CS 81 - Principles of Robot Design & Programming

Carter Kruse

Homework 4 - State Estimation

Purpose

In this assignment, you will put in practice some of the state estimation concepts learned in class.

General Instructions

Answer the following questions, showing all the relevant drawings and intermediate math work by uploading a PDF file containing the work. You can include code written by yourself if used for doing some of the repeated calculations, but write down the formulas used on paper. Feel free to add comments in the text box.

Question 1

Consider a robot constrained to move on a line world. There are 10 cells and the robot is positioned on the 7th cell. The robot doesn't know where it starts from and doesn't have any sensor.

The world is bounded, so the robot cannot move to outside of the specified area. Furthermore, assume that at each time step the robot can execute either a move forward or a move backward command. Unfortunately, the motion of the robot is subject to error, so if the robot executes an action it will sometimes fail. When the robot moves forward (increasing numbering of the cell) we know that the following might happen:

- With a 25% chance the robot will not move.
- With a 50% chance the robot will move to the next cell.
- With a 25% chance the robot will move two cells forward.
- There is a 0% chance of the robot either moving in the wrong direction or more than two cells forwards.

Assume the same model also when moving backward, just in the opposite direction.

Since the robot is living on a bounded world, it is constrained by its limits. This changes the motion probabilities on the boundary cells, namely:

- If the robot is located at the last cell and tries to move forward, it will stay at the same cell with a chance of 100%.
- If the robot is located at the second to last cell and tries to move forward, it will stay at the same cell with a chance of 25%, while it will move to the next cell with a chance of 75%.

Again, assume the same model when moving backward, just in the opposite direction.

Given this problem, write the models and calculate using a discrete Bayes filter the final belief on the position of the robot after having executed 3 consecutive move forward commands and 2 consecutive move backward commands. You can write a program to calculate the results and submit that as well.

To answer this question, let us first address the idea of state estimation, which is a way to estimate the state x of a system given an observation z and controls u, represented as $p(x|z,u)$. Using probabilistic reasoning, we are able to account for unobserved variables and imperfect computation, giving us a framework for managing our beliefs and knowledge.

Using a forward kinematics model, we are able to determine the next state given a current state and control, given as follows: $x_{t+1} = f(x_t, u_t)$. In this situation, we do not have a sensor attached to the robot, nor is there a way to directly observe it's location, so we do not need to worry about the expected measurement given the robot's current state. That is, we are simply trying to determine the possible locations of the robot (on the line) along with the associated probabilities.

The localization problem answers the question as to where the robot is, particularly in cases involving an environment map and sensor data, by returning a pose estimate of the robot. Global localization is introduced when a robot does not know it's initial pose, as in this question.

To account for noise/errors, probabilistic approaches can be used to model the belief of a robot's pose. As the example in class demonstrated, global localization on a line may be used to determine which "door" a robot started at. A uniform probability distribution is initially provided, as the robot does not know where it started from.

To determine the localization of a robot, a Bayesian filter may be introduced, which estimates the state x from data Z. That is, the Bayesian filter aids in determining the probability that a robot is at x (in this case, where x, the state, is the robot location). Z represents data including sensor readings, which we do not have to worry about in this case.

The Bayesian filter recursively computes the posterior distribution, as follows: $\text{Bel}(x_t) = P(x_t|Z_t)$. By iterating the Bayesian filter, we are able to propagate the motion model, as follows:

$$\text{Bel}(x_t) = \int P(x_t|a_{t-1}, x_{t-1})\,\text{Bel}(x_{t-1})\,dx_{t-1}$$

In this sense, we may compute the current state estimate before taking a sensor reading by integrating over all possible previous state estimates and applying the motion model. Given this particular problem, we do not need to update the sensor model, so this will not be considered.

To summarize, when a mobile robot moves, we are able to determine the localization probabilistically, using a prediction/propagation model, which is used to determine the robot's state/pose x after action A. The question then becomes what the probability density function is that appropriately describes the uncertainty of the robot's state/pose.

In the discrete case, we may use the following:

$$P(x|u) = \sum P(x|u, x')\,P(x')$$

This is based on the motion model, which describes the relative motion of the robot $p(x_t|x_{t-1}, u_t)$.

