ASSIGNMENT 2

Probabilistic reasoning over time

Måns Alklint
ma7230al-s

Måns Alklint
ma7230al-s

Statement

I regret that I was unable to complete the peer review for this lab assignment due to personal circumstances before and during the assignment's original hand-in date. Unfortunately, my family, especially my parents, experienced a significant loss when my aunt passed away. I received the sad news shortly before the submission deadline, and the emotional and logistical challenges that followed made it difficult for me to focus on my academic responsibilities. I apologise for any inconvenience this may have caused and deeply appreciate the understanding and support I received from this course's teachers during this time.

Summary

A Hidden Markov Model (HMM) is a statistical model used to represent systems where the underlying state is not directly observable (hidden) but can be determined through the observable data. The model consists of a set of states connected by probabilities. HMMs are used in applications such as speech recognition, bioinformatics, and in this case, robot navigation, where the goal is to infer the hidden state from the observed data.

In this task, an HMM was applied to track the position of a robot on a grid over time. The robot moves around the grid according to its transition model, and its true position is hidden. At each time step, a sensor provides a noisy observation, or rather an approximation, of the robot's position. The objective is to use the sequence of these noisy observations to determine the robot's most likely position at each time step.

The two sensor/observation models provided represent different types of sensor failure modes. The "0" model represents a non-uniform failure where certain grid positions are more likely to be misread by the sensor. The "1" model represents a uniform failure where all positions have an equal probability of being misread.

The difference between these two models affects the accuracy of the position tracking when doing the HMM-filtering-approach. With the non-uniform failure model, the algorithm can learn that certain positions are unreliable which in short will deprioritise them. This results in more accurate tracking than with the uniform failure model. The difference in accuracy is small but still notable which will be shown in the provided diagrams which describes the failure-plot as the Manhattan distance between the guessed position from the model and the actual position.

Based on my results, tracking the robot's position using filtering provides much better accuracy than pure guessing with no sensing. Filtering outperforms the raw sensor observations as it effectively reduces the impact of sensor noise and inaccuracies.

The HMM approach is well-suited for this robot localisation task because it models the hidden state (the true position) and relationship between state and observations (sensor model). This allows inferring the hidden state from the sequence of observations. The HMM makes reasonable assumptions about the robot's motion and sensor characteristics that makes the position tracking with relatively low failure.

Models

TransitionModel

This model contains rules for how the robot changes its direction/heading when moving through the environment.

If the robot does not encounter a wall on its current move, there is a 0.7 probability that it will continue in the same heading on the next move, and a 0.3 probability that it will change to a different, random heading. However, if the robot does encounter a wall on its current move, there is a 0.0 probability that it will continue in the same heading (since it cannot move through the wall). Instead, there is a 1.0 probability that it will change to a different, random heading that does not involve the wall.

The reason for having these probabilistic rules is to make so that the robot has some randomness or unpredictability in how it moves, rather than moving in a completely deterministic way. The robot first "decides" whether to change heading or not based on the probabilities, and then actually takes a step in that new or same heading. This simulates the robot first processing sensor data about walls, then updating its heading, then making a move in that direction.

StateModel

This model is for representing the positions and orientations of the robot in the grid environment. The "state" represents a specific pose, which combines the x and y position coordinates along with the direction/heading of the robot. The direction is represented by an integer value 0, 1, 2, or 3, which corresponds to the directions North, South, East and West. The state numbering starts from 0, which represents the pose x=0, y=0 and h=0, meaning the object is at the top-left corner of the grid, facing South. The states are numbered such that the heading value cycles through the 4 so called “cardinal directions” for each X and Y position.

In addition to states representing poses, there are also "sensor readings" which represent just the X and Y position, without the heading information. The purpose of this model is to provide a way to identify and encode the pose and position of the object within the environment. This is useful for the localisation.

This model also provides a way to transform between pose (x y h) and state (X Y) representations, as well as between position and state to sensor reading representations.

ObservationModel_NUF

In this model, the probabilities are fixed and predetermined based on the sensor's position. The model assigns a 0.1 probability to the sensor correctly identifying its exact position. For positions in the immediate surroundings, this probability drops to 0.05, and it further decreases to 0.025 for positions in the secondary surrounding areas. The probability of the sensor reporting "nothing" is calculated based on what remains after these probabilities are allocated. This approach does not adapt to specific grid configurations, meaning it applies the same rules whether the sensor is in the centre of the grid or at an edge.

