Artificial Intelligence

Assignment 2

Probabilistic reasoning over time

Cullen de Erquiaga, Magdalena Itziar

Mutz, Matias

Exchange Students

LUND UNIVERSITY

LTH

FACULTY OF ENGINEERING

February 24, 2025

Contents

Statement

We collaborate with Group 2 (Franco Ghigliani & Lucas Campoli). They helped us a lot by improving our implementation. Our implementations were similar but the main difference we noticed was that theirs was shorter, while being just as effective as ours. Another difference was our forward-backward algorithm in filters.py file, while they have implemented it on another file. Both approaches are correct, but we believe our approach is better for future maintainability, code reuse and improvements.

Lucas and Franco helped us when comparing results, they identified that our method for plotting results was incorrect. We were not calculating the average error comparisons. Furthermore, they also pointed out we were comparing the different models. We were using two different localizers to compare the filters, when we should have been using the same true trajectory. We corrected both of these errors, and now our model comparisons are accurate.

Summary of the task

For this assignment, we implemented and evaluated different filtering methods using the Hidden Markov Model(HMM) localisation approach to estimate the robot position in a grid environment. The robot can move in four different directions, and a sensor provides the approximate location of the robot.

The main challenge is that the sensor is not always reliable; it sometimes gives incorrect readings, or no readings at all. This is where the HMM-based localization plays a role in the approximation of the localisation of the robot. Using the different filtering approaches from the HMM-model to estimate where the robot was. The first is a forward filtering, which uses the current sensor reading and the robot´s movement model to estimate its position. We implemented a more complex algorithm, forward-backward smoothing, which uses a sequence of five past (forward) and future (backward) observations to improve predictions.

An interesting aspect of this assignment was testing these methods under different conditions. We used two types of sensor observation models: the Uniform Failure (UF), where the sensor reading probabilities are equally distributed in all positions (0.1) in the grid, and the Non-Uniform Failures (NUF), where sensor reading probabilities depend on the robots specific location.

The results showed how each filtering method performed differently depending on the grid size and the observation model used. We measured their performance using the Manhattan distance between true and estimated location, and tracked the frequency of hitting the robot's location, as well as, tracking the no sensor readings.

Explanation

There are three models: State, Transition, and Observation.

The State model describes the dimensions of the grid and the methods for transforming a pose or position of the robot into a certain state while collecting the sensor readings at that time.

The Transition model contains the transition matrix containing the probabilities of transitioning from one hidden state to another.

The Observation model defines probabilities of receiving sensor reading in a given hidden state. There are two types of sensors, each handling failure to read a location differently: the Uniform Failure(UF) probability and the Non-Uniform Failure(NUF) probability.

The NonUniform Failure model works like this: if the robot is near a wall, it will have fewer cells to detect around it, which affects the probability of getting a "*nothing*" reading.

Case 1 - Corner Position:	**Case 2 - Center Position:**

Ls1 (adjacent cells) = 3 pos. (marked with 1)
Ls2 (secondary cells) = 5 pos. (marked with 2)
 	
`2 2 2 # #`
`2 1 1 # #`
`2 1 X # #`
`# # # # #`
`# # # # #`
	Ls1 (adjacent cells) = 8 pos. (marked with 1)
	Ls2 (secondary cells) = 16 pos. (marked with 2)
	
	`2 2 2 2 2`
	`2 1 1 1 2`
	`2 1 X 1 2`
	`2 1 1 1 2`
	`2 2 2 2 2`

Table 1: Comparison of sensor readings in corner and center positions

If the robot is in the corner, it only has 3 adjacent cells and 5 secondary cells, so the probability of reading "*nothing*" is:

$$\begin{aligned} P(\text{nothing}) &= 1.0 - 0.1 \quad \text{(correct reading)} \\ &\quad - (3 \times 0.05) \quad \text{(adjacent cells)} \\ &\quad - (5 \times 0.025) \quad \text{(secondary cells)} \\ &= 0.7375 \quad \text{or } 73.75\%. \end{aligned}$$

