International Journal of Automation and Computing 15(3), June 2018, 325-334
DOI: 10.1007/s11633-016-1050-y

An Improved FastSLAM Algorithm Based on Revised Genetic Resampling and SR-UPF

Tai-Zhi Lv Chun-Xia Zhao Hao-Feng Zhang

School of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing 210094, China

Abstract: FastSLAM is a popular framework which uses a Rao-Blackwellized particle filter to solve the simultaneous localization and mapping problem (SLAM). However, in this framework there are two important potential limitations, the particle depletion problem and the linear approximations of the nonlinear functions. To overcome these two drawbacks, this paper proposes a new FastSLAM algorithm based on revised genetic resampling and square root unscented particle filter (SR-UPF). Double roulette wheels as the selection operator, and fast Metropolis-Hastings (MH) as the mutation operator and traditional crossover are combined to form a new resampling method. Amending the particle degeneracy and keeping the particle diversity are both taken into considerations in this method. As SR-UPF propagates the sigma points through the true nonlinearity, it decreases the linearization errors. By directly transferring the square root of the state covariance matrix, SR-UPF has better numerical stability. Both simulation and experimental results demonstrate that the proposed algorithm can improve the diversity of particles, and perform well on estimation accuracy and consistency.

Keywords: Simultaneous localization and mapping (SLAM), genetic algorithm, square root unscented particle filter (SR-UPF), fast Metropolis-Hastings (MH), double roulette wheels.

1 Introduction

Localization and incremental mapping occur simultaneously when a mobile robot is navigating in an unknown environment, the so called simultaneous localization and mapping[1, 2]. This process is commonly abbreviated as simultaneous localization and mapping problem (SLAM), and is also known as concurrent mapping and localization (CML). The idea of SLAM was introduced at the 1986 IEEE Robotics and Automation Conference, and is considered by many to be a key prerequisite for truly autonomous robots. SLAM has been applied to a number of different applications, stretching from search and rescue, over reconnaissance to commercial products[2]. However, SLAM research still faces many challenging problems, such as large-scale and complex environments, reliable data association, nonlinearity, and unknown priori knowledge.

From a probabilistic perspective, SLAM approaches are divided into two groups: offline and online. Offline SLAM is introduced by Lu and Milios (1997) and it optimizes the complete trajectory estimation and map after all data has been recorded. Batch algorithms such as smoothing and mapping (SAM)[3] and GraphSLAM[4] are offline methods. SAM involves not just the most current robot location, but the entire robot trajectory up to the current time[4]. GraphSLAM transforms the SLAM posterior into a graphical network, representing the log-likelihood of the data[3]. Online SLAM involves estimating the posterior over the momentary pose along with the map. Probabilistic approaches, i.e., extended Kalman filters (EKF) and particle filters (PF) have become dominant in the online SLAM research. EKF and PF are both mathematical derivations of the recursive Bayes rule. For computational complexity, EKF-SLAM is not suitable in real-time and large environments[5]. FastSLAM is an instance of Rao-Blackwellized particle filter (RBPF), which separates the full SLAM posterior into a product of a robot path posterior and landmark posteriors conditioned on the robot path estimation[6]. By the factorization, it needs less memory usage and computational time. Each particle has its respective map and performs its own data association in FastSLAM algorithm. This multi-hypothesis data association ability makes FastSLAM more robust to data association problems than EKF-SLAM. There are many successful implementations of FastSLAM to solve different SLAM applications. In [7], FastSLAM was chosen to solve the visual SLAM problem. Chen et al.[8] extended FastSLAM from the single-robot SLAM to the multi-robot case.

FastSLAM, however, has some drawbacks as well. One is that FastSLAM linearizes the motion model in the same manner as EKF-SLAM. Inaccurate approximation of the nonlinear function leads to filter divergence[9]. To reduce the linearization errors, unscented particle filter (UPF) was introduced to estimate the robot pose and landmark estimations in the FastSLAM framework[10, 11]. This method combines particle filter with unscented Kalman filter (UKF)

Research Article
Manuscript received March 4 2014; accepted May 9, 2014; published online December 29, 2015
This work was supported by National Natural Science Foundation of China (No. 61101197) and Research Fund for the Doctoral Program of Higher Education of China (No. 20093219120025).
Recommended by Associate Editor Min Tan

which extracts the sigma points from the Gaussian and passes these through the nonlinear function. By avoiding the linearizion based on Taylor series expansion, UKF outperforms the EKF in accuracy and robustness. More recently, the central difference particle filter and robust iterated sigma point methods were introduced, and they are both similar to UKF[12, 13].

The other noticeable drawback is the particle depletion. FastSLAM has been known to degenerate over time, and it is impossible to be prevented as proven by Kong-Liu-Wong theorem[6, 14, 15]. Resampling process aims for amending the particle degeneracy, but it always leads to lose the particle diversity, the so called particle depletion. A reasonable design of the resampling process plays an important role, and avoiding loss of diversity remains an ongoing topic in the application domain of particle filter such as target tracking, SLAM, etc. Liu[16] introduced the effective number of particles to estimate how well the current particle set represents the true posterior. After that, most resampling algorithms use it as an indicator to determine when to resample, and drastically reduce the risk of replacing good particles[17]. Many resampling approaches have been researched in the field of particle filters, and the most often encountered algorithms are systematic resampling, residual resampling, multinomial resampling, stratified resampling. Although they differ mainly in the generation of ordered sequence of random numbers, all of them simply replace lower weighted particles with higher weighted particles[18, 19]. Auxiliary particle filter (APF) and regularized particle filter (RPF) are another two variants of sequential importance sampling/resampling (SISR) algorithm, yet the efficiency of the above methods can further be improved[19]. Some biological evolution algorithms have been newly proposed to keep particle diversity as long as possible. Genetic resampling, which produces a new particle generation by selection, crossover and mutation operators, can increase the diversity of particles in state space and eliminate the disadvantage of random resampling[20, 21].