ObservationModel_UF

In contrast, this model adopts a more dynamic approach by adjusting failure probabilities according to the number of surrounding positions. While it also assigns a 0.1 probability for correct identification, it distributes the remaining probability uniformly across the surrounding positions based on their count, both in the immediate and secondary rings. This model reserves a constant 0.1 probability for the sensor reporting nothing, regardless of the actual state. It also explicitly accounts for edge and corner cases, adjusting the probabilities to ensure consistency even when the number of neighbouring positions varies. The ObservationModel_NUF offers simplicity and predictability, making it better for uniform environments, while the ObservationModel_UF provides greater adaptability, which could be advantageous in some settings, though with some variating performance.

Results

The various filtering methods applied to robot localisation reveals some important insights into their effectiveness. When using only the sensor outputs, the robot's estimated position frequently deviates significantly from its true position, as shown by the high variance in the Manhattan distance over time. This indicates that relying solely on sensor data results in considerable errors due to the noise and inaccuracies of the sensors.

Forward filtering, however, improves the situation by providing a more accurate estimate of the robot's position. This method helps mitigate some of the sensor's inaccuracies, keeping the Manhattan distance relatively low most of the time. Nonetheless, occasional spikes in the distance suggest that forward filtering alone is not entirely immune to the effects of noise or sensor failures. The model that accounts for non-uniform sensor failure appears to produce more stable results, reducing the impact of sensor inaccuracies and leading to a more consistent localisation performance compared to the uniform sensor failure model, which shows slightly more variability in the robot’s estimated position.

The introduction of smoothing enhances the accuracy of localisation furthermore. Smoothing corrects for both sensor noise and the errors that might occur during the forward filtering process, resulting in fewer and less severe deviations in the estimated position. This method, which uses future information to refine the estimates, provides a more accurate and stable prediction of the robot's location over time.

This highlights that while forward filtering improves localisation accuracy over using sensor data alone, the best results are obtained when it is combined with smoothing. The combination can handle uncertainties and reduces the impact of noise, leading to the most reliable and consistent localisation performance.

