```

```
    return bin_index
```

For the idealReadings function：

```
def idealReadings(wallSegs, robotPoses):

    ideal_readings = []

    for robotPose in robotPoses:

        # Check the intersection points with each wall section

        for wall_seg in wallSegs:

            intersection = wall_seg.intersection(beam_segment)

            if intersection:  # If there is an intersection point

                dx = intersection.x - sonar_global_x

                dy = intersection.y - sonar_global_y

                distance = math.sqrt(dx*dx + dy*dy)

                # Update Minimum Distance

                if distance < min_distance:

                    min_distance = distance

        # Discretization Distance

        discrete_reading = discreteSonar(min_distance)

        ideal_readings.append(discrete_reading)

    return ideal_readings
```

Wk.12.2.3: Localization

1. Preprocessor (at time 0):

Input: an instance of io.SensorInput:

sonars = (0.8, 1.0, ...)

odometery = Pose (1.0, 0.5, 0.0)

Output: a tuple (obs, act); if the output is None, enter None in both boxes.

obs = None

act = None

Explain:

Since this is the first run and there is no previous data, the output, obs (the index of a discretized sonar reading at time 0) and the act (the index of a discretized motion started at time 0), is None.

2. Preprocessor (at time 1):

Input: an instance of io.SensorInput:

sonars = (0.25, 1.2, ...)

odometry = Pose(2.4, 0.5, 0.0)

Output: a tuple (obs, act); if the output is None, enter None in both boxes.

obs = 5

act = 2

Explain:

The sonar reading at time 0 (0.8) for obs: Now calculate the index. Since sonarMax = 0.15, 0.8 / 0.15 = 5.333, which falls within the range of index 5 (rounded), so obs = 5.

The act is based on the change in odometry: As given in the problem, xMin = 0.0, xMax = 10.0, numStates = 10, so the state interval width w = (10.0 - 0.0) / 10 = 1. At time 0, the odometry x = 0, and at time 1, the odometry x = 2.4, $\Delta x = 2.4 - 0 = 2.4$, *so act =* $\Delta x / w \approx 2$.

3. Preprocessor (at time 2):

Input: an instance of io.SensorInput:

sonars = (0.16, 0.2, ...)

odometry = Pose (7.3, 0.5, 0.0)

Output: a tuple (obs, act); if the output is None, enter None in both boxes.

obs = 1

act = 5

Explain:

The sonic reading of obs based on time 1 (0.25): Calculate the index, 0.25 / 0.15 = 1.666, which falls within the range of index 1 (0 < 1 ×0.15 = 0.15 < 2 ×0.15 = 0.3), so obs = 1.

Act based on the change in the odometer: Time 1 odometer x = 2.4, Time 2 odometer x = 7.3, $\Delta x = 7.3 - 2.4 = 4.9$, so act = $\Delta x / w \approx 5$.

4. Estimator (at time 0):

Input: (obs, act) the output tuple from Preprocessor

Output: probability distribution over robot states (x indices)

Probability distribution: Each probability for states 0 to 9 is 0.10

Explain:

Since the output of the Preprocessor is None, the belief state remains unchanged and is initially uniformly distributed. The probability of each state is 1/10 = 0.100.

5. Estimator (at time 1):

Input: (obs, act) the output tuple from Preprocessor

Output: probability distribution over robot states (x indices)

Probability distribution:

State 2: 0.250

State 5: 0.250

State 9: 0.500

The probability of other states is 0

Explain:

Since the ideal (discretized) sonar readings for each state are: ideal = (5, 1, 1, 5, 1, 1, 1, 5, 1, 5)，and when Preprocessor (at time 1)obs is 5,so for s = 0, 3, 7, 9, P(obs=5|s) × P(s) = 1 × 0.100 = 0.100; for other states, it is 0. After normalization, the probabilities of states 0, 3, 7, and 9 are each 0.250; the probabilities of other states are 0.

State transition:

State 0 transitions to 0 + 2 = 2, with a probability of 0.250;

State 3 transitions to 3 + 2 = 5, with a probability of 0.250;

State 7 transitions to 7 + 2 = 9, with a probability of 0.250;

State 9 transitions to 9 + 2 = 11, and is truncated to 9, with a probability of 0.25.

New state probability:

State 2: Originating from State 0, with a probability of 0.250;

State 5: Originating from State 3, with a probability of 0.250;

State 9: Originating from States 7 and 9, with a probability of 0.250 + 0.250 = 0.500.

6. **Estimator (at time 2):**

Input: (obs, act) the output tuple from Preprocessor

Output: probability distribution over robot states (x indices)

Probability distribution:

State 7: 0.500

State 9: 0.500

The probability of other states is 0

Explain:

Preprocessor (at time 1)obs is 1,for s = 1, 2, 4, 5, 6, 8, P(obs = 1 | s) = 1; otherwise, the probability is 0. The current belief state is:

State 2: P(obs=1|2) × P(2) = 1 × 0.250 = 0.250

State 5: P(obs=1|5) × P(5) = 1 × 0.250 = 0.250

State 9: P(obs=1|9) × P(9) = 0 × 0.500 = 0

After normalization:：

State 2: 0.250/0.500 = 0.500

State 5: 0.250/0.500 = 0.500

Then the state transition: (when act = 5)

State 2 transitions to 2 + 5 = 7, with a probability of 0.500;

State 5 transitions to 5 + 5 = 10, and is truncated to 9, with a probability of 0.500.

New state probability:

State 7: Originating from State 2, with a probability of 0.500;

State 9: Originating from State 5, with a probability of 0.500;

Appendix: The Description of AI Usage in the Report

In this report, we used AI to complete the translation and express our ideas more clearly. Additionally, AI was used to enhance the understanding of certain concepts in the document.