Two approaches to increase the performance of the FastSLAM algorithm are presented in this paper.

1) Square-root UKF (SR-UKF) has better numerical properties and guarantees positive semi-definiteness of the underlying state covariance. SR-UPF which combines SR-UKF and particle filter samples particles in a highly accurate manner.

2) A revised resampling technique, which uses a genetic method based on the fast metropolis-hastings (MH) and double roulette wheels, reduces the risk of particle depletion and increases the diversity.

The rest of the paper is organized as follows. In Section 2, the SLAM problem and FastSLAM2.0 algorithm are presented in brief. SR-UPF and a revised genetic resampling method are proposed in Sections 3.1 and 3.2, respectively. The realizations of the proposed algorithm is presented in Section 3.3. Section 4 experimentally compares the proposed algorithm with FastSLAM2.0 by using simulation environment and "Car Park Dataset". Finally, Section 5 presents the conclusions.

2 Background

2.1 SLAM problem

To describe SLAM, the robot pose at time t is denoted as x_t, and the entire map is denoted as Θ. The control and observation at time t are represented as u_t and z_t, respectively. The standard motion and observation models as nonlinear functions with independent Gaussian noise are as follows:

$$x_t = g(x_{t-1}, u_t) + \varepsilon \tag{1}$$

$$z_t = h(x_t, \Theta) + \delta \tag{2}$$

where g and h are nonlinear functions, and ε and δ are Gaussian disturbances with covariance Q and R, respectively.

In the Bayesian probabilistic framework, the online SLAM problem attempts to estimate the posterior probability distribution over all possible maps and robot poses conditioned on the full set of controls and observations at time t.

$$p(x_t, \Theta | z_{1:t}, u_{1:t}). \tag{3}$$

The following recursive formula, known as the Bayes Filter, is used to compute the posterior of (3). By using the Bayes rule, the law of total probability, and the Markov hypothesis[1], the online SLAM problem can be rewritten as

$$\underbrace{p(x_t, \Theta | z_{1:t}, u_{1:t})}_{\substack{\text{The posterior distribution} \\ \text{at time } t}} \propto \underbrace{p(z_t | x_t, \Theta)}_{\substack{\text{The observation} \\ \text{model}}} \int \underbrace{p(x_t | x_{t-1}, u_t)}_{\substack{\text{The motion} \\ \text{model}}} \underbrace{p(x_{t-1}, \Theta | z_{1:t-1}, u_{1:t-1})}_{\substack{\text{The posterior distribution} \\ \text{at time } t\text{-}1}} \mathrm{d}x_{t-1} \tag{4}$$

In general, the integral of (4) cannot be evaluated in closed form. EKF-SLAM and FastSLAM are simply approximations of the general Bayes filter[5].

2.2 FastSLAM 2.0 algorithm

The posterior distribution of (4) possesses no closed form from which we can easily draw samples. FastSLAM 2.0 is an efficient algorithm based on a straightforward factorization. This factorization separates the full SLAM posterior into a product of a robot path posterior and N landmark posteriors conditioned on the robot path[6].

$$p(x_{1:t}, \Theta | z_{1:t}, u_{1:t}) = p(x_{1:t} | z_{1:t}, u_{1:t}) p(\Theta | z_{1:t}, u_{1:t}) = p(x_{1:t} | z_{1:t}, u_{1:t}) \prod_{j=1}^{N} p(m_j | z_{1:t}, u_{1:t}) \tag{5}$$

where $x_{1:t}$ is the path of the robot from the start to time t.

The decomposed posterior in (5) can be approximated by a particle filter, with each particle representing a sample of the robot path[6]. Attached to each particle, N independent landmark estimations are implemented as EKFs. At time t, the i-th particle is described by the following equation.

$$X_t^{[i]} = < (x_t^{[i]},\ \omega_t^{[i]}),\ (\mu_t^{[i,1]},\ \Sigma_t^{[i,1]}),\ (\mu_t^{[i,2]},\ \Sigma_t^{[i,2]}),\ \cdots,\ (\mu_t^{[i,j]},\ \Sigma_t^{[i,j]}),\ \cdots,\ (\mu_t^{[i,N]}, \Sigma_t^{[i,N]}) > \tag{6}$$

where $x_t^{[i]}$ is the estimated robot pose of the i-th particle, and $\omega_t^{[i]}$ is the weight of this particle. $(\mu_t^{[i,j]},\ \Sigma_t^{[i,j]})$ are the mean and the covariance matrix of the Gaussian representing the j-th landmark conditioned on this particle.

At time t, the updating flow of each particle is shown in Fig. 1.