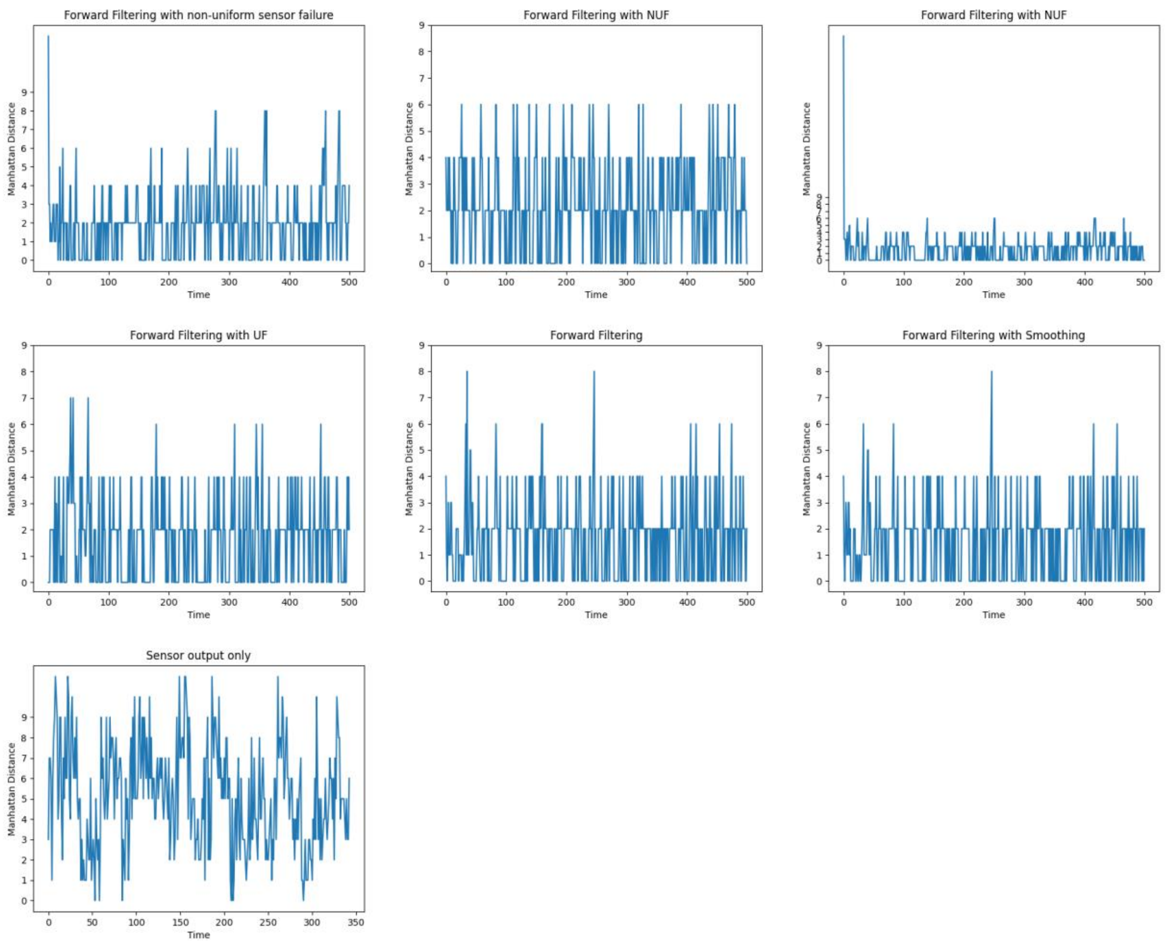

Figure 1: *The results from different filtering and smoothing techniques for robot localisation. The figure compares the Manhattan distance between the estimated and true positions over time using various methods: raw sensor output, forward filtering (with both uniform and non-uniform sensor failure models), and forward filtering combined with smoothing.*

Comparing HMM and MCL

The HMM approach implemented for the robot localisation assignment is effective for basic grid-based environments with noisy sensor data. The use of filtering and smoothing techniques based on HMM in this assignment enhances the accuracy of the robot's position estimates and the probability distributions over time. HMM's flexibility also makes it adaptable to various robot localisation challenges, such as different sensor inputs, complex terrain, or uncertainty.

In contrast, the article presents Monte Carlo Localisation (MCL) as a preferable alternative to grid-based Markov localisation, particularly in more complex or real-world scenarios. MCL offers a flexible and computationally efficient approach to localisation by using sampling to approximate the robot's belief distribution. Therefore, MCL can better handle environments with high uncertainty or dynamic conditions. This adaptive sampling strategy also allows MCL to allocate computational resources, according to the article, where they are most needed, providing a significant advantage in performance and scalability compared to HMM.

While HMM is a solid choice for this specific localisation task, MCL could offer enhanced efficiency and adaptability, making it a better fit for more demanding localisation tasks in real-world applications. MCL has proven effective in global localisation and in recovering from extreme localisation errors, consistently demonstrating superior performance compared to grid-based methods, particularly in terms of accuracy, memory usage, and computational efficiency. This is especially true in challenging environments and when dealing with complex data sources, such as vision data.

Conclusion

The Hidden Markov Model (HMM) approach implemented in this robot localisation assignment has proven to be effective for managing noisy sensor data in a grid-based environment. The use of filtering and smoothing techniques significantly enhances the accuracy of position tracking, demonstrating the HMM's ability to handle the uncertainties in robot localisation.

However, when considering more complex or real-world scenarios, Monte Carlo Localisation (MCL) could be a more powerful alternative. MCL's ability to handle environments with high uncertainty through adaptive sampling offers clear advantages in terms of accuracy, computational efficiency, and scalability. The article highlights MCL's superiority in challenging conditions, particularly in scenarios that involve dynamic environments or complex data inputs.

While HMM provides a solid foundation for the specific localisation task addressed in this assignment, MCL's proven effectiveness in global localisation and its ability to recover from extreme errors make it a choice for more demanding applications. While HMM serves well in this controlled scenario, MCL would likely outperform it in more diverse and unpredictable environments.