MatLab Simulation of a Rao-Blackwellized Particle Filter with Improved Techniques for Grid Mapping in Mobile robots

Adrian Llopart
P.h.D, DTU

Abstract

Rao-Blackwellized Particle filters have been introduced in simultaneous localization and mapping (SLAM) effectively in the past years. Improved techniques have helped reduce the number of particles needed and allowed a faster and more robust solution. These solutions can be seen, and used, in the C++ ROS *gmapping* package. This paper presents the conversion to a simpler but easy-to-understand MatLab implementation and will describe the RBPF-SLAM principle in a similar manner.

1 Introduction

One of the clearest functionalities of mobile robots is building maps of its surroundings and navigating through them; this is often referred to as SLAM, Simultaneous Localization and Mapping. There exists, however, certain difficulties when implementing SLAM, mainly, for a robot to localize (know its true position), a very precise map has to be built before; but, for a very precise map to be built, a robot must know exactly where it is. Friction, control loss or small obstacles are often the cause of a bad odometry which leads to a poor estimate of the true position. The Rao-Blackwellized Particle filter (as shown in (Doucet et al. 2000) and (Murphy 1999)) solves this issue by generating estimates of the possible position (particles) and giving them a weight. Those particles that have a higher weight, because their estimate matches better the reality, will survive, and the rest will die out in the next generation. To keep a continuous amount of particles and not let all of them die out, resampling stages are carried out where those surviving particles reproduce and keep the particle count constant. It is evident that the more particles used, the more possible descriptions of the reality one has and the higher probability of having at least one particle which is almost perfect. This also induces a higher computational necessity and stops the solution from becoming a real-time application.
The real issue is therefore weighing these particles to know how close they are to the reality. This is known as the *proposal distribution*, and in the latter iterations of the RBPF-SLAM solution, optimal techniques have been developed to obtain a more accurate representation whilst keeping the particle count to a minimum.
In summary, what the RBPF-SLAM algorithm does is given an initial estimate of the true pose, it will use the high precision of a laser scan and the latest map generation and converge the estimate to the true position. This leads to a more accurate map representation and a considerable reduction in errors and number of particles.

2 RBPF joint posterior

As presented by (Murphy 1999), the key idea of RBPF-SLAM is to estimate the joint posterior $p(x_{1:t}, m|z_{1:t}, u_{1:t-1})$. In other words, the algorithm should compute the true (or as close as possible) position (x) of the robot and a map (m) for every iteration, given solely the laser scan data (z) and the odometry measurements (u). This posterior can be factorized as follows:

$$p(x_{1:t}, m|z_{1:t}, u_{1:t-1}) = p(m|x_{1:t}, z_{1:t}) \cdot p(x_{1:t}|z_{1:t}, u_{1:t-1}) \quad (1)$$

This allows us to estimate the position of the robot using the logged odometry and the laser scan data, and then use this position and the laser data again to generate more parts of the map. Thus, the generated map depends heavily on the estimate of the position. This technique is known as Rao-Blackwellization.

2.1 Posterior over maps

The posterior over maps ($p(m|x_{1:t}, z_{1:t})$) can be computed analytically using several methods. This paper used the approach presented by (Thrun, Burgard, and Fox 2006) known as *Occupancy grid mapping* (Algorithm 1) and the *Inverse sensor model* (pages 284-292) to generate a 2-D floor plan occupancy grid map.
Therefore, a map *m* is partitioned into finitely many grid cells $m = m_i$, where each cell contains a value that refers to the probability of it being occupied ($p(m_i)$) with values ranging from '0' (free) to '1' (fully occupied). The posterior over maps can then be approximated as the product of its marginals.

$$p(m|x_{1:t}, z_{1:t}) = \prod_i p(m_i|x_{1:t}, z_{1:t})$$

It is worth noting that this algorithm will use the *log-odds* representation of occupancy to avoid numerical instabilities for probabilities close to zero or one:

$$l_{t,i} = log\frac{p(m_i|z_{1:t}, x_{1:t})}{1 - p(m_i|z_{1:t}, x_{1:t})}$$

and that the probability ration can be easily recovered from the *log-odds* ratio as:

$$p(m_i|z_{1:t}, x_{1:t}) = 1 - \frac{1}{1 + exp(l_{t,i})}$$

Finally, the algorithm makes use of the variables l_0, $l_{occupied}$ and l_{free} which can be adjusted as one pleases. The values applied for this simulation are 0.5, 1 and -1. Also, α makes reference to the average width of walls; and β is the width of the sensor beam. Considering that the simulation uses one cell to represent $10cm^2$, α is set to be equal to 2 (thus 20 cm) and β will be equal to 0.1.