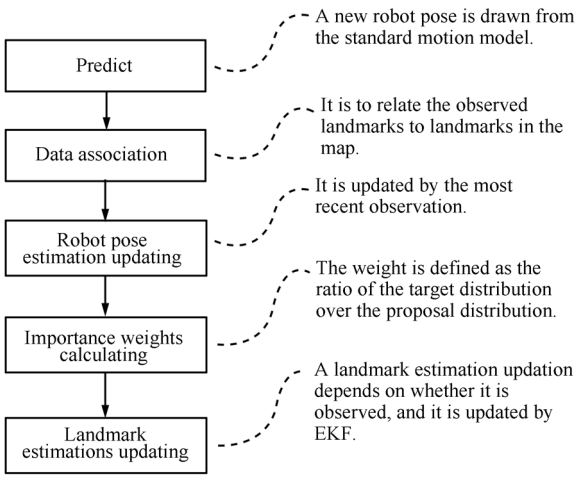


Fig. 1 Flowchart of each particle updating in FastSLAM 2.0

1) Predict

Instead of sampling a new robot pose from the motion model in FastSLAM 1.0, FastSLAM 2.0 samples a new pose from the motion model and the most recent observation z_t.

$$x_t^{[i]} \ \sim\ p(x_t \mid x_{t-1}^{[i]},\ u_t,\ z_t). \tag{7}$$

A new robot pose is drawn from the motion model and updated by the most recent observation.

$$\hat{x}_t^{[i]} \ =\ g(x_{t-1}^{[i]},\ u_t) \tag{8}$$

where $\hat{x}_t^{[i]}$ is the predicted pose of the i-th particle from the motion model at time t. By incorporating the current observation into the proposal distribution, FastSLAM 2.0 can get better match the posterior, and it is superior to FastSLAM 1.0 in nearly all respects[6].

2) Data association

Data association is the process of relating landmarks observed in the environment to landmarks in the map.

3) Update the robot pose estimation

In this step, the robot pose will be updated by the most recent observation z_t.

$$\hat{z}_t^{[i]} = h(\hat{x}_{t-1}^{[i]},\ \Theta) \tag{9}$$

$$x_t^{[i]} = \hat{x}_t^{[i]} + \beta(z_t - \hat{z}_t^{[i]}) \tag{10}$$

where $\hat{z}_t^{[i]}$ is the predicted observation, and β is calculated from the landmark estimation covariance matrices, Q_t, R_t and the Jacobian matrices.

4) Calculate the importance weight

In the FastSLAM framework, the importance weight is defined as the ratio of the target distribution over the proposal distribution.

$$\omega_t^{[i]} = \frac{\text{Target distribution}}{\text{Proposal distribution}} = \frac{p(x_{1:t}^{[i]}|z_{1:t}, u_{1:t})}{q(x_{1:t}^{[i]}|z_{1:t}, u_{1:t})}. \tag{11}$$

The proposal distribution $q(x_{1:t}^{[i]}|u_{1:t}, z_{1:t})$ can be represented by a recursive form, as

$$q(x_{1:t}^{[i]}|z_{1:t}, u_{1:t}) = q(x_t^{[i]}|x_{t-1}^{[i]}, z_t, u_t) q(x_{t-1}^{[i]}|z_{1:t-1}, u_{1:t-1}, x_0). \tag{12}$$

The Bayes rule is used to calculate the importance weight, as follows:

$$\omega_t^{[i]} \propto \frac{p(z_t|x_t^{[i]}) p(x_t^{[i]}|x_{t-1}^{[i]}, u_t) p(x_{t-1}^{[i]}|x_{1:t-1}^{[i]}, z_{1:t-1}, u_{1:t-1}, x_0)}{q(x_t^{[i]}|x_{t-1}^{[i]}, z_t, u_t) q(x_{t-1}^{[i]}|z_{1:t-1}, u_{1:t-1}, x_0)} = \omega_{t-1}^{[i]} \frac{p(z_t|x_t^{[i]}) p(x_t^{[i]}|x_{t-1}^{[i]}, u_t)}{q(x_t^{[i]}|x_{t-1}^{[i]}, z_t, u_t)}. \tag{13}$$

The choice of the proposal distribution is one of the most critical issues in the FastSLAM framework. The choice in FastSLAM 1.0 is the transitional prior.

$$q(x_t^{[i]}|x_{t-1}^{[i]}, z_t, u_t) = p(x_t^{[i]}|u_t, x_{t-1}^{[i]}). \tag{14}$$

The proposal distribution in FastSLAM 2.0 is as

$$q(x_t^{[i]}|x_{t-1}^{[i]}, z_t, u_t) = p(x_t^{[i]} \mid x_{t-1}^{[i]},\ u_t,\ z_t) \tag{15}$$

where $\omega_t^{[i]}$ is normalized as

$$\bar{\omega}_t^{[i]} = \frac{\omega_t^{[i]}}{\sum_{k=1}^{M} w_t^{[k]}} \tag{16}$$

where M is the number of particles.

5) Update landmark estimations

Since the landmark estimations are conditioned on the robot pose, N EKFs are attached to each particle. In [6], a landmark estimation updating depends on whether the landmark is observed at time t. If a landmark is newly observed, its mean and the covariance are initialized as follows:

$$\mu_t^{[i,N+1]} = h^{-1}(z_t, x_t^{[i]}) \tag{17}$$

$$\Sigma_t^{[i,N+1]} = H_{rz} R_t H_{rz}^{\mathrm{T}}. \tag{18}$$

where the matrix H_{rz} is the Jacobian of h^{-1}. If the j-th landmark in the map is not observed, the estimation remains unchanged.

$$\mu_t^{[i,j]} = \mu_{t-1}^{[i,j]} \tag{19}$$

$$\Sigma_t^{[i,j]} = \Sigma_{t-1}^{[i,j]}. \tag{20}$$

If the j-th landmark in the map is observed, the updating is specified through the following equations.

$$\mu_t^{[i,j]} = \mu_{t-1}^{[i,j]} + K_t(z_t - \hat{z}_t^{[i]}) \tag{21}$$

$$\Sigma_t^{[i,j]} = (I - K_t H_z)\Sigma_{t-1}^{[i,j]} \tag{22}$$

where K_t is the Kalman gain coefficient, and the matrix H_z is the Jacobian of h.

