On Simulation and Analysis of Mobile Robot SLAM using Rao-Blackwellized Particle Filters

Peng Qi and Lu Wang

Abstract—The simultaneous localization and mapping (SLAM) is considered as a crucial prerequisite for purely autonomous mobile robots. In this paper, we demonstrate the mobile robot SLAM using Rao-Blackwellized particle filters (RBPF) through computer simulations under MATLAB platform, while an analytical investigation into the involved algorithms is presented. Then we make further comparisons, not only in parallel between the FastSLAM 1.0 and FastSLAM 2.0, also in vertical between FastSLAM performance and EKF-SLAM performance which used to be the dominant approach to the SLAM problem. Vivid simulations and numerical analysis make the paper illustrated with clarity and perception.

I. INTRODUCTION

In the domain of autonomous mobile robot navigation, simultaneously localizing itself and building up an accurate map without a priori knowledge in an unknown environment is not an easy work, and significantly more difficult than each aspect of localization and mapping. For roboticists, SLAM is a familiar term of address to this simultaneous localization and mapping problem. During the recent decades, researchers have devoted lots of effort to develop the efficient SLAM algorithms. A brief history of developments, state of the art, and essential concepts of SLAM are given in [1], [2], and a useful survey focusing on different filtering strategies can be found in [3]. Specifically, much attention has been focused on three crucial issues: computational complexity, data association, and representation.

Recently estimation theories to deal with the uncertainty in robot perception and action receive more and more recognition. In localization and mapping aspects, it raises methods to figure out robot pose and unknown map by the calculus of probability densities. Here the popular and fundamental tools are (extended) Kalman filter and particle filter under the recursive Bayesian estimation skeleton. The former successfully works on Gaussian or Gaussian mixture distributions, while the latter could represent any distribution, while here FastSLAM 1.0 only uses the system control signal to sample poses [5]. In this paper, simulations of FastSLAM 1.0 and FastSLAM 2.0 are given under MATLAB platform and an analytical investigation into their corresponding performances are proposed, including vertical comparison between EKF-SLAM and FastSLAM and lateral comparison between FastSLAM 1.0 and FastSLAM 2.0. A detailed review of FastSLAM algorithm is beyond the scope of this paper and interested readers are pointed to the related chapters in the textbook [5] for additional information.

The rest of this paper is organized as follows. In Section II we review some main SLAM solutions, including their respective advantages and disadvantages. Section III defines the robot motion model and measurement model. Then, in Section IV, the framework of FastSLAM is presented, while the EKF and PF algorithms are also briefly mentioned here. Section V highlights the simulation and comparative analysis of mentioned solutions in this paper. Finally, we provide some conclusions in Section VI.

II. RELATED WORK

With probabilistic techniques popularity in robotic mapping, SLAM algorithm with the use of the extended Kalman filter (EKF-SLAM) becomes the most common method, which is also considered as arguably earliest SLAM solution. EKF-SLAM algorithm accommodates the nonlinearities from the real world [6], to be more explicit, it builds upon the Gaussian noise assumption and the local linearization of motion and measurement model. However EKF-SLAM will be not effective when the distribution is far different from Gaussian model or the non-linearity is large due to the inherent nature of EKF algorithms. On the other hand EKF-SLAM also suffers from the problem of computational complexity which limits it from handling a large amount of landmarks. A concrete example in [7] reveals the inevitable drawback inconsistency.
An alternative SLAM approach using unscented transformation to approximate the nonlinear functions comes out as UKF-SLAM, which can be more precise than the EKF based approach [8], and Andrade-Cetto et al. [9] carry out a reduced unscented transformation application which means to only use it for computing the robot state and its covariance in SLAM problems. Inverse depth parameterization for SLAM problems permits an efficient and accurate representation of uncertainty during undelayed initialization and beyond, all within the standard EKF based approach [10].

In the application aspects, although SLAM problem has been considered already solved and different improved algorithms are proposed constantly, few publications appear. Actually the scale and structure of many environments are quite limited to almost current SLAM approaches.