So... if we have a robot constrained to move on a line world, with 10 cells, where the robot does not know where it starts from (and does not have any sensor), we may set up the model according to the specifications.

Let us start at the beginning (with time $t = 0$), and describe the appropriate probabilities of motion, which allow us to determine the probability that the robot is located at a certain position after a given time. That is, using the following model, we are able to describe the probability that the robot is in a given cell ($1 \leq n \leq 10$, where n is the cell) at time t, where each time step is representative of a single action.

As indicated, the world is bounded, so the robot cannot move to outside of the specified area. Furthermore, assume that at each time step the robot can execute either a move forward or a move backward command (as emphasized).

As the motion of the robot is subject to error, when the robot executes an action, it will sometimes fail. Further, since the robot is living on a bounded world, it is constrained by its limits. This changes the motion probabilities on the boundary cells. The specifics are given in the statement of the problem.

If the command is 'move forward', we may describe the probability is at the n^{th} cell (after the command, so at time $t+1$) by the following system:

$$\begin{aligned}
P_{t+1}(n) &= 0.25 \times P_t(n) && (n = 1) \\
P_{t+1}(n) &= 0.25 \times P_t(n) + 0.50 \times P_t(n-1) && (n = 2) \\
P_{t+1}(n) &= 0.25 \times P_t(n) + 0.50 \times P_t(n-1) + 0.25 \times P_t(n-2) && (3 \leq n \leq 9) \\
P_{t+1}(n) &= 1.00 \times P_t(n) + 0.50 \times P_t(n-1) + 0.25 \times P_t(n-2) && (n = 10)
\end{aligned}$$

If the command is 'move backward', we may describe the probability is at the n^{th} cell (after the command, so at time $t+1$) by the following system:

$$\begin{aligned}
P_{t+1}(n) &= 0.25 \times P_t(n) && (n = 10) \\
P_{t+1}(n) &= 0.25 \times P_t(n) + 0.50 \times P_t(n+1) && (n = 9) \\
P_{t+1}(n) &= 0.25 \times P_t(n) + 0.50 \times P_t(n+1) + 0.25 \times P_t(n+2) && (2 \leq n \leq 8) \\
P_{t+1}(n) &= 1.00 \times P_t(n) + 0.50 \times P_t(n+1) + 0.25 \times P_t(n+2) && (n = 1)
\end{aligned}$$

As implied, these systems reflect the different probabilities, according to a given situation, and thus, are responsible for composing the discrete Bayesian filter. Given that we are trying to determine the probability of the robot being located at a cell n after the transition, we must carefully consider the different aspects of the previous state that contribute to the overall probability of the current state.

To calculate the probabilities (according to the models), using a discrete Bayesian filter, we use a simple Python script. This allows us to determine the final belief on the position of the robot after having executed 3 consecutive move forward commands and 2 consecutive move backward commands.

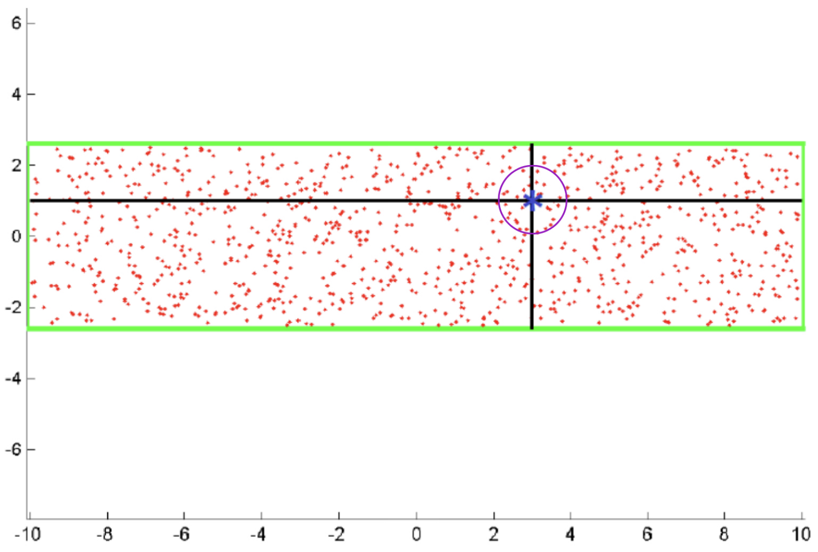
From the statement of the problem, the robot does not know where it starts from and does not have any sensor. Thus, we begin with a uniform probability distribution function over the 10 cells of the environment, so the probabilities are initialized as follows: $P_0(n) = 0.1$ for all $1 \leq n \leq 10$.

