EDAP01 - Assignment 3

da7308ca-s

February 2020

1 Introduction

The objectives of this assignment was to implement robot localization/tracking based on forward filtering with a hidden Markov Model

2 Task description

The task consist of being able to as accurately as possible track the location of a robot in a grid. The robots location is described with a state vector consisting of it's x and y coordinate as well as it's heading (north, west, south, east). Every time step the robot sends a reading of it's current position. This reading can be effected by gaussian noise, offsetting the true measurement by some distance. We are most likely to get the true measurement with a probability of 0.1. The probability of getting a false measurement offset one step is 0.05 for every square and getting a reading 2 steps of is 0.025 each. We can also get a nothing reading with a probability of one minus the probabilities of getting all the other readings. Note this also means we are more likely to get nothing readings when the robot is closer to the wall.

Every time step the robot makes a move to one of the neighbouring squares using the following strategy:

Pick random start heading h_0. For any new step pick new heading h_{t+1} based on the current heading h_t according to:

P($h_{t+1} = h_t$ | not encountering a wall) = 0.7
P($h_{t+1} \neq h_t$ | not encountering a wall) = 0.3
P($h_{t+1} = h_t$ | encountering a wall) = 0.0
P($h_{t+1} \neq h_t$ | encountering a wall) = 1.0

It then moves in the direction h_{t+1} by one step in the grid. This means essentially that it a) will always move one step and b) it can only move straight.

In case a new heading is to be found, the new one is randomly chosen from the possible ones (facing a wall somewhere along the wall leaves three, facing the wall in a corner leaves two options of where to turn, see one example below in hint 1).

The objective is to given our knowledge of how the robot moves and how the sensor readings are given to track the robot as good as possible.

3 Implementation

The basic idea of forward filtering is that we want to get a better approximation of where the robot is given our observations. This is done through continuously updating the the f vector which tells us the probabilities of the robot being in a certain state according to:

$$f_{t+1} = \alpha O T^T f_t \tag{1}$$

where T is the transition model, O a matrix from the observation model and α a normalizing constant.

3.1 Transition model

The transisiton model is a matrix which encodes the probabilities of going from one state to another. That is $T_{i,j}$ tells us the probability of going from S_i to S_j. In this implementation we have a total of $xrows*cols*directions$ states, meaning our transisiton matrix is a $(rows*cols*directions)$ by $(rows*cols*directions)$ matrix.

3.2 Observation model

For the observation model i implemented one matrix for every possible reading. That is $(rows*cols+1)$ different readings. For each reading we create a diagonal $(rows*cols*directions)$ by $(rows*cols*directions)$ matrix where the diagonal elements tells us the probability getting the reading we got given that we where in a specific state.

3.3 Normalizing constant

Lastly in our feed forward algorithm we multiply with a normalizing constant α to make sure that the sum of f always stays equal to one.

4 How to run code

To run the code simply run the Main.java file located in the control folder within the zip-file. To change the size of the grid adjust the parameters when creating the Localizer in the main method.

5 Results

I experimented with tracking the robot in 4 different grid sizes of which the results can be seen in the figure below. To evaluate my implementation i measure the manhattan distance between the robots true position and my best estimate. This can be seen in figure 1. To get a feeling for just how good this is I also compare our tracking to just randomly guessing the robots position every time step. This is visualized in figure 2. Lastly I also compare just the sensor readings with the true position. This can be seen in figure 3

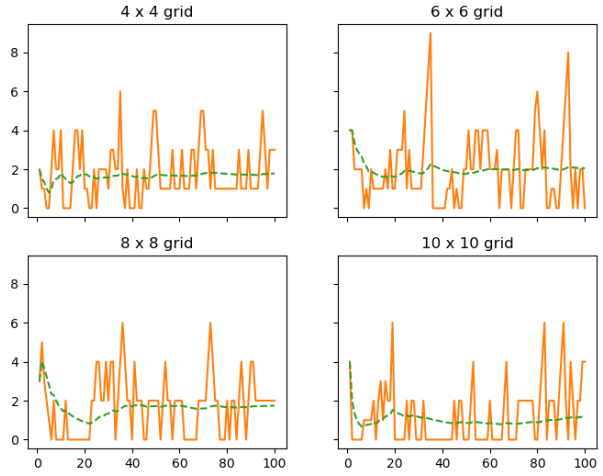

Figure 1: The orange line represents the current manhattan distance between the true position and our best estimate that time step. The green line shows the average manhattan distance up until that time step.