III. PROBLEM FORMULATION

A. System dynamic motion model

Regarding SLAM problems, here the interested states being estimated not only contain the coordinates and heading of the robot, but also continuously argument the coordinates of the landmarks that have been detected by the robot so far:

\[
X = \begin{bmatrix} x_t & y_t & \theta_t & x_{m_1} & y_{m_1} & x_{m_2} & y_{m_2} & \ldots & \ldots & x_m & y_m \end{bmatrix}^T
\]

throughout the paper, variables with footnotes \(x, m\) denote variables related to robot and landmark, respectively.

The system dynamical motion model of the robot is needed for the states prediction, as shown below:

\[
X_t,k = f(X_{t,k-1}, U_{k-1})
\]

where \(X_{t,k} \in \mathbb{R}^n\) is robot state at step \(k\) given knowledge of the process prior to step \(k\). \(U\) is the control signal, and it is preferably to be the translational velocity and angular velocity of the robot, i.e. \(U = [v \quad \omega]^T\). And usually the function \(f\) is nonlinear.

In our simulation cases, the control signal \(v\) and \(\omega\) are derived based on the mobile robot odometry information:

\[
v = \frac{\omega_R \cdot r_R + \omega_L \cdot r_L}{2}
\]

\[
\omega = \frac{\omega_R \cdot r_R - \omega_L \cdot r_L}{B}
\]

where \(r_{L, R}\) is the modeled radius of each wheel, and \(B\) is the modeled wheels base length. \(\omega_{L, R}\) can be calculated from the odometry information, such as wheel encoder readings.

B. Measurement model

Below is the measurement model that gives the calculation of the expected measurement based on robot states and landmark states.

\[
Z = h(X_t, X_m)
\]

where \(Z\) is the measurement vector. In our cases, it is composed of bearing measurements \(b\) and range measurements \(r\), which respectively are modeled as:

\[
b = \arctan \frac{y_m - y_t}{x_m - x_t} - \theta_t
\]

\[
r = \sqrt{(y_m - y_t)^2 + (x_m - x_t)^2}
\]

Here the SLAM problem brings out a “chicken or egg” causality dilemma, where we use the map information to localize our robot, while that map is needed to be built based on an accurate pose estimate. The key point of the probabilistic SLAM solution is to view the SLAM problem as a localization problem, and the only difference is that the map information used in update procedure comes from the estimated state vector, instead of a given map dataset.

We also have to notice that commonly the process and measurement noise should be considered in motion model and measurement model, which will influence the estimation processes.

IV. FILTERING ALGORITHMS

Rao-Blackwellized particle filter (RBPF) is an approach to use particle filters for estimating the robot pose, while to estimate the map by using EKFs. In this section, we will firstly describe the general EKF [5], [11] and PF [5] algorithms, and then come up with the FastSLAM solution which combines the both above.

A. General EKF algorithm

EKF complies with a recursive Bayesian estimation process. It receives the previous posteriori state estimate \(X_{t-1}^+\), error covariance estimate \(P_{t-1}^+\) and the control signal and measurements as the inputs, and then outputs current state estimate \(X_t^+\) and error covariance \(P_t^+\) for next loop. In the prediction phase, a priori belief of the state and estimation error is produced by motion model. And measurements are introduced to cooperate with the measurement update step. One loop of the general EKF algorithm is depicted in Algorithm 1 below.

Algorithm 1 EKF \((X_{t-1}^+, P_{t-1}^+, U_{k-1}, Z_k)\)

1. Prediction:
   1. \(X_k^+ = f(X_{k-1}^+, U_{k-1})\)
   2. \(P_k^- = A_k P_{k-1}^+ A_k^T + Q\)

4. Measurement update:
   5. \(K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R)^{-1}\)
   6. \(X_k^+ = X_k^+ + K_k (Z_k - h(X_k^+, X_m))\)
   7. \(P_k^+ = (I - K_k H_k) P_k^-\)
8. return \(X_k^+\), \(P_k^+\)

In the algorithm, - / + represent the priori / posteriori estimate of the corresponding terms. For example, \(X_k^-\) denotes the priori state estimate at step \(k\) given knowledge of the process prior to step \(k\), while \(X_k^+\) refers to the posteriori state estimate after further taking current measurements into account. \(A\) and \(H\) are the Jacobian matrices of the motion model \(f\) and the measurement model \(h\) respectively. And \(Q\) and \(R\) are the process noise covariance and measurement
noise covariance respectively. The Kalman gain $K$ calculated in Line 5 is the real trick of Kalman filter, which balances the weights of odometry information and measurements in the update step. These variables denote the same meanings throughout the paper.

