Occupancy Gridmaps

AI and Robotics - Lab 02

Giacomo Gaiani - giacomo.gaiani@studio.unibo.it

Submission Date: 23/04/2024 Submission Number: 2

Contents

1 Task 1: Implementation of the sonar-based occupancy gridmap using probability theory

The description of the sections of the program will not cover the parts already described in the previous laboratory exercise, if not modified for this specific task.

1.1 Problem Description

The objective of this first task is to implement the sonar sensors reading and storing mechanism, and the sonar model, using as uncertainty the probability theory to create an occupancy gridmap of the simulated environment. This is done by modifying the python files provided for the laboratory activity [1]: `robot_gtw.py` and `observer.py`, and updating them with the implementations performed during the execution of the previous lab exercise.

1.2 Objects and Data Structures

Most of the sub-tasks encountered in the resolution of the problem are solved by modifying the existing functions and classes present in the python files or designing new ones. Specifically, the implementation of a new function in `robot_gtw.py`, called *get_sonar_data()* to read the sonar sensor data from the simulated tiago robot, and the adjustment of the *Sonar_Model()* class to correctly calculate the support and fusion strategy.

1.3 Control and Information Flow

The operational structure of the program can be divided in two sections, as the resolution of the task is performed in two different steps. The first one is the refers to the gathering of the sonar and position data. The program `toplevel.py` has been extended in order to allow the program to run in different modes, namely:

- 'DEFAULT' to move the robot following a specified movimentation mode
- 'SAVE_DATA' to simply store the sonar and position data

- 'COMP_GRID_FROM_DATA' to retrieve the stored sonar and position data and compute the occupancy gridmap.

The mode used in this first step is SAVE_DATA. The sonar sensor reading, performed in `robot_gtw.py`, is implemented using multiple rospy subscriber, all linked to a different topic, reading in parallel thanks to the used of a corresponding number of threads. The position of the robot is retrieved either via odometric calculations using the wheels encoder information or via the `ground_truth_odom` topic. The sonar data and the relative of the robot are added to a list that is saved as a local file after a specified amount of reading step. The sleep mechanism has been modified so that the single reading step last at least the specified amount of second, instead of waiting a fixed amount of time. During this phase, the motion simulated robot was controlled using the *key_teleop* command. The second section of the program refers to the iterative computation of the occupancy gridmap using the data gathered in the previous step. This way of separation allows to save time, having to collect the data only once, and reduce the energy consumption as the gridmap computations are performed without the need of the simulated environment. This section has been performed by modifying the *prior*, *support* and *fuse* methods of the *Sonar_Model()* class defined in the `observer.py` file. The prior probability was set to 0.5 for both occupied and empty. The support was set in two different ways:

$$P_1(Occ|r) \begin{cases} \varepsilon & if\,|\varphi| \leq \Delta\, and\, \rho < r - \delta \\ 1.0 - \varepsilon & if\,|\varphi| \leq \Delta\, and\, |\rho - r| \leq \delta \\ 0.5 & if\,|\varphi| > \Delta\, or\, \rho > r + \delta \end{cases}$$

This first method assign uniform probability inside the different areas of the sensor reading. In order to give a more precise estimation i developed a second support function that takes in consideration also the values of φ and ρ:

$$P_2(Occ|r) \begin{cases} f(r,\rho,\Delta,\varphi)\cdot\varepsilon & if\,|\varphi| \leq \Delta\, and\, \rho < r - \delta \\ f(r,\rho,\Delta,\varphi)\cdot(1.0 - \varepsilon) & if\,|\varphi| \leq \Delta\, and\, |\rho - r| \leq \delta \\ 0.5 & if\,|\varphi| > \Delta\, or\, \rho > r + \delta \end{cases}$$

where the function $f(r, \rho, \Delta, \varphi)$ is defined as:

$$\left(\frac{|r - \rho|}{r} + \frac{\frac{\Delta}{2} - |\varphi|}{\frac{\Delta}{2}} \right) \cdot \frac{1}{2}$$