The solution (representing the probabilities after the actions), is given as follows ($t = 5$):

$P_5(1) = 0.06152343750$	$P_5(2) = 0.06230468750$	$P_5(3) = 0.08281250000$
$P_5(4) = 0.09453125000$	$P_5(5) = 0.09892578125$	$P_5(6) = 0.11865234375$
$P_5(7) = 0.16875000000$	$P_5(8) = 0.18125000000$	$P_5(9) = 0.10625000000$
$P_5(10) = 0.02500000000$		

Question 2

In the following picture, a robot is deployed in the a rectangular room at the asterisk, where the walls are depicted with green lines. The robot is pointing upwards and the black lines indicate four measurements from the sonar sensor that returns the distances to the walls. The red points indicate the particles of a particle filter at the initialization – the robot doesn't know where it is in the environment. Please draw the areas where after the update of the particle filter, particles will have the weight increased.



After the update of the particle filter, the particles located around the robot location will have their weights increased. This is because we assume that the robot is using the sonar measurements to update the particle filter, so the particles that are consistent with the sonar measurements will have their weights increased.

Depending on the degree of error we include with the sonar measurements, a circle (or disc) of a given size (radius) will encompass the points that have increased weight (for the filter). As indicated, the exact size of this area will depend on the uncertainty of the particle filter introduced, along with the resolution of the sonar measurements.

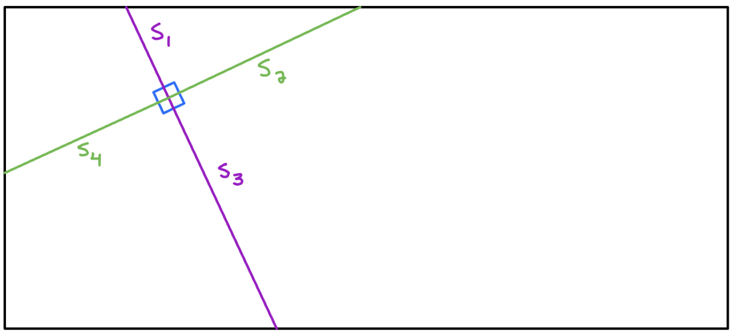
Given this problem, write whether the robot can globally localize in that room. Please motivate the answer showing the motions that it has to do to localize (if possible to localize uniquely) or to minimize the number of hypotheses (if not possible to localize uniquely).

Given the circumstances of the problem, the robot will not be able to globally localize in the room. If the robot were able to access additional sensors (or other sources of information, including a GPS system), then the robot would be able to globally localize. In this case, the robot would be able to use the sonar measurements and the GPS (positioning) system to determine its position and orientation (pose).

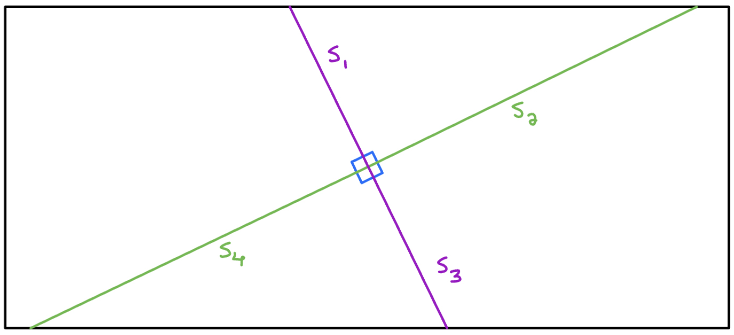
On the contrary, in this situation, the robot only has the sonar sensor, so it is unable to globally localize within the room, as the sonar measurements alone do not provide enough information to uniquely determine the robot's pose (position and orientation).