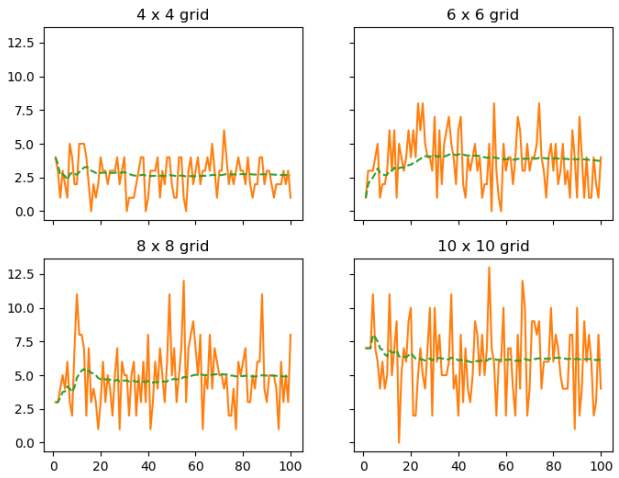

Figure 2: The orange line represents the manhattan distance between the true position and a random guess. The green shows the average up until that point

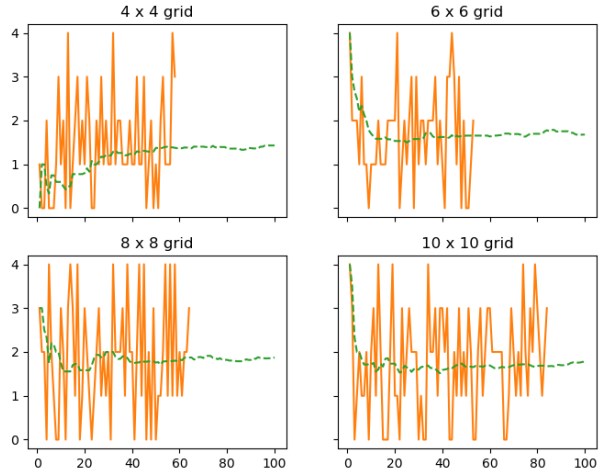

Figure 3: The orange line represents the manhattan distance between the latest sensor reading and the true position. The green line the average manhattan distance. Note that the orange line ends before the green one. This is due to getting "nothing" readings from the sensor at which point I did not calculate the distance

6 Discussion

We can see that for all the different grid-sizes we are capable of tracking the robot quite well. After just about 20 steps the average has fallen below 2 for all the grid-sizes.

It seems to be easier to track in the bigger grid-sizes. This probably has to do with the fact that for the larger grid-sizes we have a lower corner/edges square to non corner/edges squares ratio meaning we are less likely to get a nothing reading and the tracking therefore works better.

If we compare the results from figure 1 with figure 2 and 3 we see that our estimation greatly outperforms just guessing the position, but when we compare it to the sensor readings we just get a marginal improvement.

7 Dissertation

The article **Monte Carlo Localization: Efficent Position Estimation for Mobile Robots** presents a new localization method **M**onte **C**arlo **L**ocalization (in short: MCL). MCL uses fast sampling technique to represent the robots beleif. This gives it 3 main advantages:

- In contrast to existing Kalman filtering based techniques it is able t represent multi-modal distributions and thus can globally localize a robot
- It drastically reduces the amount of memory required compared to grid-based Markov Localization and can integrate measurements at a considerably higher frequency
- It is more accurate than Markov localization with a fixed cell size, as the state represented in the samples is not discretized.
- It is much easier to implement

The article then tells us how normal Markov Localization works which is it based on. The main idea is to update the our belief of where the robot is according to equation 1.

The key difference for MCL is that we represent the posterior beliefs by a set of N weighted random samples. A sample set constitutes a discrete approximation of a probability distribution.

MCL uses few samples when tracking the robot's position, but increases the samples set size when the robot loses track of it's position, or is otherwise forced to globally localize the robot.

8 Is the HMM approach as implemented suitable for solving the problem of robot localization?

As we can see from our results the implemented HMM does a good job of tracking the robot. The grid-sizes we have been using have however been relatively small and for bigger ones the computation time would increase and it would therefore be reasonable to look at other solutions because of this.

My implementation is also incredibly memory consuming and would be greatly outperformed by the MCL in that regard.

My implementation is also not suited for global tracking and would there be outperformed by the MCL.