Artificial Intelligence - Assignment 3

Szymon Stypa - sz6102st-s

March 3, 2021

1 Overview

The main goal of this assignment consisted of implementing robot localization in an environment without any landmarks. This was to be done using forward filtering with a Hidden Markov model. A skeleton notebook with a state model, transition model, observation model and visualization code was provided by the course staff for this task.

2 Implementation

I started this assignment by implementing the robot simulation class, responsible for transitioning the robot between states and providing sensor readings. This class keeps an internal state which consists of the robot position and heading, and has two methods which are called through the update method of the localizer. Firstly, we have the `move` method which generates a new position for the robot based on the movement probabilities provided in the description of this assignment. Secondly, the `sense` method returns a sensor reading which also probabilistically conforms to the mentioned description.

Besides this, a `HMMFilter` class is present and responsible for the Hidden Markov Model forward filtering. This class uses both the transition and observation models to obtain the forward vector at each time step and calculates it for the localizer through its `update` method. This is done by obtaining the transition matrix from the transition model class and the sensor matrix corresponding to the current sensor readings from the sensor model class. Together with the previous forward vector, these matrices are multiplied accordingly and normalised to form a new forward vector at each update. To evaluate the solution, I used two measures: mean Manhattan distance and accuracy. After testing for multiple long runs, these measures took the values of approximately 1.66 and 0.33 respectively.

3 Peer Review

During the implementation phase, I discussed and consulted with my classmates Johannes Bastmark and Axel Domell. These were very early stages of implementation so our work together consisted of no more than figuring out what we are supposed to do and how things are expected to work.

My peer review was done with Filip Alberius, with who I discussed my solution and possible improvements. Our Hidden Markov Model filter implementations turned out to be pretty much exactly the same, differing only syntactically. In the robot simulation class however, Filip's solution used no internal state whatsoever. Instead, he effectively used the transition and observation models to transition between states and generate sensor readings. Even though my original approach worked fine, I thought this was more efficient and decided to implement my own version of this simulation approach for the final hand-in.

4 Model Explanation

The notebook skeleton code consists mainly of three models - state model, transition model and observation model. The state model acts simply as a translation tool from coordinates and headings to robot states and sensor readings. It enables every state of the environment to be described with one number as opposed to coordinate and direction values. This, in turn enables us to describe the probabilities of transitioning between different states with a transition matrix. Each entry ij in this matrix describes the probability of transitioning from state i to state j. Lastly, the observation model contains the diagonals of the matrices representing the probability of achieving a certain sensor reading given a certain state. Along with the Markov Assumption and Bayes Rule, this allows us to use the formula

$$f_{1:t+1} = \alpha \, O_{t+1} \, T^T \, f_{1:t} \tag{1}$$

to calculate the forward vector f containing the probabilities of being in a certain state given a sensor reading and previous forward vector.

The notebook also contains code which lets us visually represent the models with heat maps. For the transition model, this visualisation is very easily interpreted as the matrix shape is still present while the probabilities are simply expressed as colors. The observation model on the other hand, is visualised as a matrix where each row is a diagonal of the observation matrix. This is slightly less intuitive, but also shows us a color-coded matrix of probabilities. Here, each row corresponds to a certain sensor reading and each column value at index i to the probability of state i creating this very sensor reading. We can clearly see that the last row of this matrix corresponds to the empty sensor reading, as the probabilities increase for corner states which have less options and are more likely to produce an empty sensor reading.

5 Discussion of results

The tracking results of this approach turned out to be quite limited, as the localizer usually settles at an accuracy of around 0.33 and mean Manhattan distance (MMD) of 1.66. These results are certainly better than pure guessing, which would have an accuracy of 1/64 and MMD of approximately 2.75 given 64 unique states. Also, it should outperform pure sensor readings which yield the true state of the robot 1/10 times and would have to be combined with random guesses, as there is a significant chance that readings will be empty. Assuming that both first and second surrounding ring of the true state are all

valid positions, the MMD for using only sensor data should be around 2. Then again, positions near corners limit the probability of returning any sensor data at all and thus increase the probability of simply guessing the position - so this approach would definitely not do better than the HMM filter.

6 Article Summary

The article provided for this assignment begins by outlining the differences between position tracking and global localization, the former being localization with an initial position and the latter without. It intoduces Markov localization along with Kalman filters and discusses the drawbacks that come with fixed/coarse resolutions or strictly using Gaussians to represent a robots belief. It then goes on to present the premises of Monte Carlo Localization (MCL) and particle filters in general. The key idea of this approach is to represent the robot's beliefs with a set of weighted random samples, which form a discrete approximation of the state probability distribution. When the robot performs an action, new samples are generated to approximate the new position by choosing the most likely samples from the previous set. Sensor readings are used to re-weight the samples with respect to the likelihood of a state yielding a sensor reading. One of the biggest advantages of this technique is the fact that the number of samples can be dynamic, making it very efficient and flexible when it comes to distributing computing power. Lastly, the article claims that the MCL approach outperforms previous implementations of Markov localization while saving major amounts of computational resources.

7 Article Discussion

The MCL approach described in the article seems to outperform Markov localization in pretty much every aspect. Seeing as the accuracy of the current approach for this assignment is about 0.33, there is a lot room for improvement. However, the current solution does not suffer because a of an attempt to discretely describe a continuous environment. The grid-world in which our robot lives in can be fully described by the 256 (assuming a size of 4x4) unique states, which eliminates the need for sampling random states. MCL would certainly be more effective at using resources, but for small grid-worlds this is not really an issue. If we were to scale up the environment, at some point the HMM filter would have trouble keeping up with all the computations. Frankly, this is where I think MCL would be a suitable approach. I believe the reason for the relatively low accuracy obtained with the HMM filter is mostly caused by the erroneous nature of the sensor.