Occupancy grid Mapping based on Rao-Blackwellised particle filters

Zhengming Zhu zhzhu@kth.se KTH Royal Institute of Technology

January 17, 2024

Abstract

This is an academic report for the final project of the course *EL2320 Applied Estimation* at KTH, Stockholm. Simultaneous Localization and Mapping has been the solution for mobile robots for a long time. There are many different types of SLAM technique is used like Kalman filters based SLAM, Graph based SLAM, particle filters based SLAM, etc. Recently, Rao-Blackwellized Particle Filters(RBPFs) have been intriduced as an effective means to solve this puzzle. Additionally, Gazebo is prominent open-source robotics simulator widely used in the robotics community, particularly in conjunction with the Robot Operating System(ROS). This is a high-quality simulation environment to explore special properties of different estimators.

1 Introduction

SLAM plays the key role in the application of mobile robot or autonomous vehicle, the goal of SLAM is to use the environment to update the real-time position of the robot and the accurate map. Since the odometry of the robot often unreliable and has large uncertainty, we cannot depend directly on the data of odometry. So, we need use laser scan which has smaller uncertainty to help refine the solution. In the EKF based SLAM, its update process is rely on some features. FastSLAM[1] proposed the use of the EKF for incrementally estimating the posterior distribution, which really decreases the needed features when updating.

In recent year, Doucet and Murphy introduced Rao-Blackwellized particle filters[2] as an accurate and effective method to solve SLAM problem. In the approach, each particle represent an estimation of the map and the pose of the robot, RBPFs divides the SLAM problem into two part, the map is more accurate since the current observation is incorporated into the estimated map after taking consideration of the estimation of pose. For a method based on particle filter, it cannot avoid the major challenges like the particle depletion, resampling strategy, etc. The Gmapping[3] has presented two approaches to improve the performance of RBPFs applied to the occupancy grid mapping problem. One is that the proposal distribution could be narrowed by the accurate sensors, the another one is applying an adaptive resampling technique to reducing the risk of particle depletion. However, these solution rely on available, accurate odometry and spend much computing resources.

In the project, I applied the RBPFs based SLAM with ROS platform, Gazebo can provide the environment and robot with needed sensors. After collection data from virtual environment, the RBPFs based SLAM will solve the grid map which can be displayed on the rviz. In that case, I can research the properties of algorithms with low cost. In this work, what I have done as follows:

- Configured the simulation with indoor environment and equipped robot in Gazebo, and controled the robot to move around the environment to record own data.
- Applied RBPFs based SLAM algorithm to solve problem and discussed the properties of estimators like the number of particles, uncertainty of motion model and measurement model, etc.
- Explored the influence of odometry, the RBPFs estimator solves with odometry or without odometry, discussed about the reason

and compared with the improved algorithm which does not depend on odometry.

This paper will be organized as follows. After showing the related works, I first illustrate the SLAM problem solved by RBPFs. Then I will provide algorithm details referenced in [3]. Finally, I did the experiments under the simulation environment and discuss the limitation as well as improvement algorithm.

2 Related work

In the realm of mobile robots and autonomous vehicle mapping and localization, Rao-Blackwellised Particle Filters play a pivotal role in enhancing the precision and efficiency of probabilistic estimations of trajectories and maps. Doucet and Murphy[2] shows that RBPFs lead to more accurate estimates than standard PFs.The core concept involves decomposing SLAM estimation into two parts. Each particle represents a hypothesis of the robot's trajectory, with the mapping being estimated as a conditional probability rely on these trajectory hypotheses. The FastSLAM[1] proposed a RBPFs based algorithm and combined the EKF. It efficiently represents map by occupancy grids. However, FastSLAM is not performance well because of particle depletion. Giorgio and Cyrill[3] proposed two approaches to improve the performance of RBPFs, the first one is using a modified proposal distribution which considers the accuracy of the robot's laser scan, because the sensor often has lower covariance than odometry or IMU. For another approach, It applies an adaptive resampling technique to keep the diversity of particle while reducing the risk of particle depletion. However, Gmapping rely on accurate odometry because the priori estimation of robot's trajectory has strong connection with the data from odometry. In recently, there are also many algorithms overcome this limitation like HectorSLAM[4] which is based on a roboust scan matching approach without odometry. HectorSLAM is not a RBPFs based algorithm, but it can create real-time high-quality 2D maps using high-frequency laser data. It shows remarkable performance in dynamic environments.