Algorithm 1 Complete Occupancy Grid Mapping

procedure OCCUPANCY_GRID_MAPPING ($l_{t-1,i}$, x_t, z_t)
  for all cells m_i **do**
    if m_i in perceptual field of z_t **then**
      $l_{t,i} = l_{t-1,i} +$ **inverse_range_sensor_model**$(m_i, x_t, z_t) - l_0$
    else
      $l_{t,i} = l_{t-1,i}$
  return $l_{t,i}$

procedure INVERSE_RANGE_SENSOR_MODEL (m_i, x_t, z_t)
  Let x_i, y_ibe the center of mass of m_i
  $r = \sqrt{(x_i - x)^2 + (y_i - y)^2}$
  $\phi = atan2(y_i - y, x_i - x) - \theta$
  $k = argmin_j|\phi - \theta_{j,sens}|$
  if $r > min(z_{max}, z_t^k + \alpha/2) or |\phi - \theta_{j,sens} > \beta/2|$ **then return** l_0
  if $z_t^k > z_{max} and |r - z_t^k| < \alpha/2$ **then return** $l_{occupied}$
  if $r \leq z_t^k$ **then return** l_{free}

2.2 Posterior over potential trajectories

The posterior over potential trajectories $p(x_{1:t}|z_{1:t}, u_{1:t-1}$ will be solved applying a particle filter. This means that one particle will represent one potential trajectory over one time step and will also generate its own map.
Usually, a non-optimal particle filter called SIR (sampling importance resampling) is applied. this filter follows 4 main steps:

- *Sampling*: The next generation of particles $x_t^{(i)}$ is obtained from the previous generation $x_{t-1}^{(i)}$ by sampling form the proposal distribution π. Normally, a probabilistic odometry motion model is used here.
- *Importance weighing*: each particle is assigned with a weight $w_t^{(i)}$ according to the importance sampling principle:

$$w_t^{(i)} = \frac{p(x_{1:t}^{(i)}|z_{1:t}, u_{1:t-1})}{\pi(x_{1:t}^{(i)}|z_{1:t}, u_{1:t-1})} \quad (2)$$

  which incorporates all observations and positions up until that specific point in time.
- *Resampling*: New particles are drawn which replace the old ones depending on their weight. This keeps the particle count constant. the new generation of particles have all the same initial weight.
- *Map estimation*: for every particle, a map is generated according to the approach presented before (Algorithm 1).

One might rapidly notice that this filter requires a trajectory weight evaluation after every observation/time step. hence, this schema becomes highly inefficient over time as trajectories grow. (Doucet, de Freitas, and Gordan 2001) obtained a recursive formulation to compute the importance weights, which allows the calculation of the next weight based on the previous one (and not having to calculate over and over again all weights until that point):

$$w_t^{(i)} = \frac{p(z_t|m_{t-1}^{(i)}, x_t^{(i)})p(x_t^{(i)}|x_{t-1}^{(i)}, u_{t-1})}{\pi(x_t|x_{1:t-1}^{(i)}, z_{1:t}, u_{1:t-1})} \cdot w_{t-1}^{(i)} \quad (3)$$

Generic particle particle filters rely heavily on the recursive structure of equation 3, whilst leaving open what the proposal distribution will be and when the resampling should take place. This paper will focus on the improved techniques presented in (Grisetti, Stachniss, and Burgard 2007) to obtain an optimal proposal distribution based on both odometry and sensor data.