To minimize the number of hypothesis, the robot should move in such a way that minimizes the value of *each* sonar measurement, which (in this case) will make opposite sonar measurements equal. Due to the symmetry of the environment, this will confirm that the robot is positioned in the center of the environment (as the environment is a rectangle).

In particular, let's say that the robot starts at a given unknown position with an unknown orientation. This may be shown as follows, where S_1, S_2, S_3, and S_4 represent the sonar sensor readings/measurements, with S_1 representing the direction in front of the robot.



By moving the robot down and to the right (without changing the angle/orientation of the robot), we are able to minimize the each of the values of S_1, S_2, S_3, and S_4 *individually*. That is, given the rectangular environment (and the symmetry induced by it), we are able to ensure $S_1 = S_3$ and $S_2 = S_4$. This is shown by the following image.



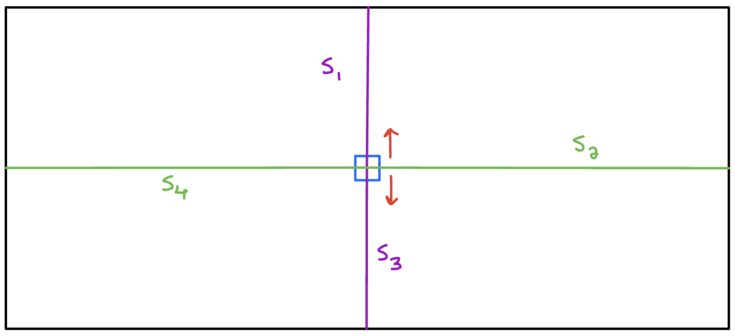
Of particular importance is that due to the nature of the environment, we may not be able to move linearly to the right (in the robot's frame of reference) followed by linearly down (in the robot's frame of reference). In other words, we would need to iteratively check the sonar sensors readings, continuing to move until the readings are equivalent. (That is, we could not simply equate $S_1 = S_3$, and then do the same for $S_2 = S_4$, before moving on, but would rather have to go back and check $S_1 = S_3$, and update if this is not the case).

This is due to the rectangular environment creating a situation in which the sensor readings switch from reading one of the walls to an adjacent wall. Regardless, this is a minor point that does not negatively impact the performance of such an iterative algorithm.

To minimize the sonar readings/measurements, the robot would simply perform a series of movements and sonar measurements to determine the direction of motion to take to minimize the sonar readings.

Once the robot is known to be located in the center of the environment, we are able to minimize the number of hypotheses to 2, according to the orientation of the robot. To do so, we simply rotate the robot in place, until S_1 is at it's absolute minimum. This would involve rotating the robot 360º and determining the absolute minimum value from that.

The result will be two "peaks" where the absolute minimum occurs; one will be with the robot facing upward, while the other will be with the robot facing downward (in the reference frame of the environment). This is shown as follows.



At this point, as indicated, the robot's pose (position and orientation) will be one of two possibilities. We know (given the environment of a rectangle) that the robot is located in the center of the rectangle, and though we are unable if the robot is pointed upward or downward, at least we know that it is one of the two.

As a consequence of rotating the robot in place at the center of the rectangle, the values of S_2 and S_4 will be local minima. This is not particularly important, though could be used to confirm the previous results.

Even without considering the algorithm given above, it is relatively straightforward to determine the robot's position is unable to be globally localized, due to the symmetry of the environment. For example, if the robot were facing directly upward $1m$ away from the top wall and $1m$ away from the left wall, the sensor readings would exactly match that as if the robot were facing directly downward $1m$ away from the bottom wall and $1m$ away from the right wall. This symmetry of the environment gives indication as to why an additional source of information would be necessary to completely localize the robot in the global reference frame.

Typically, globally localizing a robot involves a series of movements and measurements (sonar, GPS, etc) that is designed to reduce the number of possible locations that the robot could be in. By moving the robot around the environment and comparing this to the different measurements that are given, we are able to place the robot within a map of the environment, reducing the number of hypotheses.

As demonstrated, this approach is not guaranteed to uniquely determine the robot's pose (position and orientation) with limited information, yielding multiple possible hypotheses.