After all particles are updated, the number of effective particles is used as a robust indicator to determine when to resample[18]. The number is defined as

$$N_{eff} = \frac{1}{\sum\limits_{k=1}^{M}(\bar{\omega}_t^{[k]})^2}. \tag{23}$$

When N_{eff} is less than a given threshold, the resampling process would be performed. After that, all particle weights are reset to

$$\omega_t^{[j]} = \frac{1}{M}. \tag{24}$$

3 Combine SR-UPF with the revised genetic resample in FastSLAM framework

3.1 SR-UPF algorithm

SR-UPF combines the particle filter with square root unscented Kalman filter (SR-UKF) to sample over robot paths. In contrast to UPF, SR-UPF can have the added benefit of numerical stability and guaranteed positive semi-definiteness of the state covariances[22]. SR-UPF makes use of three linear algebra techniques[23]. QR decomposition, Cholesky factor updating and efficient least squares are briefly review below:

1) QR decomposition: A QR decomposition (also called a QR factorization) is often used to solve the linear least squares problem, and is the basis for a particular eigenvalue algorithm, the QR algorithm. The QR decomposition of matrix of A is given by

$$A^{\mathrm{T}} = QR \tag{25}$$

where A is an m-by-n matrix, Q is an m-by-m unitary matrix and R is an m-by-n upper triangular matrix. The shorthand notation $qr\cdot$ to denote a QR decomposition of a matrix where only R is returned.

2) Cholesky factor updating: The cholesky factor updating is give by

$$S1 = cholupdate(S, x, \pm U) \tag{26}$$

where S is the original Cholesky factory of $P = AA^{\mathrm{T}}$. Cholupdate is rank 1 update to Cholesky factorization, and returns $S1 = P \pm \sqrt{U}xx^{\mathrm{T}}$. This algorithm is available in Matlab as cholupdate.

3) Efficient least squares: The solution of the equation $(AA^{\mathrm{T}})x = A^{\mathrm{T}}b$ corresponds to the overdetermined least squares problem $Ax = b$. The solution can be solved efficiently by a QR decomposition with pivoting which is implemented by the "/" operator in Matlab.

The implementation of SR-UPF is as follows:

1) Initialize

Suppose that x_0 and P_0 are the mean and the covariance of the robot pose estimation at the start, the mean, the square root of the covariance, and a set of scalar weights are initialized.

$$\hat{x}_0 = x_0, \quad S_0 = chol(P_0) \tag{27}$$

where the square root of the covariance, S_0, is calculated by a Cholesky factorization which decomposes a symmetric, positive-definite matrix into the product of a lower-triangular matrix and its transpose.

$$\omega_0^m = \frac{\lambda}{L+\lambda},\ \omega_0^c = \frac{\lambda}{L+\lambda} + (1 - \alpha^2 + \beta) \tag{28}$$

$$\omega_i^m = \omega_i^c = \frac{1}{2(L+\lambda)},\quad i = 1, 2, \cdots, L, \cdots, 2L \tag{29}$$

where $\{\omega_i\}$ is a set of the scalar weights. L is the dimension of the state variable, and $\lambda = \alpha^2(L + \kappa)$ is a scaling parameter. The constant α determines the spread of the sigma points around $\hat{x}$, and β is used to incorporate prior knowledge of the distribution of state variable ($\beta = 2$ is optimal for Gaussian distribution), and κ is a secondary scaling parameter.

2) Calculate the sigma points

$$X_{t-1} = [\hat{x}_{t-1} \quad \hat{x}_{t-1} + \eta S_{t-1} \quad \hat{x}_{t-1} - \eta S_{t-1}] \tag{30}$$

where S_{t-1} is the square root of the covariance at time $t-1$, and the scaling constant η is calculated from

$$\eta = \sqrt{L + \lambda}. \tag{31}$$

3) Update by the control data

The sigma points are passed through the nonlinear motion model.

$$X_t^* = g(X_{t-1}, u_t). \tag{32}$$

The estimated mean and square root of the covariance are calculated from the transformed sigma points using

$$\hat{x}_t = \sum_{i=0}^{2L} \omega_i^m X_{t,i}^* \tag{33}$$

$$\hat{S}_t = qr\{[\sqrt{(\omega_i^c)}(X_{t,1:2L}^* - \hat{x}_t) \quad \sqrt{Q}]\} \tag{34}$$

$$\hat{S}_t = cholupdate\{\hat{S}_t,\ X_{t,0}^* - \hat{x}_t,\ \omega_0^c\} \tag{35}$$

where Q is the covariance matrix of the control noise. $\hat{S}_t$ is calculated by a QR decomposition in (34), and updated by a

Cholesky updating operation in (35). Since the weight may be negative, the subsequent Cholesky updating operation is necessary.

$$X_t = [\hat{x}_t \quad \hat{x}_t + \eta\hat{S}_t \quad \hat{x}_t - \eta\hat{S}_t]. \tag{36}$$

The transformed sigma points are then used to predict the observation by the nonlinear observation model.

$$\hat{Z}_t = h(X_t, \Theta) \tag{37}$$

$$\hat{z}_t = \sum_{i=0}^{2L} \omega_i^m \hat{Z}_{t,i}. \tag{38}$$

4) Update by the observation data

The same two steps, QR decomposition and Cholesky updating operation are both applied to the calculation of $\hat{S}_{Z_t}$.

$$\hat{S}_{Z_t} = qr\{[\sqrt{(\omega_1^c}(\hat{Z}_{t,1:2L} - \hat{z}_t) \quad \sqrt{R}]\} \tag{39}$$

$$\hat{S}_{Z_t} = cholupdate\{\hat{S}_{Z_t},\ \hat{Z}_{t,0} - \hat{z}_t,\ \omega_0^c\} \tag{40}$$

where R is the covariance matrix of the observation noise.