Every time $P(Occ|r)$ is computed, the value of $P(Ept|r)$ is updated as $1.0 - P(Occ|r)$ At each step, the support is computed for each cell of the gridmap. After that, the values are fused with the previous support probability. The fusion mechanism follows the Bayes probability theory and is defined as:

$$P(Occ) = \frac{P(Occ|r) \cdot P(Occ)}{(P(Occ|r) \cdot P(Occ)) + (P(Ept|r) \cdot P(Ept))}$$

$$P(Ept) = 1.0 - P(Occ)$$

After all the data entries are processed, the gridmaps are store as local files in the *.pgm* format. During this last phase, the values of the single cells that represents probabilities in the range [0,1] are multiplied by 100, in order to be stored as percentage and the number of levels of the *.pgm* are set to 100 as well. This allows to appreciate small differences in the developed approaches.

1.4 Parameters

The parameters of the robot introduced for this laboratory exercise are relative to the sonar sensor (maximum range, angle of reading, relative positions of the sensors with respect to the robot) and the gridmap (map size, offset and cell size). The map size and offset have been modified in order to better fit the shape and dimension of the simulated environment. The remaining parameters are relative to the support functions, namely ε and δ. The first was set to 0.10 for both P_1 and P_2, while the second was set to 0.05 in the first case and 0.15 in the second case. A more detailed observation on the tuning of these parameters is addressed in the chapter regarding the third task.

1.5 Results

Figure 1: Occupancy gridmap obtained using $P_1(Occ|r)$, ε = 0.10, δ = 0.05 and using computed robot position

Figure 2: Occupancy gridmap obtained using $P_1(Occ|r)$, ε = 0.10, δ = 0.05 and using ground truth robot position

Figure 3: Occupancy gridmap obtained using $P_2(Occ|r)$, ε = 0.10, δ = 0.15 and using ground truth robot position

The figure 1 shows the occupancy gridmap obtained using as support strategy the probability distribution $P_1(Occ|r)$. It show poor precision regarding both inner and outer walls. Because the robot position data was obtained via state estimation using omometric calculation, the resulting gridmap is affected by noise due to misalignment between the computed position with respect to the real position of the robot. Additionally, because the state estimation is computed after the retrieval of the sonar data, that in this particular examination requires between 1 and 3 seconds, the state estimation will result highly imprecise due to the large variation of the wheels encoder position at each step. On top of that, the misalignment is aggravated by the length of the data gathering operation that lasts for 500 steps. Because of this, all the successive occupancy gridmap are computed using as position data the ground truth position of the robot. The figure 2 shows the occupancy gridmap obtained using as support strategy the probability distribution $P_1(Occ|r)$. It shows good precision in both outer and inner walls detection, with little noise in the empty areas. The figure 3 shows the occupancy gridmap obtained using as support a more complicate strategy, $P_2(Occ|r)$. This second map shows little less noise in the empty areas but has less precision in occupied areas, especially the inner walls. This is due to the fact that using this second technique the probability of a cell to be occupied raises more slowly. Still, in both maps is possible to see erroneous reading of empty ares outside the outer walls, given by mistakes in the sensor reading procedure.

1.6 Conclusions

The main limitation of this approach lies in the quality of the sensor readings: the file used to compute the gridmaps contains 500 data entries, but the reading were taken at a variable interval (between 1 and 3 seconds) while the simulated robot was moving. This may inject inconsistencies in the data and cause the mistakes visible in the figures. A way to solve this problem would be to stop the robot at every step in order to do the the sensor reading, or to speed up the latter part.

2 Task 2: Implementation of the sonar-based occupancy gridmap using fuzzy set theory

2.1 Problem Description

The objective of this second task is to implement a second support strategy using another uncertainty theory, in this case fuzzy sets.

2.2 Objects and Data Structures

The objects and data structures are the same as the previous task, with the sole exception of the *Sonar_Model()* class, which was modified to represent a different uncertainty theory.