If the robot is in a center position, it has 8 adjacent cells and 16 secondary cells, so the probability of reading "*nothing*" is:

$$\begin{aligned} P(\text{nothing}) &= 1.0 - 0.1 \quad \text{(correct reading)} \\ &\quad - (8 \times 0.05) \quad \text{(adjacent cells)} \\ &\quad - (16 \times 0.025) \quad \text{(secondary cells)} \\ &= 0.1 \quad \text{or } 10\%. \end{aligned}$$

On the other hand, the Uniform Failure model treats all failures in the same way, having a constant 0.1 probability of failing to read. So if the sensor fails, we do not have any additional information about the robot's location.

The difference between failure probability affect greatly the filtering algorithm In the NUF sensor, a "nothing" reading provides valuable information by suggesting the robot is likely to be near a wall or corner where there are fewer cells around it, rather than in open spaces where there are more cells around.

In contrast, the UF sensor's constant 10% of failure probability means that "*nothing*" readings provide no information at all about where the robot is. This forces the filter to only rely on the sensor readings for localization. If there are sequences of "*nothing*" readings, it is very unlikely that the position of the robot can be guessed. This is why the NUF model is generally more accurate than the UF model, because it gives more information when failing to sense.

Discussion

How accurately can you track the robot with filtering (or smoothing), is it even worth doing that?

As time passes the probability of not detecting the robot's position decreases. The filtering accuracy improves over time, meaning the estimate gets closer to the true position the longer the process runs. The accuracy of the filtering and smoothing also depends on the grid size and the failure probability taken (uniform or non-uniform). However, sensor failures, when no readings are obtained, the performance is impacted. In cases of frequent failure the worse the model is in accurately finding the position.

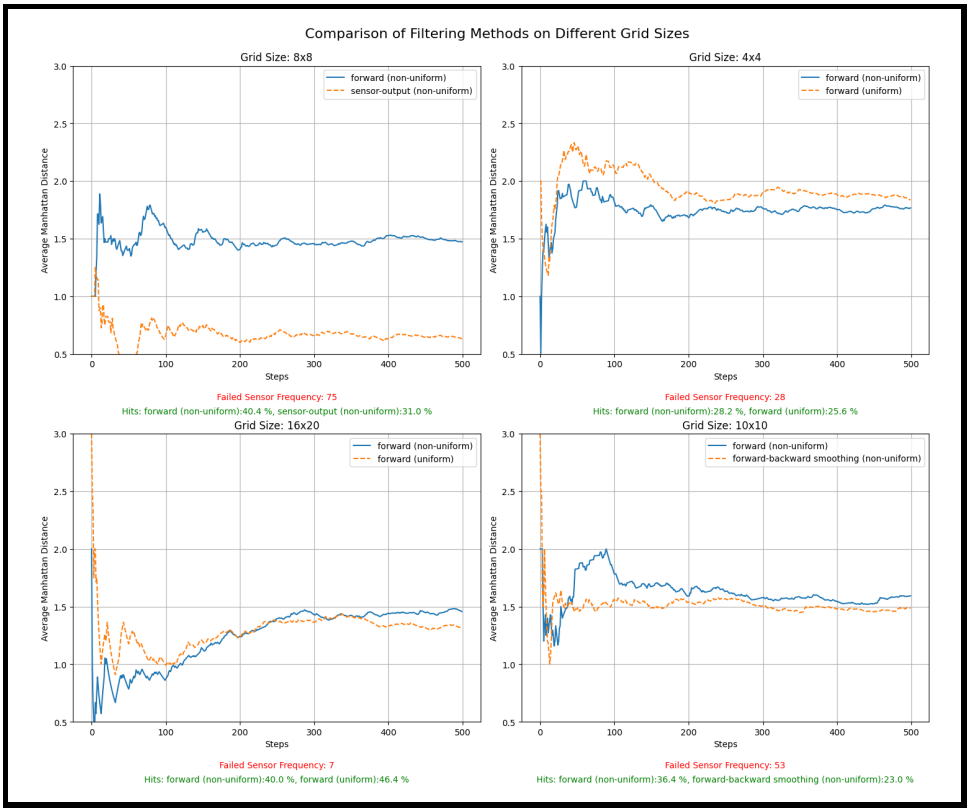

Figure 1: Comparison of filtering methods under uniform and non-uniform failure models across different grid sizes.