In order to determine how much to adjust the estimated mean and covariance based on the observation data, the Kalman gain matrix K_t is calculated

$$P_{x_t z_t} = \sum_{i=0}^{2L} \omega_i^c (X_{t,i} - \hat{x}_t)(\hat{Z}_{t,i} - \hat{z}_t)^{\mathrm{T}} \tag{41}$$

$$K_t = \frac{P_{x_t z_t}/\hat{S}_{Z_t}^{\mathrm{T}}}{\hat{S}_{Z_t}}. \tag{42}$$

The nested inverse (or least squares) solutions are used in (42). Since $\hat{S}_{Z_t}$ is square and triangular, efficient "back-substitutions" can be used to solve for K_t directly without the need for matrix inversion.

Finally, the mean and covariance are updated by

$$x_t = \hat{X}_t + K_t(z_t - \hat{z}_t) \tag{43}$$

$$U = K_t \hat{S}_{Z_t} \tag{44}$$

$$S_t = cholupdate(\hat{S}_t, U, -1). \tag{45}$$

3.2 Genetic resampling based on fast metropolis-hastings mutation and double roulette wheels selection

Genetic algorithm (GA) is an artificial intelligence procedure. It is based on the theory of natural selection and evolution. GA evolves to the next generation by genetic operators such as selection, crossover and mutation. GA used in resampling process can improve the diversity of particles, but the mutation strategy in genetic resampling process has an important influence on the quality of new generation particles. Fast Metropolis-Hastings (MH) mutation can solve the divergence problem of the traditional mutation and produce the typical particles to reflect the target density more quickly[24].

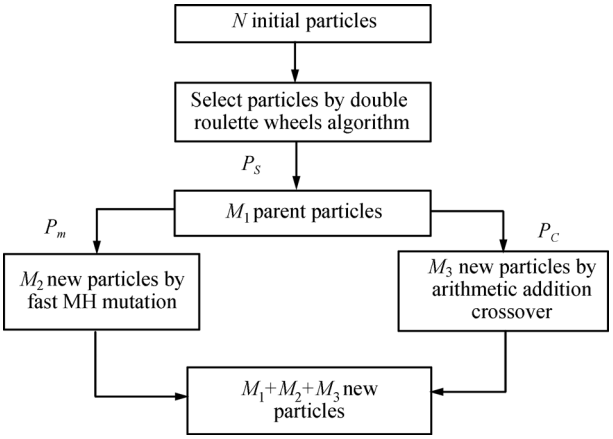


Fig. 2 Flow of the genetic resampling based on fast MH mutation and double roulette wheels selection

The process of the genetic resampling algorithm based on fast MH mutation and double roulette wheels selection is described in Fig. 2, where P_s, P_c and P_m are probabilities of selection, crossover and mutation, respectively. Let P_s be 0.5, and the values of P_c and P_m depend on the particle set diversity which is represented as V_d.

$$V_d = \frac{H_t}{H_{best}} \tag{46}$$

$$H_t = \sum_{i=1}^{M} f_2(x_t^{[i]}) \tag{47}$$

$$f_2(x_t^{[i]}) = \sum_{j=1}^{M}(\sum_{k=1}^{L} |x_t^{[i,k]} - x_t^{[j,k]}|) \tag{48}$$

where L is the dimension of the state variable, and M is the number of particles. The function $f_2(x_t^{[i]})$ is the sum of the Hamming distance between particle $x_t^{[i]}$ and the other particles. H_{best} represents maximum value of H from the start till time t. P_c and P_m are calculated as

$$\begin{cases} P_c = 0.8, P_m = 0.2 & \dfrac{3}{4} < V_d \le 1 \\ P_c = 0.6, P_m = 0.4 & \dfrac{1}{2} < V_d \le \dfrac{3}{4} \\ P_c = 0.4, P_m = 0.6 & \dfrac{1}{4} < V_d \le \dfrac{1}{2} \\ P_c = 0.3, P_m = 0.7 & 0 < V_d \le \dfrac{1}{4}. \end{cases} \tag{49}$$

1) Selection

Particles which breed a new generation are selected by a double roulette wheels algorithm. This method requires the following steps.

Step 1. Sort particles by a fitness function $f(x_i)$.

Step 2. Generate a uniform random number r between $[0, \sum_{i=1}^{M} Fitness(x_i)]$.

Step 3. Select the particle if $q_{i-1} < r \le q_i$, where $q_i = \sum_{j=1}^{i} Fitness(x_j)$ and $q_0 = 0$.

Step 4. Repeat Steps 2 and 3 until the number of new particles reaches a given value.