EKF approximation is done by linearizing the nonlinear function at the mean point of the Gaussian, and projecting the original Gaussian distribution through the linearized function, see Fig. 1. This kind of linearization usually is good enough when the nonlinearity is not enormous.

### B. General PF algorithm

Particle filter (PF) is another popular and fundamental estimation tool used in mobile robot localization problems, whose idea is also known as Monte Carlo localization (MCL). Such an algorithm compared with EKF could represent any distribution non-parametrically and deal with nonlinearity flexibly. Its basic concept is to bundle the states by numbers of individual particles with weight. And the tricky part of the PF is the re-sampling part of the algorithm, which fairly selects significant particles while discarding the unimportant ones.

For simplicity, we consider the states to be estimated are the coordinates and orientation of the robot, and they together with a weight will be included in one particle:

$$s = \{x \ y \ \theta \ w\}$$ (8)

Usually it is intuitive to see the particle set $S$ by presenting them at estimated robot positions. And PF also uses two main steps to prompt the particle set to evolve. One loop of the general PF algorithm is depicted in Algorithm 2.

**Algorithm 2** $PF(S_{k-1}, U_{k-1}, Z_k)$

1: **Prediction and measurement update** :
2:   for $i = 1 : M$ do
3:     draw $X_k^i \sim p(X_k^i|U_{k-1}, X_{k-1}^i)$
4:     $w_k^i = p(Z_k|X_k^i)$
5:   end for
6:   normalize $w_k$
7:   $X_k = \Sigma_{i=1}^M w_k^i \cdot X_k^i$
8:   **Resampling** :
9:   for $i = 1 : M$ do
10:    draw $j$ with probability $\propto w_k^j$
11:   add $s^j$ to the new particle set
12:   end for
13:   $\forall s^j \in S_k, w_k^j = 1/M$
14: return $S_k$

The first step involves the prediction and measurement update. For each particle, predicated states will be derived according to the previous estimated states and the motion model (2). Here it is necessary to add the random variances on theoretically predicated states, so that particles could randomly maneuver towards the true state. To update the weight of the particle, the robot has to obtain a measurement, and then the new weight is given by computing the measurement likelihood, which is:

$$p(Z_k|X_k) = \prod_{l=1}^L \frac{1}{\sqrt{2\pi R}} \exp\left(-\frac{1}{2} \cdot \left(\frac{z_k^l - z_l^k}{\sigma_k^l}\right)^T R^{-1} \left(\frac{z_k^l - z_l^k}{\sigma_k^l}\right)\right)$$ (9)

where $\hat{\cdot}$ refers to the estimation value with regard to its true value. $l$ denotes the index of the measurement pairs among $L$ measurement pairs received at a time instance $k$. $z$ is a measurement pair (i.e. $z = [b, r]^T$ in our cases) regarding to a certain landmark, which is as an element included in $Z$.

Usually we normalize the weights to sum to one. And to represent the state estimate, a common choice is to use the weighted average of all the particles, see Line 7 in Algorithm 2.

The second step is called re-sampling. In order to utilize the particles more efficiently, we hope to reuse the significant particles more times than the unimportant ones and this will keep more particles in the right place. Particles are wisely selected and pruned according to their weights. Specifically speaking, how likely one particle will appear again in the new particle set depends on how heavy this particle weights in the evolved particle set. Then after generating the new particle set, every particle in the set will be given an equal weight again, see Line 9-13 in Algorithm 2.