3 RBPF Improved proposal

As described previously, one needs to draw particles from the proposal distribution π to obtain an estimate on the next generation of particles. logically, the more accurate the proposal is, the better the end result of the estimation. typically, particle filters use the odometry proposal distribution because it is easily computed and replacing said odometry distribution in equation 3 yields a very simple weight calculation:

$$w_t^{(i)} = p(z_t|m_{t-1}^{(i)}, x_t^{(i)}) \cdot w_{t-1}^{(i)} \quad (4)$$

However, this approach leads to a suboptimal result. Specifically because the precision of the laser scan data is not taken into account. In other words, if we only sample from the odometry proposal, the importance weights between particles will end up differing significantly because the drawn samples only cover a fraction of the state space region, as shown in Figure 1. If the laser scan data was incorporated, and knowing that it provides a very pinpointed area of high likelihood, the resulting estimation would be significantly more precise. Therefore, integrating the sensor data z_t into the proposal will focus the sampling only in the meaningful

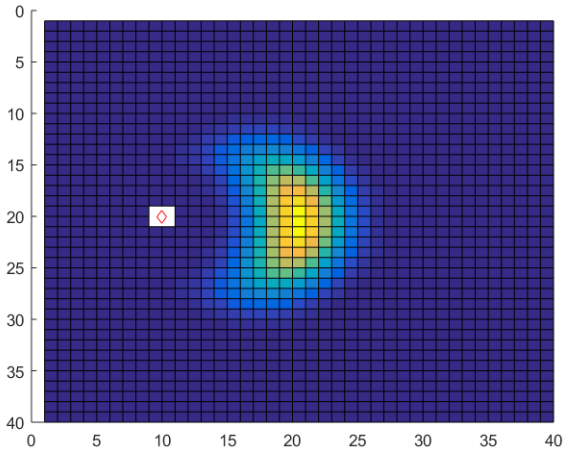

Figure 1: Odometry proposal distribution for a 1 meter command

regions of the observation likelihood. The resulting distribution, according to (Doucet 1998) is as follows:

$$p(x_t|x_{t-1}^{(i)}, m^{(i)}, z_t, u_t) = \frac{p(z_t|x_t, m^{(i)})p(x_t|x_{t-1}^{(i)}, u_t)}{p(z_t|x_{t-1}^{(i)}, m^{(i)}, u_t)} \tag{5}$$

It is common practice to redefine the target distribution in equation 5 as:

$$\tau(x_t) = p(z_t|x_t, m^{(i)})p(x_t|x_{t-1}^{(i)}, u_t)$$

The major problem this approach has with grid maps is that the closed-form approximation of the proposal is not available due to an unpredictable shape of the observation likelihood function.
This could be solve applying an *adaptive* particle filter which samples potential poses x_j from the motion model and then weights them through the observation likelihood to obtain an approximation of the optimal proposal. Since the observation likelihood is typically peaked in very small areas, a dense sampling is necessary to capture correctly the peak, leading to a sub-optimal proposal, much like the motion model, because of its high computational requirements. Thus this paper will not pursue this approach.
Another way of solving this major issue is by taking into consideration that the observation likelihood typically has only one maxima allowing us to sample around this maxima and avoiding other less meaningful regions. Specifically, the posterior $p(x_t|x_{t-1}^{(i)}, m^{(i)}, z_t, u_t)$ is locally approximated around the maxima of the observation likelihood via a *scan matching* procedure.

3.1 Scan matching approach

This approach has a series of steps:
Initially, the next generation of particles is sampled from the odometry distribution. These particles are then approximated to the meaningful area of the observation likelihood via a *Scan matching* method. In this process, the current scan data z_t is compared to the particles map generated up until this point m_i to converge the initial guess of the particles position to one that is closer to the reality. The algorithm used in this paper is known as ICP (Iterative Closest Point) which returns translational and rotational vectors that are applied to x_t to reach a better position estimate. The algorithm takes into account two sets of point clouds and iteratively overlays them by minimizing the squares error.
We now want to sample around this new estimated position. To do so, first a Gaussian has to be determined around it. To calculate the mean $\mu^{(i)}$ and the variance $\Sigma^{(i)}$ we also take into account the odometry information:

$$\mu^{(i)} = \frac{1}{\eta^{(i)}} \cdot \sum_{j=1}^{K} x_j \cdot p(z_t|m_{t-1}^{(i)}, x_j) \cdot p(x_j|x_{t-1}^{(i)}, u_{t-1}) \tag{6}$$

$$\Sigma_t^{(i)} = \frac{1}{\eta^{(i)}} \cdot \sum_{j=1}^{K} p(z_t|m_{t-1}^{(i)}, x_j) \cdot p(x_j|x_{t-1}^{(i)}, u_{t-1}) \cdot (x_j - \mu_t^{(i)})(x_j - \mu_t^{(i)})^T \tag{7}$$

with the normalization factor:

$$\eta^{(i)} = \sum_{j=1}^{K} p(z_t|m_{t-1}^{(i)}, x_j) \cdot p(x_j|x_{t-1}^{(i)}, u_{t-1}) = \sum_{j=1}^{K} \tau(x_t) \tag{8}$$

Now we have a closed-form approximation of the optimal proposal. As shown in (Grisetti, Stachniss, and Burgard 2007), using the new proposal distribution, the new weights are calculated as:

$$w_t^{(i)} = w_{t-1}^{(i)} \cdot \eta^{(i)} = w_{t-1}^{(i)} \cdot \sum_{j=1}^{K} \tau(x_t) \tag{9}$$

The main advantage of the improved proposal is that it takes both the odometry data and laser scans into consideration which reduces notably the proposals densities uncertainty and allows more efficient sampling. The only problem this approach presents is when the observation likelihood is multimodal. Since the scan matching algorithm maximizes the observation likelihood around the closest local maxima to the initial guess, it is possible that this approximation fails because it misses other additional maxima due to the multimodlity of the likelihood. In other words, the reported local maxima is not the global maxima. However, in reality, the distribution tends to be uni-modal, as reported in (Grisetti, Stachniss, and Burgard 2007), and so this issue is of little concern in common practices.
Another problem is when the scan matcher fails to report a scan conversion. This is generally because of poor observations or low overlapping areas between scans. For this case in particular, the proposal will only be determined by the raw motion model. This issue, however, is, in reality, not very common.

4 RBPF Adaptive resampling

The second main difference, this paper presents, with other general particle filters concerns the resampling step. During resampling, particles with low weight are replaced by new ones with higher weight. This is essential because only a finite amount of particles are used to approximate the target distribution; but, if not applied carefully, the resampling can remove good samples and lead to impoverishment of the particle filter. Also, the criterion for deciding when to resample is of utmost importance for a good overall performance.
The approach used in this paper relies in the so-called effective sampling size for target posterior estimation (liul 1996). This value was proposed by (Doucet, de Freitas, and Gordan 2001) to be:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (\bar{w}^{(i)})^2} \tag{10}$$

where $\bar{w}^{(i)}$ is the normalized weight of the particle i.
The key idea behind N_{eff} is that the worse the particle approximation is, the larger variance between weights there would be. If the estimate was perfect, all particles would end up with the same weight value. As presented in (Grisetti, Stachniss, and Burgard 2007), the resampling step should carried out when the value of N_{eff} drops below $N/2$, where N is the total number of particles.

5 The RBPF improved algorithm

The algorithm 2 presents the full schema for the improved RBPF utilized in this paper. The steps are as follows:

- An initial estimate of each particles position $x_t^{(i)}$ is obtained from the previous pose $x_{t-1}^{(i)}$, the odometry measurements u_{t-1} and a realistic approximation of the errors the odometry might propose.
- A scan matching algorithm, in this case an ICP, is executed between the latest laser scan data z_t and a point cloud derived form the latest map generated for that particle. To calculate the point cloud from the map, only that area inside what would be the region of the laser scanner is evaluated. This saves important resources and does not calculate point clouds in regions that cannot be matched with the laser scan data. The rotational and translational vectors derived through the ICP are used to reconstruct a new pose estimate $\dot{x}_t^{(i)}$ from the motion model estimate $x_t^{(i)}$. Finally, if the scan matcher fails at converging both point clouds, the pose estimate will remain as the one obtained from the motion model
- A set of sampling points x_j is selected in an interval around $\dot{x}_t^{(i)}$. The mean and covariance values for creating a Gaussian distribution are calculated as in equations 6 and 7 via a pointwise evaluation of the target proposal. The first half of the target proposal, $p(x_j|x_{t-1}^{(i)}, u_t)$, is obtained by evaluating the sample x_j in the odometry motion model distribution. The second half of the target distribution, $p(z_t|x_j, m^{(i)})$, is obtained evaluating the observation likelihood for that sample. Several approaches can be taken; this paper has used a point to point comparison between the true scan and the estimated scan a robot in the estimated position in the generated map would see. In this stage the weighing factor $\eta^{(i)}$ is also computed.
- A new final pose estimate for particle (i) is drawn from the Gaussian approximation $\mathcal{N}(\mu_t^{(i)}, \Sigma_t^{(i)})$ of the improved proposal.
- Update importance weights as in equation 9
- The particles map m_i is updated with the final estimated position $\dot{x}_t^{(i)}$ and the laser scan data z_t.
- The resampling stage is carried out depending on the value of N_{eff}.

6 Simulations and results

The *Matlab* code to which this paper refers to works with a simulated RBPF-SLAM. The user is able to create its own map (Figure 2), generate a path which the robot should follow, adjust a series of parameters related to the motion model of the robot or related to the particle filter itself, and simulate the results for a RBPF-SLAM (Figure 3).

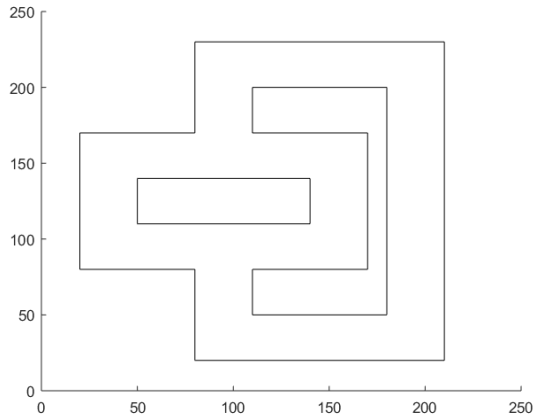

Figure 2: Map created by the user

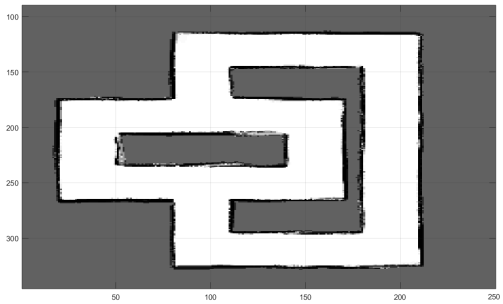

Figure 3: occupancy grid map generated after running RBPF-SLAM

The resulting grid map will have a resolution of $10\frac{cm^2}{cell}$

The errors generated via this schema can be seen in Figure 4. They are minimal except for a particular point in which