Two different fitness functions are used to build double

roulette wheels. The first fitness function is the particle weight which reflects the pros and cons of a path estimation.

$$f_1(x_t^{[i]}) = \omega_t^{[i]}. \tag{50}$$

As a measure of the particle diversity, the function $f_2(x_t^{[i]})$ in (48) is used to improve the particle diversity as the second fitness function. Two particle sets X_{f1} and X_{f2} are drawn from the different roulette wheels. The lengths of two particle sets are both $\frac{M}{4}$. If there are repeated particles between the two particle sets, the double roulette wheels algorithm is used to generate new particles till all the particles from the two particle sets are different.

2) Crossover

Two subsets $\tilde{X}_{f1}$ and $\tilde{X}_{f2}$ are randomly drawn from X_{f1} and X_{f2}, respectively. The lengths of $\tilde{X}_{f1}$ and $\tilde{X}_{f2}$ are both $\frac{P_c M}{2}$. New particles are defined as

$$x_i^{new} = \alpha \tilde{X}_{f1,i} + (1-\alpha)\tilde{X}_{f2,i} \tag{51}$$

where α is a uniform random number in $[0.3, 0.7]$.

3) Mutation

New particles are generated by the proposal function $q(x, \widetilde{x})$, which is a spherically symmetric random walk:

$$q(x, \widetilde{x}) = q(|x - \widetilde{x}|) \propto \mathrm{e}^{\frac{(x-\bar{x})^2}{2\sigma^2}} \tag{52}$$

where σ is the jump size, and adjusted by the covariance of the path estimation. Once the new candidates are generated, the algorithm keeps or discards them according to the acceptance ratio $a(x, \widetilde{x})$.

$$a(x, \widetilde{x}) = \frac{p(z|\widetilde{x})q(x, \widetilde{x})}{p(z|x)q(\widetilde{x}, x)}. \tag{53}$$

According to the acceptance ratio, only the first $\frac{P_m M}{2}$ of new candidates are kept.

3.3 Realization of the proposed FastSLAM algorithm

Compared with FastSLAM 2.0, the proposed algorithm has two important improvements. One is to use genetic algorithm based on fast MH mutation and double roulette wheels selection in resampling process. The other is to use SR-UPF in sampling process.

The flow of the proposed FastSLAM algorithm is shown in Fig. 3.

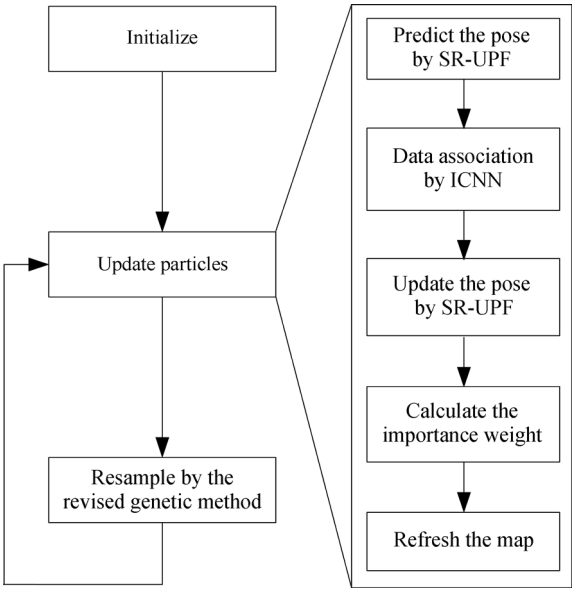


Fig. 3 Comparison between FastSLAM 2.0 and the proposed algorithm

1) Initialize

According to the particle number and the noise covariance matrices of the control and observation, the initial set is generated.

2) Update particles

By the following steps, each particle is updated.

Step 1. Predicting the pose. The predicted robot pose is drawn by (30) – (38).

Step 2. Data association. This paper adopts individual compatibility nearest neighbor (ICNN) to implement data association[25].

Step 3. Updating the pose. The robot pose is updated by (39) – (45).

Step 4. Calculating the important weight and normalizing it.

Step 5. Refreshing the map. The flow is same as that of FastSLAM 2.0.

3) Resample

The revised genetic resampling would be carried out to generate a new particle set when N_{eff} drops below a given threshold of $\frac{M}{2}$, where M is the number of particles. After resampling, all particle weights are reset to $\frac{1}{M}$.

4 Experimental analysis

4.1 Simulation experiment

In the simulation environment, the SLAM state is described by the robot pose (position and heading) and landmark locations. The state at time t is represented by a joint state-vector X_t.

$$X_t = [x_t, \Theta] = [(x_{x_t}, y_{x_t}, \theta_{x_t}), (x_{m_1}, y_{m_1}), \cdots, (x_{m_j}, y_{m_j}), \cdots, (x_{m_N}, y_{m_N})]. \tag{54}$$

The map Θ does not have time subscript as landmarks are modeled as stationary. The motion and observation models are as follows:

$$x_t = g(x_{t-1}, u_t) = \begin{bmatrix} x_{x_{t-1}} + \Delta T V_t \cos(\theta_{x_{t-1}} + \alpha_t) \\ y_{x_{t-1}} + \Delta T V_t \sin(\theta_{x_{t-1}} + \alpha_t) \\ \theta_{x_{t-1}} + \dfrac{\Delta T V_t \sin(\alpha_t)}{L} \end{bmatrix} + \varepsilon \quad (55)$$

$$z_t(m_j) = h(x_t, m_j) = [l_t(m_j), \beta_t(m_j)]^{\mathrm{T}} = \begin{bmatrix} \sqrt{(x_{m_j} - x_{x_t})^2 + (y_{m_j} - y_{x_t})^2} \\ \arctan(\dfrac{y_{m_j} - y_{x_t}}{x_{m_j} - x_{x_t}}) - \theta_{x_t} \end{bmatrix} + \delta. \quad (56)$$

The control and the observation are denoted as u_t and z_t, respectively. u_t includes the velocity V_t and steering angle α_t. While $z_t(m_j)$ represents the range-bearing observation from the robot to the j-th landmark, $l_t(m_j)$ is the distance, and $\beta_t(m_j)$ is the angle from 0 to 2π clockwise. L is the wheel-base.