3 Method

3.1 SLAM problem definition

According to Thrun.[5], we can define the online SLAM problem as the estimation of the posterior over the momentary pose along with the map:

$$p(x_t, m|z_{1:t}, u_{1:t})$$

Here x_t is the pose at time t, m is the map, and $z_{1:t}$ and $u_{1:t}$ are the measurements and controls. Online SLAM problem only involves the estimation of variables that persist at time t. For the second SLAM problem that called full SLAM problem:

$$p(x_{1:t}, m|z_{1:t}, u_{1:t})$$

Where the $x_{1:t} = x_1, x_2, \cdots, x_t$ is the trajectory of the robot.

3.2 SLAM with RBPFs

As shown above, the full SLAM problem is a joint posterior, so we can rewrite it as:

$$\begin{aligned} &p(x_{1:t}, m|u_{1:t}, z_{1:t}) \\ &= p(x_{1:t}|u_{1:t}, z_{1:t})p(m|x_{1:t}, u_{1:t}, z_{1:t}) \\ &= p(x_{1:t}|u_{1:t}, z_{1:t})p(m|x_{1:t}, z_{1:t}) \end{aligned} \tag{1}$$

In the Eq.(1), the $p(m|x_{1:t}, u_{1:t}, z_{1:t})$ does not depend on $u_{1:t}$. In this way, we divide the joint posterior into two part, the first part is estimation of the posterior of the trajectory and the second part is the estimation of map based on the known trajectory and observation. The estimation of the follow the Bayes rule:

$$\begin{aligned} &p(x_{1:t}|u_{1:t}, z_{1:t}) \\ &= p(x_{1:t}|z_t, u_{1:t}, z_{1:t-1}) \\ &= \eta p(z_t|x_{1:t}, u_{1:t}, z_{1:t-1})p(x_{1:t}|u_{1:t}, z_{1:t-1}) \\ &= \eta p(z_t|x_t)p(x_{1:t|u_{1:t}, z_{1:t-1}}) \\ &= \eta p(z_t|x_t)p(x_t|x_{1:t-1}, u_{1:t}, z_{1:t-1}) \\ &p(x_{1:t-1}|u_{1:t}, z_{1:t-1}) \\ &= \eta p(z_t|x_t)p(x_t|x_{t-1}, u_t)p(x_{1:t-1}|u_{1:t-1}, z_{1:t-1}) \end{aligned} \tag{2}$$

As shown in Eq.(2), $p(z_t|x_t)$ is the observation, $p(x_t|x_{t-1}, u_t)$ is the predicted robot's pose with control input, $p(x_{1:t-1}|u_{1:u-1}, z_{1:t-1})$ is the posterior probability of last time. To estimate the posterior $p(x_{1:t}|z_{1:t}, u_{1:t})$, one can use particle filter. In

this way, each particle is one hypothesis of robot's trajectory. If the trajectory and observation are known, the estimation of map will be much easier to solve. For particle filter, the most importance step is resample. To make use of as much as possible valid particles, the next generation of particles X^t is sampled from the proposal distribution π. In general, the prior distribution is obtained from motion propagation, which relies on the data from odometry. The importance weights can be represent as:

$$\text{Weight} = \frac{\text{Target distribution}}{\text{Proposal distribution}}$$
$$\omega_t^{(i)} = \frac{p(x_{1:t}^{(i)}|z_{1:t}, u_{1:t})}{\pi(x_{1:t}^{(i)})|z_{1:t}, u_{1:t}} \tag{3}$$

Particle resampling has a direct relationship with importance weight. Resampling is achieved with substitution, sometimes, these particles with large weight may be copied many times, whereas these particles with small weight will be discarded. This phenomenon will reduce the diversity of particles and leads to poor exploration, which is called particle depletion. The quality of useful particles decides the performance of estimation.