### C. FastSLAM algorithm

The basic idea of FastSLAM is that each particle independently contains the robot pose estimate as well as its own proposal on landmarks, like:

$$s = \{x \ y \ \theta \ \mu_{m1} \ \Sigma_{m1} \ \mu_{m2} \ \Sigma_{m2} \ \ldots \ \mu_{mn} \ \Sigma_{mn}\}$$ (10)

where $\mu_m = [x_m \ y_m]^T$ refers to the landmark state, and $\Sigma_m$ is its corresponding estimation error covariance. $N$ is the number of landmarks held by each individual particle.
In the FastSLAM solution, robot pose estimation is totally like that done in particle filter. When deriving map information, the received measurements are also used to correct the estimated states of the corresponding landmark by the EKF method, and each EKF for a single landmark correction. Here each landmark estimation error is independent from others. In each particle, each landmark states are updated according to:

\[
\mu_k = \mu_{k-1} + K_k (\hat{z}_k - \hat{z}_k) \\
\Sigma_k = (I - K_k H) \Sigma_{k-1}
\]

and

\[
K_k = \Sigma_{k-1} H^T W^{-1} \\
W = H \Sigma_{k-1} H^T + R
\]

where \(W\) is the single measurement covariance.

The re-sampling step is identical to the basic particle filter ideas above, i.e. particles are selected proportionally according to their weights. And here weight is calculated in the way:

\[
w = \prod_{i=1}^{L} \frac{1}{\sqrt{2\pi W}} \exp\left(-\frac{1}{2} (z_k - \hat{z}_k)^T (W)^{-1} (z_k - \hat{z}_k)\right)
\]

Once a new landmark is detected, it is added into the state vector in every particle. Newly detected landmarks states are initialized as:

\[
\mu_0 = h^{-1}(X_0, z) \\
\Sigma_0 = H^{-1} R (H^{-1})^T
\]

The flowchart in Fig. 2 illustrates the basic FastSLAM solution in a comparative all-round way.

Regarding the difference between the algorithms of FastSLAM 1.0 and FastSLAM 2.0, it is embodied in the methods how the proposal robot pose distribution is obtained. FastSLAM 1.0 samples the robot pose only based on the control signal, while FastSLAM 2.0 takes measurements into account besides the control signal, in which way it places the particles more efficiently, see Fig. 3. This is achieved by the following steps:

Firstly it predicts the robot pose \(\hat{X}_k\) by importing the control and the previous state estimate into motion model; Then the predicted measurement \(\hat{z}\) is calculated based on this predicted robot pose with the measurement model; Thirdly calculate the Jacobian linearization \(H_z\) and \(H_m\) of the measurement function \(h\) with regarding to robot pose and landmark states respectively; Finally the covariance and mean of the proposal robot pose distribution are:

\[
\Sigma_k = (H^T G^{-1} H + Q)^{-1} \\
\mu_k = \Sigma_k H^T G^{-1} (z - \hat{z}) + \hat{X}_k
\]

where \(G\) represents the measurement information which is calculated as the following:

\[
G = R + H_m \Sigma_m H_m^T
\]

V. SIMULATION RESULTS AND ANALYSIS

A. Simulation Setup

The simulation environment contains a mobile robot and 17 static landmarks on a 2D plane. The robot is equipped with a laser sensor which measures the bearing angles and the range lengths towards the landmarks. The maximum sensible range is restricted to a small value so that a certain landmark can be sensed only when the robot is close enough to it. In simulations, the robot travels along a certain pre-defined trajectory through all the landmarks. And in order to avoid data association problem, here we suppose that robot can match measurement pairs with exact corresponding landmark IDs. Throughout the cases the robot achieves simultaneous localization and mapping. Fig. 4 shows the whole process of exploration for the mobile robot in simulations.

In this section simulations for EKF-SLAM, FastSLAM 1.0 and FastSLAM 2.0 solutions are conducted in the same environment for comparisons.

---

Fig. 2. Flowchart of FastSLAM algorithm.

Fig. 3. Illustration of pose sampling applied in FastSLAM 2.0. The red circles are the true robot positions at two continuous time instances \(k - 1\) and \(k\). The black circle is the estimated position held by one particle at the time instance \(k - 1\), and the gray and white circles represents the pose predicted at the time instance \(k\) by the same particle but respectively using FastSLAM 1.0 and FastSLAM 2.0. The ellipse represents the probability density function of the proposal distribution derived by FastSLAM 2.0, where the dark color indicates the higher values. It is obviously shown that the white circle is much closer to the true position than the gray one.
tions it is obvious that particles around the newly detected landmark are loosely distributed, and then gradually converge along the time. Fig. 6 shows the converging process of particle distribution around a landmark.