Algorithm 2 Improved techniques for RBPF in grid mapping

procedure RBPF-SLAM (x_{t-1}, z_t, u_t, m_{t-1})
  $\mathcal{S}_{t-1}$: sample set of the previous time step
  for $\mathcal{S}^{(i)}_{t-1} \in \mathcal{S}_{t-1}$ **do**
    $< x^{(i)}_{t-1}, w^{(i)}_{t-1}, m^{(i)}_{t-1} >= \mathcal{S}^{(i)}_{t-1}$
    // initial pose estimate
    $x^{(i)}_t = x^{(i)}_{t-1} \oplus u_t$
    // scan matching
    $\dot{x}^{(i)}_t = \text{argmax}_x p(x|m^{(i)}_{t-1}, z_t.x^{(i)}_t)$
    if $dotx^{(i)}_t$ = failure **then**
      $x^{(i)}_t \sim p(x_t|x^{(i)}_{t-1}, u_{t-1})$
      $w^{(i)}_t = w^{(i)}_{t-1} \cdot p(z_t|x_t, m^{(i)})$
    else
      // sample around node
      for $k = 1, ..., K$ **do**
        $x_k \sim \left\{ x_j || x_j - \dot{x}^{(i)}_t | < \Delta \right\}$
      // compute gaussian
      for all $x_j \in \{x_1, ..., x_K\}$ **do**
        $\mu^{(i)}_t = \mu^{(i)}_t + x_j \cdot p(z_t|m^{(i)}_{t-1}, x_j) \cdot p(x_j|x^{(i)}_{t-1}, u_{t-1})$
        $\eta^{(i)} = \eta^{(i)} + p(z_t|m^{(i)}_{t-1}, x_j) \cdot p(x_j|x^{(i)}_{t-1}, u_{t-1})$
      $\mu^{(i)}_t = \mu^{(i)}_t / \eta^{(i)}$
      $\Sigma^{(i)}_t = 0$
      for all $x_j \in \{x_1, ..., x_K\}$ **do**
        $\Sigma^{(i)}_t = \Sigma^{(i)}_t + p(z_t|m^{(i)}_{t-1}, x_j) \cdot p(x_j|x^{(i)}_{t-1}, u_{t-1}) \cdot (x_j - \mu^{(i)}_t)(x_j - \mu^{(i)}_t)^T$
      $\Sigma^{(i)}_t = \Sigma^{(i)}_t / \eta * (i)$
      // sample new pose from gaussian
      $x^{(i)}_t \sim \mathcal{N}(\mu^{(i)}_t, \Sigma^{(i)}_t)$
      // update importance weights
      $w^{(i)}_t = w^{(i)}_{t-1} \cdot \eta^{(i)}$
    // update map
    $m^{(i)}_t = \text{Occupancy\_Grid\_Mapping}(m^{(i)}_{t-1}, x_t, z_t)$
    // update sample set
    $\mathcal{S}_t = \mathcal{S}_t \cup \left\{ < x^{(i)}_{t-1}, w^{(i)}_{t-1}, m^{(i)}_{t-1} > \right\}$
  $N_{eff} = \frac{1}{\sum_{i=1}^{N} (\bar{w}^{(i)})^2}$
  if $N_{eff} < N/2$ **then**
    $\mathcal{S}_t =< textresample\mathcal{S}_t$

theta overshoots. This is definitely due to misalignments in *theta* for the case $0^o = 360^o$.

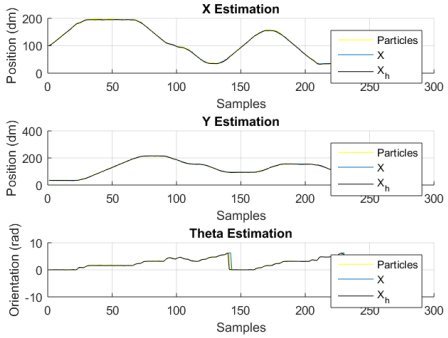

Figure 4: Error between the real and estimated *x*, *y* and *theta* values

References

Doucet, A.; de Freitas, J.; Murphy, K.; and Rusel, S. 2000. Rao-blackwellized particle filtering for dynamic bayesian networks. In *Conf. Uncertainty Artif. Intell. Stanford, CA*, 173–186.

Doucet, A.; de Freitas, N.; and Gordan, N. 2001. *Sequential Monte-Carlo Methods in Practice*. Springer-verlag.

Doucet, A. 1998. On sequential simulation-based methods for bayesian filtering. *Univ. Cambridge, Dept. Eng., Signal process. Group, cambridge, UK*.

Grisetti, G.; Stachniss, C.; and Burgard, W. 2007. Improved techniques for grid mapping with rao-blackwellized particle filters. In *IEEE Trans. Robot.*

liul, J. 1996. Metropolized independant sampling with comparisons to rejection sampling and importance sampling. In *Statist. Comput., vol 6*, 113–119.

Murphy, K. 1999. Bayesian map learning in dynamic environmnets. In *Conf. Neural Inf. Process. Syst. Denver, CO*, 1015–1021.

Thrun, S.; Burgard, W.; and Fox, D. 2006. *Probabilistic Robotics*. The MIT Press.