3.3 Approaches of improved resampling

This part is referenced from Cyrill and Giorgio[3], they proposed two methods for the efficient particle filter implementations. I will discuss the theory part of these two improvement.

Improved proposal distribution: As mentioned in the process of resampling, the particle filter performance well when the proposal distribution approximates the target distribution. In most of time, particle filters use the odometry motion model as the proposal distribution for the reason that this way is simple and straightforward. However, the covariance of odometry data is much larger than sensors. The approach takes the latest scan data into consideration, and limit the proposal distribution in a narrow area, in that case, algorithm can use less particles to approximate the distribution of robot's pose, one can inference it by Bayes rule as follows:

$$p(x_t|x_{t-1}, u_t) \rightarrow p(x_t|x_{t-1}, u_t, z_t, m)$$

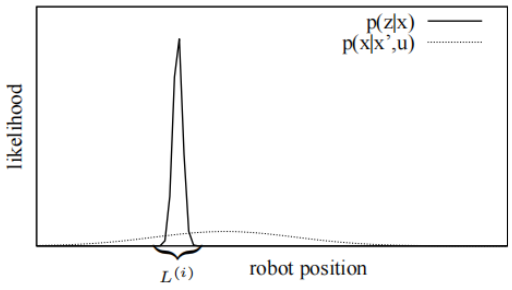

Figure 1: Shows the covariance of laser scan's scan matching likelihood is much more accurate than odometry's.[3]

$$\begin{aligned} & p(x_t|x_{t-1}, u_t, z_t, m) \\ & = \eta p(z_t|x_t, u_t, x_{t-1}, m) p(x_t|x_{t-1}, u_t, m) \\ & = \eta p(z_t|x_t, m) p(x_t|x_{t-1}, u_t) \\ & \approx \eta p(z_t|x_t, m), \quad x_t \in L^{(i)} \end{aligned} \tag{4}$$

In the Eq.4, one can assume $p(z_t|x_t, m)$ $p(x_t|x_{t-1}, u_t)$ is approximated by $\eta p(z_t|x_t, m)$ as lone as in the area $L^{(i)}$, for the reason that the distribution from odometry is wide-range and planar whereas the distrition from laser scan is peaky. So, this approach assume the motion model $p(x_t|x_{t-1}, u_t)$ is a constant and integrate it into normalization parameter η.

To efficient sample next generation of particles, we can assume these distribution are Gaussian, so the result can be computed as a Multivariate Gaussian Distribution, this method first use a scan-matcher to find the peaky area of the observation likelihood function, we are interested in this area and evaluate the sampled points based on the target distribution. For every particle i, the mean $\mu_t^{(i)}$ and the covariance $\Sigma_t^{(i)}$ of this Multivariate Gaussian Distribution is determined by the K sampled points in the area $L^{(i)}$:

$$\mu^{(i)} = \frac{1}{\eta(i)} \sum_{j=1}^{K} x_j p(z_t|x_j, m_{t-1}^{(i)}) \tag{5a}$$

$$\Sigma_t^{(i)} = \frac{1}{\eta(i)} \sum_{j=1}^{K} p(z_t|x_j, m_{t-1}^{(i)}) \tag{5b}$$
$$(x_j - \mu_t^{(i)})(x_j - \mu_t^{(i)})^T$$

In fact, in the original part of Gmapping, it also takes into account the odometry information when computing the mean and covariance. As a result, the posterior distribution of every particle can be represented as follows:

$$p(x_t^{(i)}) = det(2\pi\Sigma_t^{(i)})^{-\frac{1}{2}} \exp\{-\frac{1}{2}(x_t^{(i)} - \mu_t^{(i)})^T \Sigma^{(i)-1}(x_t^{(i)} - \mu_t^{(i)})\} \quad (6)$$

Adaptive Resampling:As we already has the appropriate importance weight, next step is resampling according to the weight. In the typical particle filter, particles with a low importance weight $\omega^(i)$ are replaced by samples with a high wight, to keep the diversity of particles, it is essential to apply a criterion to decide whether resampling or not. This approach proposes a formulation to evaluate it:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N}(\hat{\omega}^{(i)})^2} \quad (7)$$

The N_{eff} can be regarded as a measurement of the dispersion of the importance weight, as mentioned by Cyrill and Giorgio, the worse the approximation of the target distribution, the higher is the variance of the importance weights. So, N_{eff} is a useful value to evaluate the necessity of resampling. In the detail of this approach, it set the threshold as $N/2$ where N is the number of particles.

