Probabilistic Reasoning Assignment

Erik Dahlberg (er8251da-s)

February 19, 2023

1 Statement

Peer review and implementation discussion were conducted with Daniel Björklund (da1452bj-s).

2 Summary

In this task, probabilistic reasoning based on the hidden Markov model (HMM) and forward filtering was developed to estimate the position of a simulated robot. Probabilistic reasoning with HMM and forward filtering relies on the use of transition, state and observational models. The transition model provides probabilities for transitioning between a current state and all possible next states. The state model creates a virtual space for the robot and has implementations for several useful methods regarding the position and direction of the robot. Lastly, the observation model is a representation of the uncertainty in the robot's sensor. For this assignment, the observation model creates a matrix with probabilities of the robot's location given some sensory input. The EDAP01 course administration provided these models. The assignment task was to develop a method for localizing a simulated Robot by creating a forward filtering algorithm based on HMM while utilizing the three provided models.

3 Implementation

My implementation follows the description and criteria provided in the handout with the assignment. In essence, the program, named "Paul", loops in 4 steps:

1. Based on the robot's current position, the next move is calculated with probabilities from the transition model.
2. The new position is forwarded to a sensor reading algorithm that returns a sensor reading based on probabilities in the observation model.
3. The sensor reading is passed along with the previous localization probability distribution to an HMM filter. This filter calculates an updated probability distribution and returns the state of maximum value.
4. The localizer uses the filter's returned state as its estimated position for the robot and subsequently calculates the average error in Manhattan distance.

4 Peer Review

The first topic of discussion in my and Daniels's peer review was that my implementation could not calculate the robot's next move according to the assignment criteria. Whenever the Robot hit a wall, it would teleport to a seemingly random location on the board. During the peer review, Daniel and I discussed the possible reasons behind the bug. It turned out that I had misunderstood the transition

model's orientation and thus used the transpose of the transition matrix when calculating the next move. Since my implementation uses indexes of the transition matrix to calculate the robot's next move, that move would occasionally break the assignment criteria. Following the peer review, the transpose of the transition matrix is no longer used to calculate the next move.

Secondly, Daniel suggested cases where my implementation was unnecessarily complicated. For instance, my localizer function converted states to positions for inputs to my HMM filter. The filter then immediately translated these positions back to states. These conversions are, of course, unnecessary and are no longer part of these two functions.

Lastly, Daniel helped simplify my sensor reading method. My sensor reading method did not use the observation model prior to our peer review and instead had its own method for extracting an O-matrix. Swapping my implementation for the included observation model, simplified my approach and contributed to better performance overall.

5 Model explanation

The viewer has three modes which depict different metrics across the grid. By pressing "show transitions", the viewer can step through every state and show the corresponding transition probabilities. In essence, it is a visualization of the transition model. By pressing "Show sensor," a similar heat map is displayed over the grid but with values corresponding to probabilities of robot locations given a certain sensor input. This is a visualisation of the observation model. The last set of buttons is for initiating the HMM and forward filtering methods. In this mode, the robot moves according to the transition model, and after each move, the sensor model produces a sensory reading which is passed to forward filtering. This way, an estimation of the location of the robot is derived. A black box denotes the robot's actual position in all three modes. With the sensor model and running filter process running, a blue box indicates the latest sensory reading. In the running filter process, an added red box corresponding to the estimated position of the robot is visible.

Note that if the sensor does not return a value, the observation model and its resulting o-matrix will consist of one value across all indexes. This value will be one divided by the size of the matrix.

6 Results

To test the effectiveness of a probabilistic reasoning implementation, one can compare its results to random guessing. 100 iterations of the algorithm were executed twice for for random guessing and the probabilistic reasoning approach. Using sensory readings, thus overriding the HMM-filter was also tested. The average result of the two random guessing iterations was:

- Average manhattan distance: 5.02
- Number of correct guesses: 1

The average result of the sensory input iterations was:

- Average manhattan distance: 5.04
- Number of correct guesses: 9

The average result of the two probabilistic reasoning iterations was:

- Average manhattan distance: 1.43
- Number of correct guesses: 35

Note that the tests were run on a 6x8 grid. Therefore a manhattan norm average distance of 5 is more than halfway across the grid in any direction. With these results, I conclude that a hidden Markov model works well for this situation. It can successfully predict the robot's location, and the Manhattan norm is relatively small even when the guessing is off. Moreover, it outperforms the

plain readings because it also considers previous readings. A better sensor or filtering would have had to be used to further improve the localisation algorithm's performance. An example of such a filter could be a Kalman filter.

7 Paper summary

In the article "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots", the authors describe a new method for determining the position of a mobile robot in an environment. This approach, called Monte Carlo Localization (MCL), makes use of the robot's sensors and a map of the environment to estimate its location.

MCL was designed to handle the uncertainty inherent in a robot's movements and sensor readings. The method represents the robot's uncertainty about its position as a set of possible locations, with each location represented by a particle. As the robot moves and takes readings from its sensors, the particles are updated to reflect the updated uncertainty about the robot's position. The motion model takes into account the robot's movement, while the sensor measurements provide information about the environment that is used to refine the estimated location.

The authors of the article tested MCL in various simulation scenarios and compared its performance to other common localization methods. The results showed that MCL outperformed these methods and provided more accurate estimates of the robot's position. The authors also integrated MCL with a robotic platform and tested it in a practical setting, further demonstrating its effectiveness.

MCL has wide-ranging potential applications, including autonomous navigation, mapping, and exploration. By providing accurate position estimates in uncertain environments, MCL could help robots navigate through complex spaces with greater ease and efficiency. In addition, MCL could enable more precise mapping and exploration of areas that are difficult or dangerous for humans to access. Overall, MCL has been able to improve the performance of mobile robot localization. It will be interesting to see what other fields MCL can impact.

8 Paper Relevance

The paper is, of course, relevant to the subject of robot localization and thus could influence the performance of my implementation. However, the usefulness of an MCL implementation will be reflected in the complexity of its use case. For our example, the simplicity of a 6x8 grid could make the difference between MCL and HMM negligible. The paper discusses a failure of HMM during a test performed at the Smithsonian museum. This strengthens the need for implementations other than HHM in complex navigation. However, it does not speak for the task in this assignment. It is the author of this paper's opinion that an implementation of MCL could require an excess of resources in comparison to any performance increase.

9 Apendix/Peer Comments

Daniel comment was: "Erik Dahlberg (er8251da-s) had a fully functioning code filling the necessary critera for the completion of this task. Things I noticed in the code which we later on discussed was the consistency with keeping and returning states and readings, and only later when actually necessary to return position, to make the transfer. Another thing I noticed which was not wrong, was that he had an own function to calculate max, where np.argmax would fit equally good. Lastly, we noticed some unpredicted behaviour when examining the code, this was due to accidentally multiplying with the transpose of T and not T in filtering step. This was later fixed.

Performance when doing 100 steps was as expected, with an average manhattan distance of 1,43 with 34 correct guesses."