2.3 Control and Information Flow

The overall structure of the program is similar to the previous task, so it is easier to list the changes following the execution progress. In the `observer.py` file, the *Sonar_Model()* class was renamed *Sonar_Model_Fuzzy()* and *prior*, *support* and *fuse* methods modified. The prior values for both occupied and empty were changed to 1.0. The new support strategy is defined as:

$$\mu_s(Occ) \begin{cases} 1.0 - f(r, \rho, \Delta, \varphi) & if\ |\varphi| \leq \Delta\ and\ |\rho - r| \leq \delta \\ 1.0 & otherwise \end{cases}$$

$$\mu_s(Ept) \begin{cases} 1.0 - f(r, \rho, \Delta, \varphi) & if\ |\varphi| \leq \Delta\ and\ \rho < r - \delta \\ 1.0 & otherwise \end{cases}$$

where the function $f(r, \rho, \Delta, \varphi)$ is defined as:

$$\left(\frac{|r - \rho|}{r} + \frac{\frac{\Delta}{2} - |\varphi|}{\frac{\Delta}{2}} \right) \cdot \frac{1}{2} \cdot (1.0 - \varepsilon)$$

The fusion mechanism is equal to:

$$\mu_c(Occ) = \mu_{s1}(Occ|r) \otimes \mu_{s2}(Occ)$$

$$\mu_c(Ept) = \mu_{s1}(Ept|r) \otimes \mu_{s2}(Ept)$$

where $\otimes$ is implemented both as the minimum or as the product of the two factors.

2.4 Parameters

The parameters relative to the sonar sensor and the gridmap are the same of the previous task. The parameters relative to the support function, ε and δ, are both set to 0.02 in the minimum case, while in the product case ε is set to 0.80. A more detailed observation on the tuning of these parameters is addressed in the chapter regarding the third task.

2.5 Results

Figure 4: Occupancy gridmap obtained using $\mu_s(Ept)$, $\otimes = min()$, ε = 0.02, δ = 0.02 and using ground truth robot position

Figure 5: Occupancy gridmap obtained using $\mu_s(Occ)$, $\otimes = min()$, ε = 0.02, δ = 0.02 and using ground truth robot position

Figure 6: Occupancy gridmap obtained using $\mu_s(Ept)$, $\otimes = product$, ε = 0.80, δ = 0.02 and using ground truth robot position

Figure 7: Occupancy gridmap obtained using $\mu_s(Occ)$, $\otimes = product$, ε = 0.80, δ = 0.02 and using ground truth robot position

The figures 4 and 5 shows the occupancy grid obtained using fuzzy sets as support strategy and using the $min()$ functions as fusion strategy. The figures 6 and 7 shows the occupancy grid obtained using fuzzy sets as support strategy and using the product as fusion strategy. For this particular application, there seems to be an advantage in using the product as fusion mechanism. Both $\mu_s(Occ)$ show little support in the empty spaces, but using the product as fusion mechanism enhances the ability to isolate the inner walls, while both $\mu_s(Ept)$ show defined inner wall, outer walls and the tables in the bigger room, as well as noise in the empty spaces.

2.6 Conclusions

The main limitation of this approach is analogous to the one discussed in the first task. The two defined models have a similar level of complexity. Having implemented one, it is simple to add a second, as it only requires to modify the *prior*, *support* and *fuse* methods of the *Sonar_Model()* class.

3 Task 3: Sonar model parameters fine-tuning

3.1 Problem Description

The objective of this last task is to explain the reasoning behind the values of the parameters used in the different models covered in the previous chapters.

3.2 Parameters

The two parameters tuned to get better visual results in the gridmap generation are ε and δ. The first one represents the uncertainty in the computed value of single cell, while the second represent the "thickness" of the area of the sensor cone that is considered occupied. The altering of both changes the resulting gridmap in different ways.