3.4 Implementation

This part I mainly implemented a RBPFs to solve the SLAM problem. As a classical RBPFs based SLAM algorithm, Gmapping is a good reference, so I followed the structure of Gmapping and experimented based on my own collected data in Gazebo. Furthermore, in order to explore the influence of odometry on the performance of SLAM algorithm, I also applied the Hector SLAM which does not rely on odometry in the same environment.
Firstly, the algorithm of fastSLAM gives a good guideline of RBPFs, according to Thrun.[5]: ***Probabilistic robotics***
In the Gmapping, the approach applies scan-matcher because it is useful for grid map application. Usually, we can use "vasco" as a appropriate scan-matcher:

$$\hat{x}_t^{(i)} = \arg max p(x|m_{t-1}^{(i)}, z_t, x_t'^{(i)}$$

Algorithm 1 FastSLAM_occupancy_grid

procedure FastSLAM(χ_{t-1}, u_t, z_t):
  $\overline{\chi}_t = \chi_t = \emptyset$
  while $k \in M$ **do**
    $x_t^{[k]}$ = sample_motion_model($u_t, x_{t-1}^{[k]}$)
    $\omega_t^{[k]}$ = measurement_model_map
    $(z_t, x_t^{[k]}, m_{t-1}^{[k]})$
    $m_t^{[k]}$ = updated_occupancy_grid
    $(z_t, x_t^{[k]}, m_{t-1}^{[k]})$
    $\overline{\chi_t} = \overline{\chi_t} + < x_t^{[k]}, \omega_t^{[k]}, m_t^{[k]} >$
  end while
  while $k \in M$ **do**
    draw i with probability $\propto \omega_t^{[k]}$
    add $< x_t^{[k]}, m_t^{[k]} >$ to χ_t
  end while
  return χ_t
end procedure

The $x'^{(i)}$ is prediction, the scan-matcher tries to find the local maximum of the observation likelihood function. The scan-matcher aligns the previous scan or map information with the current state information from sensors. The core parts of the algorithm also follows the structure of particle filters:

- Initialize the particles all over the state space.
- Get the priori distribution from the motion model.
- Compute the importance weight.
- Resample accoding to the weights of particles
- Update the particle set

To be more specific from Gmapping algorithm that has tow improved approaches, the computation of proposal distribution takes the information of laser into consideration. In the step 2 and 3, the details is that algorithm first applies scan-matcher to find the local maximum area of likelihood function. If it can not find the local maximum, we should use the modified proposal distribution with the help of observation model, and the weights will be: $\omega_t^{(i)} = \omega_{t-1}^{(i)} p(z_t|m_{t-1}^{(i)}, x_t^{(i)})$. Then it extracts K samples which obey Gaussian distribution around this interesting area. In that way, we can compute the parameters of the Multivariate Gaussian distribution including mean $\mu_t^{(i)}$ and covariance $\Sigma_t^{(i)}$ via

Eq.(5) and Eq.(6).
To make use of the adaptive resampling approach, in the process of resampling, we need to compute the dispersion of the weights $N_{eff} = \frac{1}{\sum_{i=1}^{N}(\hat{\omega})^{(i)}}$ to decide whether to resample or not. If the N_{eff} is less than threshold, the algorithm should resample and continue next iteration.

4 Experimental results

4.1 Environment

In this work, I set up the simulation environment based on Gazebo, which is a versatile and dynamic platform well-suited for testing robotics in complex, real-wold scenarios. Gazebo offers realistic rendering of environment and robot equipped with odomtery, IMU, laser scan, etc. Besides, I employed the ROS as the primary platform for the simulation, ROS allows all working nodes to be controlled and communicate with each other via the subscription and publish of ROS topic. I used a indoor environment to experiment full SLAM problem.

