Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

Study Project on Lecture
"Probabilistic Reasoning"

Occupancy Grid Map and Particle Filter Localization in ROS2

Hamsa Datta Perur (hperur2s)
Kishan Ravindra Sawant (ksawan2s)

Supervised by

Deebul Nair, M.Sc.

November 2022

1 Problem Overview

In this project occupancy grid mapping and SLAM (Simultaneous Localization and Mapping) will be implemented in ROS2 simulation [1]. There exists many assumptions in the current open source implementations of particle filter including the ROS navigation tools. The main goal of this project is to parameterise these assumptions and build a generalised software for robot navigation and to correlate terminologies from Probabilistic Reasoning course with respect to mobile navigation methods.

2 Occupancy Grid Mapping

2.1 Overview

PENDING

2.2 Open Problems

Some of the possible modifications to the occupancy grid mapping that will be considered are:

1. In occupancy grid mapping occupancy status of individual cells are considered to be independent from that of neighboring cells, which is not an accurate assumption while dealing with uncertain perception data. Considering neighborhood of a cell in the measurement model and parameterising the mapping task by the size of the neighborhood.

2. Some works consider odometry data as certain knowledge. By using Rao-Blackwellized Particle Filter, posterior over map as well as position can be predicted [2].

3. Allowing different types of noises to model sensor and odometry data.

4. Based on confidence of localisation, updating the map. This might help to handle dynamic obstacles.

2.3 Code

The following implementation of occupancy grid mapping is derived from grid-mapping-in-ROS repository.

2.3.1 Structure

```
probabilistic_reasoning_hbrs
├── src
│   └── occupancy_grid_mapping
│       └── occupancy_grid_mapping
│           ├── gmapping_node.py
│           └── submodules
│               ├── bresenham.py
│               ├── grid_map.py
│               ├── utils.py
│               └── message_handler.py
└── README.md
```

2.3.2 Description of Code

1. The probability values for cells being occupied, free or unseen (assigned with prior probability) are set. The size, resolution and name of the grid map are set.

```
P_prior = 0.5  # Prior occupancy probability
P_occ = 0.9    # Probability that cell is occupied
P_free = 0.3   # Probability that cell is free
RESOLUTION = 0.03   # Grid resolution in [m]
MAP_NAME = 'world'  # map name without extension

map_x_lim = [-10, 10]
map_y_lim = [-10, 10]
```

2. 'GMappingClass()' is used to continuously access polar positions of individual scan data and the pose of the robot with respect to its odom frame of reference.

```
distances, angles, _ = node.distances, node.angles
x_odom, y_odom, theta_odom = node.x_odom, node.y_odom,
                             node.theta_odom
```

3. 'descritize' method from the 'GridMap' class (imported from submodules/-grid_map.py) is used to assign the index to the odometry data with respect to the initialised size of the grid. It should be noted that the size of the map should be large enough to accommodate all possible odometry values.

```
x1, y1 = gridMap.discretize(x_odom, y_odom)
```

```
def discretize(self, x_cont, y_cont):
    """
    Discretize continious x and y
    """
    x = int((x_cont - self.X_lim[0]) / self.resolution)
    y = int((y_cont - self.Y_lim[0]) / self.resolution)
    return (x,y)
```

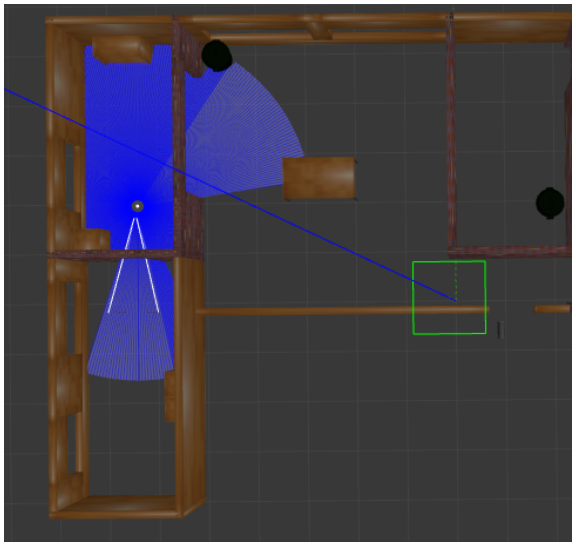
It can be observed that if the robot is at [map_x_lim[0], map_y_lim[0]], then it is assigned with the indices of [0,0].

4. $X2$ and $Y2$ variables are used to store all points measured from the laser scanner, which is represented by green colour in the grid map while visualising the map. It also consists of the points which have reached the maximum range of the laser scanner.

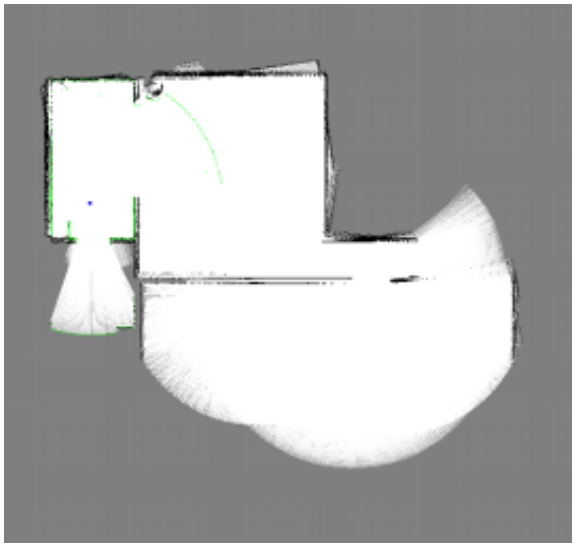
   In Figure 1(b), the green color scan data represents the cells being hit at current instance, the white cells are the detected free cells, the black cells are the occupied cells and the grey colored cells are yet to be discovered and are assigned with prior probability.

