ROB 550 BotLab Report

Saptadeep Debnath - saptadeb@umich.edu

I. Mapping and SLAM

1.1 - Mapping - Occupancy Grid

The mapping function is implemented in two stages - first we score the cells which are located at the endpoint of the rays. If the cells corresponding to the endpoint of the rays are in the given map, log odds for the cell are increased by '3', defining the boundaries. Next, the cells between the robot cell and the endpoint cell are scored. Bresenham's line algorithm is then used to rasterize the given ray. The log odds of these free cells are then decreased by '1'.

The maps generated when the robot is stationary as is in the case of convex_grid_10mx10m_5cm.log are not quite accurate at far away points. As the robot is stationary the laser rays originating from that position diverge farther away from each other at long distances, because of which some of the cells in between those rays don't get scored. This problem is mitigated when the robot is in motion; wherein because of the motion of the robot the laser rays tend to cover most of the grid cells and hence we obtain a better map in the end.

1.2.1 - MCL - Action Model

Out of the two most widely used motion models, odometry motion model is used over the velocity motion model for this project. Odometry information is obtained by integrating the wheel encoder information over periodic time intervals. It is however observed that even though the odometry action model isn't perfect, it provides a better approximation of the robot motion than the velocity model for predicting the robot motion.

In order to predict the robot pose the odometry information (i.e. x, y and θ) is changed to an RTR information given by equation 1, i.e rotation-translate-rotation which is sampled with a Gaussian noise distribution in the form described in equation 2. In the initial phases of the project the values of the noise constants were chosen as $k_1 = 0.8$ and $k_2 = 0.3$. On further inspection and to reduce the computation load to calculate the standard deviation for each odometry information, the standard deviations are given as constants, where $\sigma^2_{rot1} = 0.05$, $\sigma^2_{trans} = 0.005$ and $\sigma^2_{rot2} = 0.05$. These noise constants are chosen and tuned in order to achieve a better spread of distribution of the particles and to achieve a more accurate proposal which is used for the sensor model. It is expected to see the tuning parameters differ from one mbot to another, but the difference should be small as the make and model of the mbot is consistent The elements which could factor into a possible drastic difference between the tuning parameters from one mbot to another could be the systematic errors related to the specific mbot.

$$\begin{aligned}\delta_{rot1} &= atan2(y_t - y_{t-1}, x_t - x_{t-1}) - \theta_{t-1} \\ \delta_{trans} &= \sqrt{(x_t - x_{t-1})^2 - (y_t - y_{t-1})^2} \\ \delta_{rot2} &= \theta_t - \theta_{t-1} - \delta_{rot1}\end{aligned} \tag{1}$$

$$\begin{aligned}\hat{\delta}_{rot1} &\sim \mathcal{N}(\delta_{rot1},\, k_1 \cdot \mid \delta_{rot1} \mid) \\ \hat{\delta}_{trans} &\sim \mathcal{N}(\delta_{trans},\, k_2 \cdot \mid \delta_{trans} \mid) \\ \hat{\delta}_{rot2} &\sim \mathcal{N}(\delta_{rot2},\, k_1 \cdot \mid \delta_{rot2} \mid)\end{aligned} \tag{2}$$

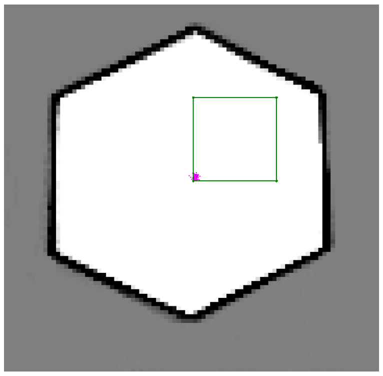
Fig. 1: Particle distribution obtained at the end of drive square with action only

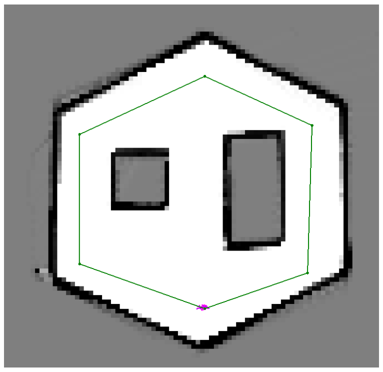
Fig. 2: Particle distribution obtained at the end of obstacle slam with action only

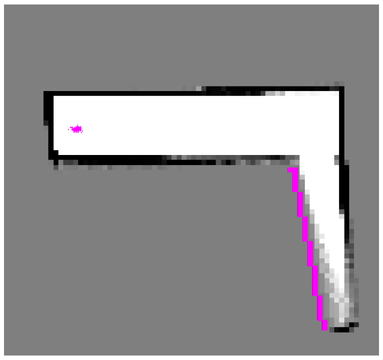

Fig. 3: Particle distribution obtained at the end of straight line calm with action only

1.2.2 - MCL - Sensor Model & Particle Filter

A simplified likelihood field model is implemented which overcomes the lack of smoothness and computational limitations of the sensor beam model. The sensor model function provides a score for a given ray, by inspecting the endpoints of the ray. If the endpoints of the ray are occupied, i.e. the endpoint of the ray is an obstacle, the log odds of that cell is returned. But in case the endpoint is not an obstacle, the ray is extrapolated in the forward and backward direction by one cell and a fraction (0.5) of their log odds is added to the score. These ray scores are then used as weights for a given particle. Since, the sensor model acts as a correction step, the pose estimate is almost similar to the true pose. It is observed that the sensor model performs well and the particles tend to remain in a tight region as the robot moves. The particles though have a tendency to spread if the robot rotates aggressively, but the come closer as the robot starts to move again.