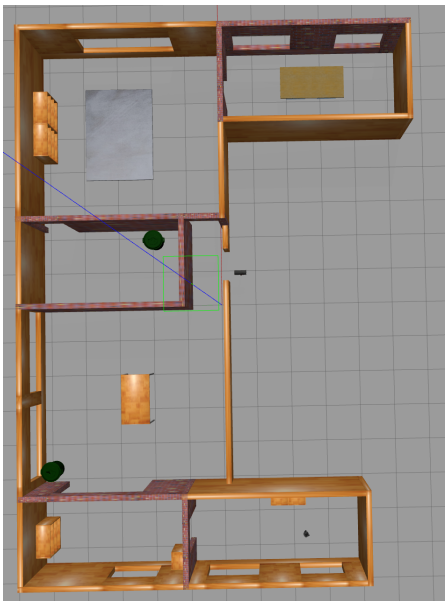

Figure 2: indoor environment in Gazebo

4.2 Data collection

To experiment efficiently, I record my own data by rosbag tool. In this procedure, I input the move control message to topic that is responsible for robot's move, it is usually $/cmd\_vel$. The robot can subscribe this topic via ROS, and traverse the entire unknown area within the Gazebo simulation. Additionally, the entire sequence of events, including the keyboard inputs, robot movements, and sensory feedback, was recorded as a rosbag. The recorded data, encapsulating the robot's interactions with the environment and the corresponding moving trajectory, is beneficial for coming SLAM algorithmic analysis and validation. By replaying this rosbag, I can accurately reconstruct the experimental conditions and ensure the reproducible and robust result. The algorithm need subscribe some topic message including $/scan$, $/odom$, $/tf$, and $/IMU$, etc. $/tf$ is about the transformation of different coordinate.

4.3 Mapping result with different parameters

Different number of particles:
First, I conducted a series of experiments utilizing the algorithm under varying particle numbers to assess the impact on its performance. As the result

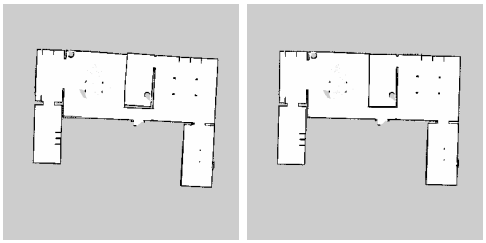

Figure 3: particles=5; particles=20

shows, the number of particles does not impact the mapping procedure much, the more particles will help generate more accurate and clear map. This may be because the simple and small environment, as well as the precise sensor, every particle could has a good hypothesis of the estimation.
Furthermore, I also compare the average entropy under different particle number conditions. The

entropy H can be computed by:

$$H = -\sum_i [p_i \log(p_i) + (1 - p_i) \log(1 - p_i)]$$

where p_i is the probability of that i-th grid has been occupied. In a occupancy grid maps, every grid is represented by free, occupied, and unknown. entropy can be used to measure the degree of uncertainty in a map. The higher entropy means larger uncertainty.

particle number	average entropy
N=5	0.3218
N=10	0.2303
N=20	0.1498
N=100	0.04604

Table 1: average entropy with different particle number

Different measurement noise: This segment of my experiment focuses on the influence of different levels of measurement noise on the performance of the SLAM algorithm, I fixed the particle number as 10, one can represent the uncertainty of measurement as $\sigma = [sigma, lsigma] \in R^2$, here $sigma$ is the standard variance of laser scan, and $lsigma$ is the standard variance of sensor when measuring long distance objects.