As seen in the figure 1, in smaller grids, the difference between the effect of the uniform failure (UF) and non-uniform failure (NUF) is more noticeable, observable in the 4x4 graph where the NUF hits 28.2% whereas the UF hits 25.6%. The NUF model performs better as it provides more details over where the robot is on the grid. However, in larger grids, the difference between failure models is less significant since there are more locations for the robot, making the additional details from NUF less impactful, seen in the 16x20 graph where in fact the UF hit 46.6% and the NUF hit 40%. Thus, the smaller the grid the more detailed the failure model the better the chance of hitting the location.

In addition, as time passes, tracking improves as the filtering process accumulates observations. However, in the 10x10 grid graph, the comparison uses the forward-backward model, which estimates based on past and future observations. In the graph, we can see that the forward-backward smoothing has a lower rate and will continue to behave this way over time, providing less fluctuation, more stability, and more accurate results. In addition, the performance was affected by the sensor failures, which were high (53 readings), impacting the overall performance. In conclusion, filtering is beneficial since it improves localisation over time. The effectiveness of the different models depends on grid size and failure probabilities. As expected, the non-uniform failure model generally performs better as it provides more detailed probability distributions, especially in the smaller grids.

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

The Hidden Markov Model (HMM) approach is suitable for solving theoretical localisation, but in the practical applications there are more factors that must be considered. The accuracy of the HMM-based model depends on the grid size, sensor reliability, the precision of the transition and observation models.

Regarding filtering and smoothing, both approaches improve localization over time. However, smoothing improves greatly in the number of observations, making it computational complex and expensive. The failure models used also affect the outcomes. In smaller grids, there is a noticeable difference between the uniform failure(UF) and non-uniform failure(NUF), with the non-uniform representing the sensor behaviour more accurately. This makes the filtering and smoothing dependent on the failure model. While

using larger grids the advantage of the NUF decreases since the number of possible positions increases, therefore, the smoothing and filtering become less sensitive to the failure model used.

Even though the filtering and smoothing improves over time, this makes their computational cost scale with the grid size, making it impractical for large environments. In conclusion, the HMM approach is effective for small to medium sized grids but for larger environments it's better to consider an alternative approach.

One alternative is the Monte Carlo Localisation (MCL), seen in the paper "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots" by Fox et al. presents Monte Carlo Localization (MCL). MCL uses particle filters, which are "fast sampling techniques to represent the robot's belief."(Fox et al., 1999). The key advantages, when compared to the grid-based markov localization, include more efficient use of memory, simpler implementation, and the ability to represent multi-modal distributions, making it capable of globally localizing a robot. The MCL approach uses adaptive sampling making the model have to analyze less data while maintaining effectiveness. This adaptability leads the robot to incorporate at a higher frequency the sensor data thus making the localisation more suitable for large environments compared to the HMM approach.

On the other hand, the markov localization algorithm is computed over the "probability distribution over all possible positions in the environment"(Fox et al., 1999) hence the algorithm is better suited for smaller grids because the computability becomes too complex in large environments and inefficient. Furthermore, the paper explains the update method: "are only applicable if the environment is Markovian, that is, if past sensor readings are conditionally independent of future readings given the true position of the robot."(Fox et al., 1999) This assumption limits the model's effectiveness in non-markovian environments.

To conclude, the HMM approach is good for theoretical robot localisation in small and medium grid environments but it's impractical for large spaces. While the MCL is better for scalability and efficiency. Therefore, for real-world applications MCL is the better choice.

References

Fox, D., Burgard, W., & Thrun, S. (1999). Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *Proceedings of the 16th AAAI Conference on Artificial Intelligence (AAAI-99).*