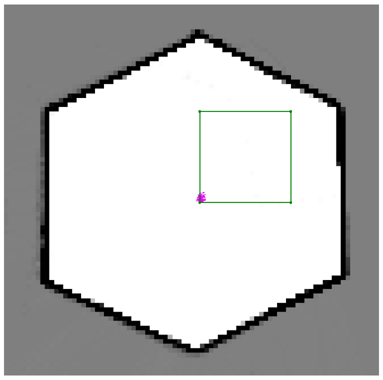

Fig. 4: Particle distribution obtained at the end of drive square with localization only

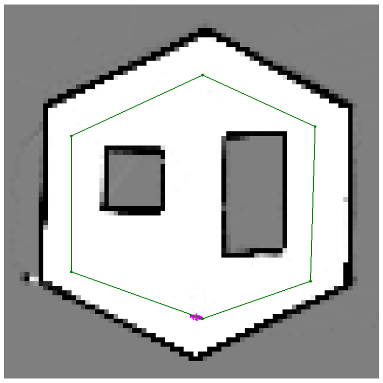

Fig. 5: Particle distribution obtained at the end of obstacle slam with localization only

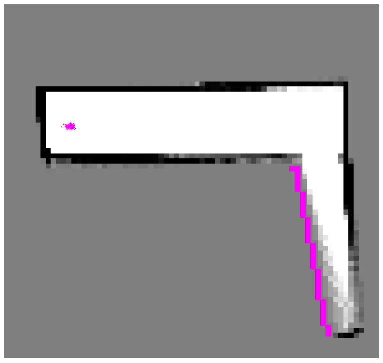

Fig. 6: Particle distribution obtained at the end of straight line with localization only

1.3 - Simultaneous Localization and Mapping (SLAM)

Write a paragraph or two description including the following:

- Did you obtain similar/close performance for obstacle_slam_10mx10m_5cm.log compared to the previous two runs? Why or why not?
- Did the change of map resolution affect your computation time? Did it improve/harm your slam results?
- Did you have to change your mapping log odds to improve on performance? If so, report them.

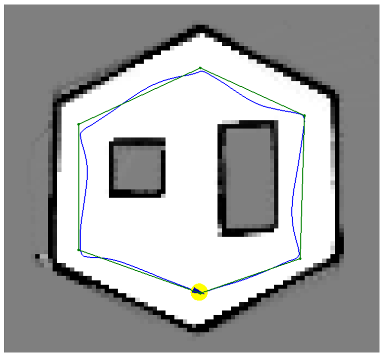

Fig. 7: Particle distribution obtained at the end of Obstacle Slam with Full SLAM

Fig. 8: Particle distribution obtained at the end of Maze LowRes with Full SLAM

Fig. 9: Particle distribution obtained at the end of Maze HiRes with Full SLAM

Fig. 10: Particle distribution obtained at the end of Maze Hard with Full SLAM

II. PATH PLANNING AND EXPLORATION

2.1 - Obstacle Distances

The obstacle distances grid code passes all the three tests in obstacle_distance_grid_test, namely test for unknown distances, test for obstacle distances and test for free space distances. The code is quite optimal and there isn't anything major which tends to slow it down. The code can be made more efficient by expanding a given node to all of it 8 neighboring cells, rather than doing it just for 4 cells.

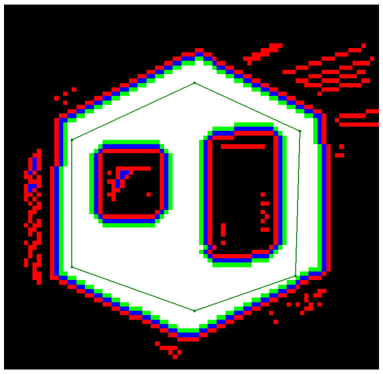

Fig. 11: Obstacle distance grid obtained at the end of Obstacle Slam with Full SLAM

2.2 - A Path Planning*

Report statistics on your path planning execution times for each of the example problems in the data/astar folder. If your algorithm is optimal and fast, great. If not please discuss possible reasons and strategies for improvement.

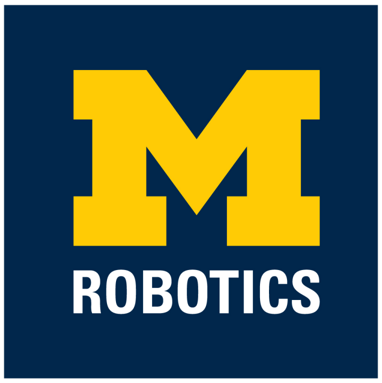

Fig. 12: Path obtained at the end of Maze HiRes with A* path planning

2.3 - Map Exploration

Comment on your exploration performance:

- What are the factors that are preventing your exploration from being ideal?
- Provide your logic behind tackling frontiers (how do you go about chosing your next frontier to go to)
- Comments on the performance of Astar and path planning with automated exploration commands.