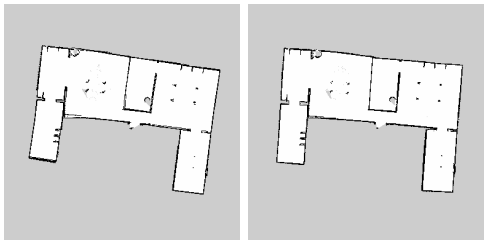

Figure 4: $\sigma = [0.2, 0.3]; \sigma = [0.5, 1.0]$

As shown above in Fig(4) and Fig(5), the increasing of the measurement noise does not effect the performance much. Here are some possible reasons: (1)The simple and small unknown environment makes the scan-matcher works well; (2)The accurate laser scan sensor decrease the influence of the uncertainty noise;

Different motion noise: This phase of my experimental study investigates the effects of different level of motion noise on the performance of the

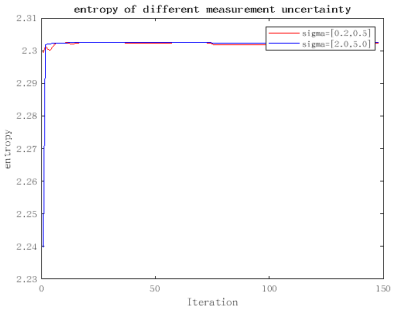

Figure 5: entropy of different measurement uncertainty

mapping algorithm. Also, I fixed the particle number as 10. The motion noise can represent as λ

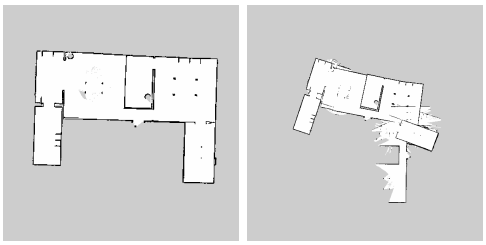

Figure 6: $\lambda = 0.5; \lambda = 2.0$

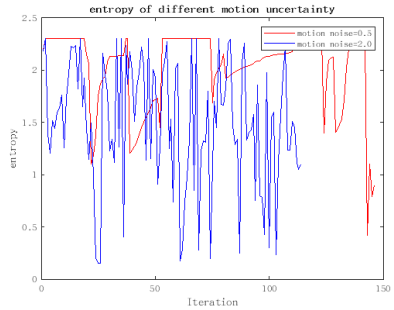

Figure 7: entropy of different motion uncertainty

From the Fig(6), We can conclude that the motion noise has bigger influence than measurement noise on the performance of this algorithm. In the real world, wheels slipping, imprecise controlling lead to the large uncertainty of motion model. It will cause an inaccurate prior estimation from the motion propagation $p(x_t|x_{t-1}, u_t)$. The possible reason is the unreliable odometry information accumulate the error of SLAM, and when the error lager

than a specific value, the solution may fall into a local optimal solution. Besides, during the process of mapping, robot need merge the current observation with the previous estimation, if the localization is not accurate, the generated map is going to overlap and distort.
Furthermore, in the Fig(7), I set the noise of motion as $\lambda = 0.5$ and $\lambda = 2.0$ respectively. Fig(7) shows the change of entropy. The latter curve fluctuate insignificantly during the process of mapping. This phenomena means the certainty that robot perceived about the environment changes over time, which reflects the unstable SLAM system.

4.4 Mapping without odometry

The part made an exploration about the influence of odometry. In the full SLAM problem, odometry data helps the algorithm understand the robot's approximate movement in space including the distance traveled and turning angles. RBPFs based SLAM divides this problem into two sub-tasks, the first is estimate the posterior distribution of the robot's poses, and the estimated map depends on the estimation of robot's poses and observation. Odometry is responsible for update the state of particles namely prior distribution of robot's poses. Therefore, I am interested in how the sloved map will be if there is no odometry, to be more specific, I set odom frame as base footprint to remove odometry:
Gmapping:

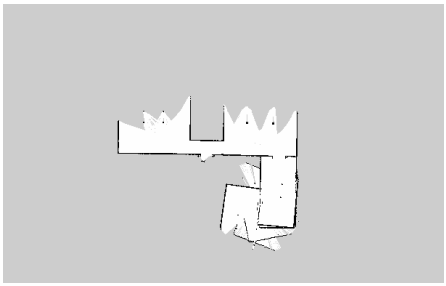

Figure 8: mapping without odometry

Hector SLAM:
From Fig.(7), it suggests that the map distorts much when the robot encounter a corner and need a turning. Without the data from odometry, the original Gmapping algorithm performance poorly.