   'bresenham' method form 'submodules/bresenham.py' is used to determine all the cells which are free and this information is updated in the grid map. All the cells which are within the maximum range of the laser scanner are considered occupied and are updated in the grid map.

```
# for BGR image of the grid map
X2 = []
```



(a) Environment for mapping



(b) Grid map visualisation

Figure 1: Visualisation of the process of mapping

```
Y2 = []

for (dist_x, dist_y, dist) in zip(distances_x, distances_y,
    distances):
    # x2 and y2 for Bresenham's algorithm
    x2, y2 = gridMap.discretize(dist_x, dist_y)
```

```
8
9       # draw a discrete line of free pixels, [robot position ->
        laser hit spot)
10      for (x_bres, y_bres) in bresenham(gridMap, x1, y1, x2, y2
        ):
11          gridMap.update(x=x_bres, y=y_bres, p=P_free)
12
13      # mark laser hit spot as ocuppied (if exists)
14      if dist < node.range_max:
15          gridMap.update(x=x2, y=y2, p=P_occ)
16
17      # for BGR image of the grid map
18      X2.append(x2)
19      Y2.append(y2)
```

5. update method from 'grid_map.py' is used to update the occupancy status of a cell based on inverse sensor model $p(x|z_t)$, where x is the state of the cell being occupied and z_t is the measurement from laser scan data at time t.

   From Bayes' rule, we know that,

$$p(x|z_{1:t}) = \frac{p(x|z_t)p(z_t)}{p(x)} \frac{p(x|z_{1:t-1})}{p(z_t|z_{1:t-1})} \tag{1}$$

   Similarly, the probability of a cell being free given the history of laserscan data will be,

$$p(\bar{x}|z_{1:t}) = \frac{p(\bar{x}|z_t)p(z_t)}{p(\bar{x})} \frac{p(\bar{x}|z_{1:t-1})}{p(z_t|z_{1:t-1})} \tag{2}$$

   The log of ratio of $p(x|z_{1:t})$ and $p(\bar{x}|z_{1:t})$ can be written as a recurcive formulation as,

$$l_t(x) = log\frac{p(x|z_t)}{p(\bar{x}|z_t)} + log\frac{p(\bar{x})}{p(x)} + l_{t-1}(x) \tag{3}$$

```
1 def update(self, x, y, p):
2     """
3     Update x and y coordinates in discretized grid map
4     """
```

```
    # update probability matrix using inverse sensor model
    self.l[x][y] += log_odds(p)
```

'log_odds(p)' method from 'grid_map.py' is used to update the occupancy probability of every cell. Here $(1 - p)$ represents the probability of cell being free.

```
def log_odds(p):
    """
    Log odds ratio of p(x)
    """
    return np.log(p / (1 - p))
```

6. In the final step, 'calc_MLE()' method is used to discretize the probability value assigned to each cell as occupied, free or unknown.

```
def calc_MLE(self):
    """
    Calculate Maximum Likelihood estimate of the map
    """
    for x in range(self.l.shape[0]):
        for y in range(self.l.shape[1]):
            # if cell is free
            if self.l[x][y] < log_odds(TRESHOLD_P_FREE):
                self.l[x][y] = log_odds(0.01)
            # if cell is occupied
            elif self.l[x][y] > log_odds(TRESHOLD_P_OCC):
                self.l[x][y] = log_odds(0.99)
            # if cell state uncertain
            else:
                self.l[x][y] = log_odds(0.5)
```

3 Particle Filter Localization

3.1 Overview

PENDING

3.2 Open Problems

Some of the possible modifications/implementations(based on [3] [4]) to the particle filter that will be considered are:

1. Initialization of particle cloud: create a particle cloud of n randomly positioned particles upon startup in the mapped surroundings.
2. Movement model: Update each particle's posture in accordance with the robot's movements.
3. Measurement model: Based on how near the robot sensor readings match what they should be, assign a weight to each particle.
4. Re-sampling: Re-sample all the particles based on the probabilities we obtain after normalizing the particles by weight.
5. Updating estimated robot pose: According to the average pose of all the particles, update the estimated robot pose.

4 Evaluation

- Quality of the occupancy grid map and localisation will be qualitatively compared with the corresponding outcomes of ROS navigation tools.

5 Work Plan

- 23/11/2022: Implementing occupancy grid mapping considering cluster of cells.
- 07/12/2022: Initial implementation of Rao-Blackwellized Particle Filter.

- 28/12/2022: Testing adaptation to change in position of minor dynamic obstacles.
- 18/01/2023: Final presentation of project.

References

[1] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[2] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proceedings of the 2005 IEEE international conference on robotics and automation*, pp. 2432–2437, IEEE, 2005.

[3] "Particle filter localization project." `https://classes.cs.uchicago.edu/archive/2021/spring/20600-1/particle_filter_project.html`. Accessed: 2022-11-09.

[4] "Particle filter localization github." `https://github.com/carlosazpurua1004/particle_filter_localization_project`. Accessed: 2022-11-09.