Assignment 3 : Probalistic reasoning over time

Christoffer Lindell Bolin

March 7, 2022

1 Background

We have been tasked in this assignment to implement a Hidden Markov Model (HMM) and apply forward filtering for an environment without any landmarks. In our case, an empty n x m rectangular room was used as the environment with a robot that moves according to a specific strategy. The only available evidence we have is a noisy sensor reading that gives a direct, but vague approximation to the robot's location. The simulation of the robots movement and sensor had to be implemented by myself.

Peer-review was another part of this assignment, I did a peer-review with Adam Forsberg and have also during the implementation-phase discussed with Pontus Persson.

2 Peer-review

During the peer-review with Adam, he pointed out to me a few possible enhancements that could be made to further improve my overall performance. First and foremost he mentioned that I normalize the probability vector by dividing with the vectors norm. I have done this part completely wrong, since it does not sum up the probabilities to 1. This has now been changed. Another noteworthy improvement he mentioned was to explore the possibility to estimate based on the sum over the state probabilities for one position. I have thought about this, but in the end due to time constraints I have decided to not implement this. I have also realized on my own that I had redundant code, I barely utilized any of the skeletons help methods, this is now tweaked. The full revised version is available on GitHub.[1]

3 Models

In the handout file, a observation, transition and state model were included. The observation model keeps track of the probabilities which the sensor reports. This is shown as the transition model keeps track of the probabilities for the different poses to be reached. Which is shown in the visualization accordingly

[1] *Github code.* URL: `https://github.com/chrisse22222/AI_Course`.

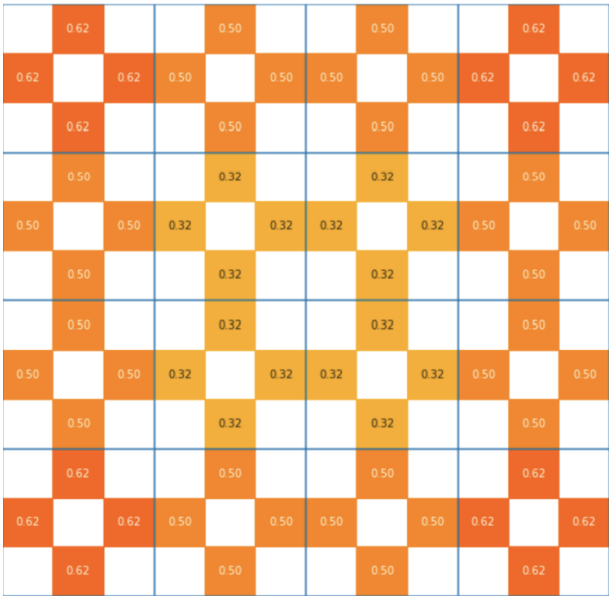

Figure 1: Probability distribution for no sensor reading in a 4x4 grid

to a 70 % chance to move straight and 30 % of a change in direction unless a wall is adjacent. The sensor model contains the probability distribution of the robots true position, given the sensor reading. Finally the state model is able to transform a pose or position into a state or sensor reading and the other way around.

The "no reading" from the sensor is sightly more likely to get when the robot is close to a wall which is demonstrated in figure 1.

4 Result

In order to evaluate the result of my implementation, I compared it to pure guessing with 100 random moves in a 8X8 grid. Pure guessing is predicting a completely random move. I also compared it to sensor reading, where it guesses the position of the robot where the sensor is predicting. Although, since the sensor can return a "no reading", a random move was selected in this case. The result from this comparison can be seen in the table below.

Method	Average Error	Correct guesses
Pure Guessing	5.26	2
Sensor reading	3.48	9
HMM	1.81	40

As we clearly can see, my HMM implementation is a substantial improvement from pure guessing and a major improvement from sensor reading, it clearly shows the major advantage of forward filtering.

5 Article Summary

In the article[2] Fox et al presents an algorithm called Monte Carlo Localization (MCL) which aims to tackle the heavy burden of memory usage and low accuracy of previous algorithms in the sensor-based localization space. MCL achieves this by reducing the sampling rate when the position of the robot is known that is based on the error rate. It also can sample based on the most likely position, this allows it to focus the computation where it is most needed. It also differs from previous algorithms in which it uses randomized samples to represent the robots assumed position. This leads to a major reduction in computation and memory consumption which in turn is beneficial since it allows a higher sample frequency that implies a much higher accuracy. MCL uses particle filtering methods, although the authors have proposed to use an adaptive sampling scheme instead. This is showcased in a series of experiments to demonstrate their points. The two algorithms fared equally well in terms of accuracy. However since the adaptive sampling scheme require less intense computation, it is far superior to particle filtering.

6 Discussion

Despite the substantial improvement with HMM from pure guessing, it is not sufficient enough to tackle the problem of robot localisation mainly due to the small accuracy. It is a way too generalized approach. Our HMM implementation assumes certain characteristics of the robots movement, sensors and the environment. For example, in our case with the movement of the robot, there are only four possible headings and it moves one step in the facing direction in a grid-based system. It would be unrealistic to assume that a robot would make these perfect moves unless it was placed on rails. In regards to the sensor probability distribution. There are many different variants of sensors with variable noise. The way we simulate our sensor probability distribution assumes uncomplicated scenarios. Another noteworthy acknowledgement is in regards to multi robot scenarios in which MCL excels. This is not the case with our implementation, if you tried to place several agents in one shared environment, the accuracy would suffer further.

References

Fox, Dieter et al. "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots". In: *AAAI* (1999). URL: https://aaai.org/Papers/AAAI/1999/AAAI99-050.pdf.

Github code. URL: https://github.com/chrisse22222/AI_Course.

[2]Dieter Fox et al. "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots". In: *AAAI* (1999). URL: https://aaai.org/Papers/AAAI/1999/AAAI99-050.pdf.