D. Comparison of FastSLAM 1.0 and FastSLAM 2.0

As previously mentioned, FastSLAM 2.0 incorporates the measurements together with control to sample the robot pose. This improvement will make FastSLAM 2.0 solution distribute and use particles more efficiently which certainly leads to a better performance, especially when the control signal has worse accuracy than that of the measurements. Fig. 7 shows a selected segment in the simulation. Six sampled frames are depicted, while different kinds of sampled robot poses by FastSLAM 1.0 and FastSLAM 2.0 are generated and compared. The integral results clearly show that the sample robot poses by FastSLAM 2.0 are closer to the true robot pose.

E. Estimation Errors Comparison

To compare the performances of EKF, FastSLAM 1.0 and FastSLAM 2.0 algorithms applied for SLAM problems, three parallel sets of simulations respectively according to the mentioned algorithms are conducted, each set is run 100 times under the same conditions (same system noise and measurement noise, and same number of particles for FastSLAM 1.0 and FastSLAM 2.0). The average estimation errors of robot pose and landmarks are recorded in Table I, from which it is seen that both FastSLAM 1.0 and FastSLAM 2.0 solutions greatly improve the accuracy of the estimation results than that of EKF-SLAM solution, and FastSLAM 2.0 solution further shows off a better performance with smaller estimation errors than FastSLAM 1.0 does, due to the improved robot pose proposal distribution.

B. Multi-hypotheses of the landmark estimations

Inheriting the nature of EKF, the EKF-SLAM solution uses a single Gaussian to estimate the robot pose and all the landmark locations jointly. In this way, the estimation of a certain landmark only contains one hypothesis with its covariance. While on the contrary FastSLAM solution takes advantages of the multi-hypotheses characteristic of particle filter. FastSLAM on one hand estimates the robot pose under the particle filter structure, and on the other hand uses the separate EKFs in particles to estimate each landmark location. Under this structure, the estimation for one landmark contains \( M \) Gaussians, and each is proposed by one particle with different weights, which are depending on the scores that the particles get after incorporating the measurements. Fig. 5 gives an intuitive illustration. And it can be seen that particle \( n_1 \) has the best estimation of the landmark among all of the three selected particles. Due to its highest weight amongst, particle \( n_1 \) has the decisive role to influence the quality of the final estimation. While EKF-SLAM solution with the single estimation hypothesis characteristic is slightly inferior. Reasonably FastSLAM solution will avoid the loss of good estimation hypotheses towards a certain landmark and guarantee the high priority of good hypothesis that is close to the true location.

C. Convergence of Particles

When a new landmark is detected, its estimate locations held by different particles are initially diversified. In simulations it is obvious that particles around the newly detected landmark are loosely distributed, and then gradually converge along the time. Fig. 6 shows the converging process of particle distribution around a landmark.

VI. CONCLUSIONS

In conclusion, the modified particle filter, Rao-Blackwellized particle filter, overcomes the non-linearity limitation from EKF and curse of dimensionality from PF. FastSLAM solution takes advantages of the multi-hypotheses characteristic which reasonably avoids the loss of good estimation hypotheses towards a certain landmark. FastSLAM 2.0 improves the proposal distribution by incorporating the measurements, which results in more accuracy than FastSLAM 1.0.

VI. CONCLUSIONS

In conclusion, the modified particle filter, Rao-Blackwellized particle filter, overcomes the non-linearity limitation from EKF and curse of dimensionality from PF. FastSLAM solution takes advantages of the multi-hypotheses characteristic which reasonably avoids the loss of good estimation hypotheses towards a certain landmark. FastSLAM 2.0 improves the proposal distribution by incorporating the measurements, which results in more accuracy than FastSLAM 1.0.

VI. CONCLUSIONS

In conclusion, the modified particle filter, Rao-Blackwellized particle filter, overcomes the non-linearity limitation from EKF and curse of dimensionality from PF. FastSLAM solution takes advantages of the multi-hypotheses characteristic which reasonably avoids the loss of good estimation hypotheses towards a certain landmark. FastSLAM 2.0 improves the proposal distribution by incorporating the measurements, which results in more accuracy than FastSLAM 1.0.