The comparison between the proposed algorithm and FastSLAM 2.0 is based on a Matlab SLAM simulator which is implemented by Tim Bailey[26]. The simulation environment is a 250 m×200 m area with 135 landmarks. The robot moves at a speed of 3 m/s, and the maximum steering angle G is 30°. The speed noise ε_v is 0.4 m/s, and the angle noise ε_β is 3°. It equips a range bearing sensor with a 180° frontal view and a maximum range of 30 m. The range noise of observations δ_l is 0.3 m, and the angle noise δ_β is 3°. The control and observation frequency are 40 Hz and 10 Hz, respectively.

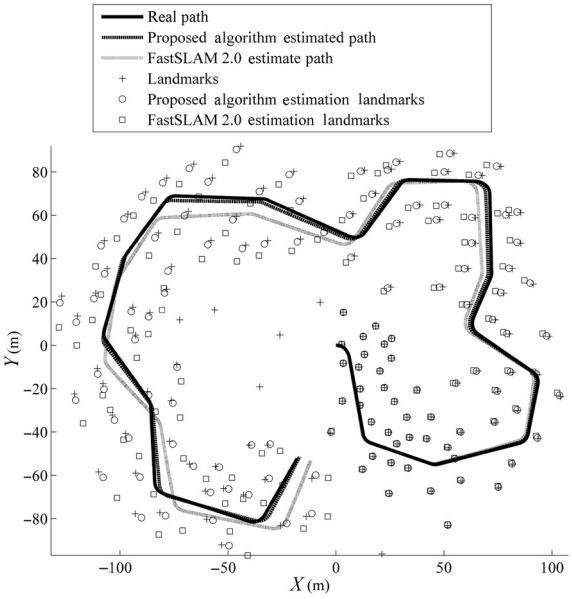


Fig. 4 Comparison of the distance errors between FastSLAM 2.0 and the proposed algorithm

Fig. 4 shows the comparison of the robot path and landmark estimations between FastSLAM 2.0 and the proposed FastSLAM algorithm with 100 particles. It is noticeable that the estimated path by the proposed algorithm is almost the same as the actual path whereas there are some errors between the path acquired by FastSLAM 2.0 and the actual path. Obviously, the proposed FastSLAM algorithm has higher precision than FastSLAM 2.0 in estimations of the robot′s motion path and landmark positions.

In the simulation of Fig. 4, the robot is assumed to run for about 200 s. Fig. 5 shows the comparison of the robot position error over times between two algorithms. Two curves represent the distance varies between the actual robot position and estimated position. It can be seen that the robot position error of FastSLAM 2.0 is more than that of the proposed algorithm. Also, the robot position error of FastSLAM 2.0 increases quickly than that of the proposed algorithm. Fig. 6 shows the comparison of the landmark position error between two algorithms after the loop is closed. It can be seen the landmark estimation error of proposed algorithm is fewer than that of FastSLAM2.0.

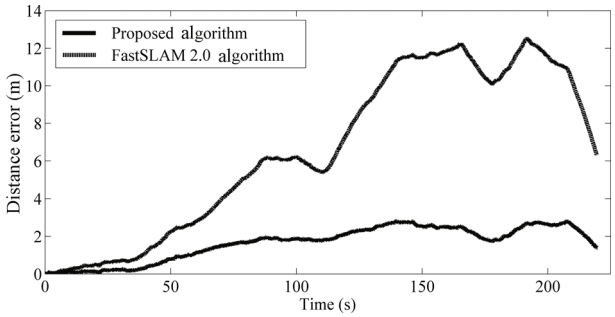


Fig. 5 Comparison of the average NEES between FastSLAM 2.0 and the proposed algorithm

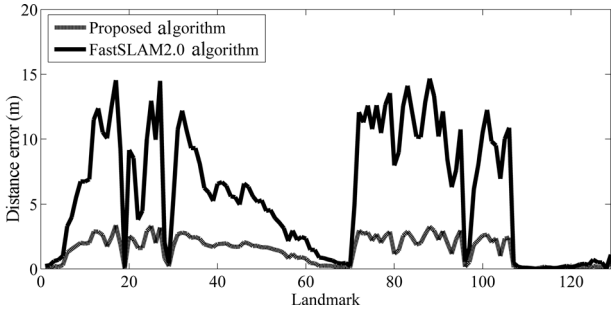


Fig. 6 The comparison of the landmark position errors between FastSLAM 2.0 and the proposed algorithm after the loop is closed

Due to the randomness of the sampling process and noises, the result of each experiment is different. In order to get more detailed and accurate evaluation between the two algorithms, 50 simulation experiments were carried out. Table 1 shows the result of these experiments, and RMSE represents root mean square error. The result shows that the proposed algorithm is more accurate than FastSLAM 2.0 in estimations of the robot pose and landmark locations, and the running time of the proposed algorithm is

more than the FastSLAM 2.0 algorithm.