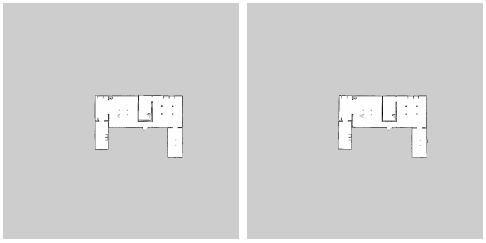

Figure 9: with odometry; without odometry

In order to form a controlled study, I applied a open-source algorithm Hector SLAM to solve this problem in the same simulation environment. Hector SLAM only relies on the Laser scan sensors and applies the scan-matching method to optimize the map. Hector SLAM minimize the error between current estimated map from laser scan sensor and last estimated map, which is a non-linear optimization problem. As we can see from Fig.(9), Hector still mapping the environment well even without odometry data.

4.5 Limitation and Discussion

For the typical SLAM problem, the estimation can be roughly classified as Beyes rule based and optimization based. The most popular approaches are extended information filters, Rao-Blackwellized particle filters and sparse extended information filters. These kinds of method emphasize probabilistic model. In that way, as the same as I have verified, they relies on both the motion model as well as the observation model. Therefore, the RBPFs based algorithm I have tested is not able to mapping the unknown environment well without odometry. Furthermore, RBPFs based algorithm leads to dramatically increase the computation load along with the increasing of map's size. Because every particle has a map to update. typical RBPFs based algorithm is not suitable for embedded system or complex environment.
For another type, the optimization based algorithm is basically solving a non-linear optimization problem, where the objective function could be minimize the error between the prediction and the observation, like Gragh-SLAM and scan-matching based SLAM. Also, I applied the Hector SLAM to solve this environment even with odometry, as a result, it can still generate a map without distorting.

It is worth mentioning, I found that if the robot rotation fast which may cause wrong scan match, the algorithm will fall into local optimal area, therefore, the generated map will also distort like Fig.(10).

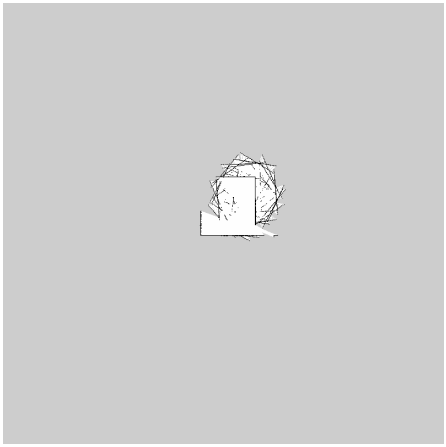

Figure 10: Distorted map because of fast steering

5 Summary and Conclusions

In this work, I mainly study the occupancy grid mapping based on Rao-Blackwellised particle filters. I set up the simulation environment in Gazebo, and controlled and connected all parts of robots including sensor data collecting, moving controlling, SLAM solving, and grid map displaying, etc. This work also applied two improved approaches aimed at the potential problems of RBPFs with referencing Gmapping algorithm.
In the experiment part, I explored the influence of significant parameters including particle number, motion noise and measurement noise on the algorithm's performance. Additionally, I verified this RBPFs based algorithm relies much on the data from odometry and discuss about this phenomenon. For the sake of comparison with algorithm which does not rely on odometry, I also applied the open-source Hector SLAM to solve this problem in the same environment.

References

[1] Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2002). FastSLAM: A factored solution to the simultaneous localization and mapping problem. *Aaai/iaai*, 593598.

[2] Murphy, K., & Russell, S. (2001). Rao-Blackwellised particle filtering for dynamic Bayesian networks. *In Sequential Monte Carlo methods in practice* (pp. 499-515). New York, NY: Springer New York.

[3] Grisetti, G., Stachniss, C., & Burgard, W. (2007). Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*, 23(1), 34-46.

[4] Kohlbrecher, S., Von Stryk, O., Meyer, J., & Klingauf, U. (2011, November). A flexible and scalable SLAM system with full 3D motion estimation. In *2011 IEEE international symposium on safety, security, and rescue robotics* (pp. 155-160). IEEE.

[5] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.