3.3 Results

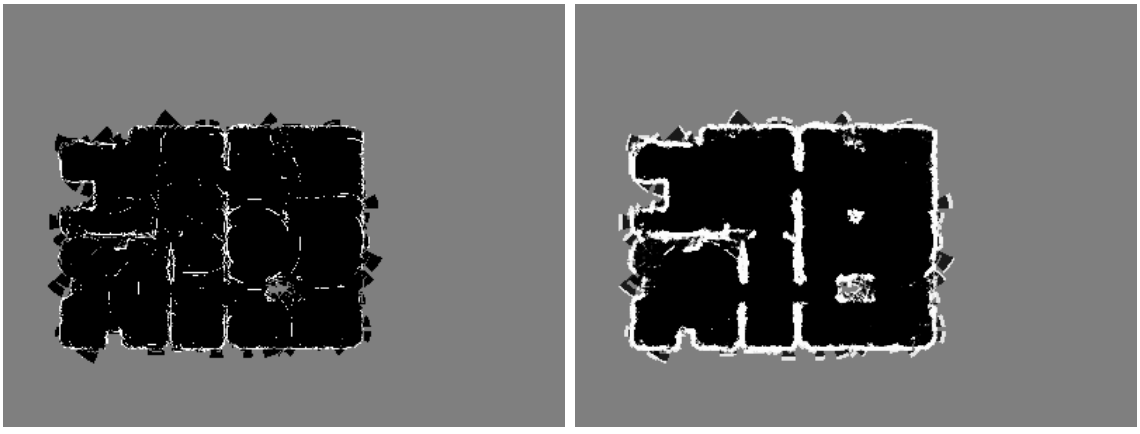

Figure 8: Occupancy gridmap obtained using $P_1(Occ|r)$, ε = *sys.float_info.epsilon*, δ = 0.05 and using ground truth robot position

Figure 9: Occupancy gridmap obtained using $P_1(Occ|r)$, ε = 0.10, δ = 0.15 and using ground truth robot position

Figure 10: Occupancy gridmap obtained using $\mu_s(Ept)$, $\otimes = product$, ε = 0.10, δ = 0.02 and using ground truth robot position

Figure 11: Occupancy gridmap obtained using $\mu_s(Occ)$, $\otimes = product$, ε = 0.10, δ = 0.02 and using ground truth robot position

Figure 12: Occupancy gridmap obtained using $\mu_s(Ept)$, $\otimes = product$, ε = 0.10, δ = 0.05, max range entry excluded and using ground truth robot position

Figure 13: Occupancy gridmap obtained using $\mu_s(Occ)$, $\otimes = product$, ε = 0.70, δ = 0.05, max range entry excluded and using ground truth robot position

3.4 Conclusions

Considering the probability method, increasing ε have a positive effect in reducing the noise in the empty space. This type of false reading are generated by the

model when the sonar sensors return as value the maximum range, inaccurately increasing the probability of an object in the relative cells. By increasing ε the model gives less weight to these false readings. Increasing δ allows to increase the area of the cone occupied but it decrease the precision of the object detection in the gridmap. Figure 8 and 9 can help with appreciating those differences with respect to figure 2 On the other hand, considering the fuzzy set method, the tuning of ε obtain a different result. By using the product as fusion method and setting a high value of ε, allows the support of Occ to decrise gradually, mitigating the negative effect of the false readings. This is not possible using the $min()$ functions as fusion mechanism as $1.0 - \varepsilon$ becomes the minimum support value each cell can achieve. The figure 11 showing the support for the occupied hypothesis, have less defined inner walls compared to figure 7 for this particular reason. As a secondary effect, the support for the empty hypothesis in the figure 10 is comprehensively lower, aggravating the effect of the noise in the empty areas. A different approach to reduce the impact of false readings that does not involve the tuning of the parameter is to exclude the readings that are equal to the maximum sensor range from modifying the "Occupied" hypothesis. The results in figure 12 and 13 shows a combination of the considerations discussed above. It uses a higher ε to compute $\mu_s(Occ)$ to reduce the effect of the false reading near the walls and it excludes the occupied hypothesis when the range of the sensor is equal to the maximum, reducing the noise in the empty areas.

References

[1] Alessando Saffiotti. *BaseCode-2024*. 2024. URL: https://github.com/asaffio/BaseCode-2024.