The average normalized estimation error squared (NEES)[14] over N Monte Carlo runs of the filter is used to measure filter consistency.

$$\epsilon_k = (x_t - \hat{x}_t)^{\mathrm{T}} P_t^{-1} (x_t - \hat{x}_t) \tag{57}$$

where x_t and P_t are the estimated mean and covariance of the robot pose, and $\hat{x}_t$ is the real pose.

Given N runs, the average NEES is computed as

$$\overline{\epsilon}_k = \frac{1}{N} \sum_{i=1}^{N} \epsilon_{i_k}. \tag{58}$$

Table 1 Comparison of experimental results between FastSLAM 2.0 and the proposed algorithm

Algorithm	RMSE of the robot position estimation	RMSE of the landmark position estimations	average running time(s)
Proposed algorithm	3.653 3	3.563 1	34.15
FastSLAM 2.0	6.284 3	5.779 4	27.83

For the 3-dimensional vehicle pose and $N = 50$, the two-sided 95% probability region for $\hat{\epsilon}_k$ is bounded by the interval [2.36, 3.72]. If NEES exceeds the upper bound, the filter is optimistic[27].

The average NEES comparison between two algorithms is shown in Fig. 7. It shows that FastSLAM 2.0 algorithm becomes rapidly optimistic, and that the average NEES in the proposed FastSLAM algorithm maintains low level for a long time.

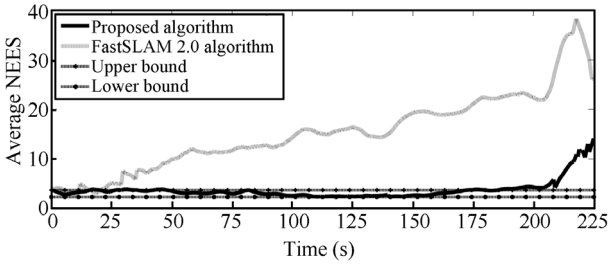


Fig. 7 Comparison of the average NEES between FastSLAM 2.0 and the proposed algorithm

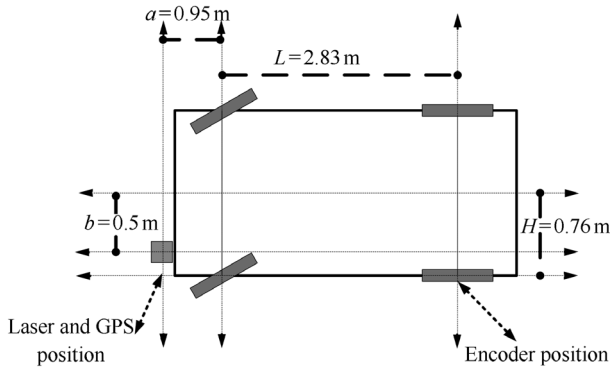


Fig. 8 Motion model of the truck

4.2 Experiment with "Car Park Dataset"

The experiments were carried out with "Car Park Dataset"[28] which is popular in the SLAM research community. A truck equipped with a GPS, an inertial sensor and a laser sensor collected the dataset in an open parking area with artificial beacons. The motion model of the truck is shown in Fig. 8. The motion and observation models are as

$$x_t = g(x_{t-1}, u_t) = \begin{bmatrix} x_{x_{t-1}} + \Delta T V_t \cos(\theta_{x_{t-1}} + \alpha_t) - \\ \dfrac{\tan(\alpha_t)}{L}(a\sin(\theta_{x_{t-1}} + b\cos(\theta_{x_{t-1}})) \\ y_{x_{t-1}} + \Delta T V_t \sin(\theta_{x_{t-1}} + \alpha_t) - \\ \dfrac{\tan(\alpha_t)}{L}(b\sin(\theta_{x_{t-1}} + a\cos(\theta_{x_{t-1}})) \\ \theta_{x_{t-1}} + \dfrac{\Delta T V_t \tan(\alpha_t)}{L} \end{bmatrix} + \varepsilon \tag{59}$$

$$V_t = \frac{V_{et}}{1 - \dfrac{H}{L\tan(\alpha_t)}} \tag{60}$$

where V_t is the velocity of the center of the axle, v_{et} is the velocity of the rear left wheel, ΔT is the interval of sampling, α_t is the steering angle, and ε is the control noise.

$$z_t(m_j) = h(x_t, m_j) = [l_t(m_j), \beta_t(m_j)]^{\mathrm{T}} \begin{bmatrix} \sqrt{(x_{m_j} - x_{x_t})^2 + (y_{m_j} - y_{x_t})^2} \\ \arctan(\dfrac{y_{m_j} - y_{x_t}}{x_{m_j} - x_{x_t}}) - \theta_{x_t} \end{bmatrix} + \delta \tag{61}$$

where $[l_t(m_j), \beta_t(m_j)]$ are the distance and angle of the observation between the robot pose and the j-th landmark. δ is the observation noise.

From Fig. 9, it can be seen that some parts of the estimated path is far away from the GPS path in FastSLAM 2.0 while the proposed algorithm shows more accuracy in the path estimation. It is reasonable to conclude that the performance of proposed algorithm is more stable and accurate than FastSLAM 2.0. The maximum and average distance between the estimated path and the GPS path are reported in Table 2.

Table 2 Distance errors comparison between FastSLAM 2.0 and the proposed FastSLAM algorithm based on "Car Park Dataset"

Algorithm	Max distance	Average distance	Running time
Proposed algorithm	1.534 9	0.362 5	17.48
FastSLAM 2.0	2.657 5	0.658 2	14.74

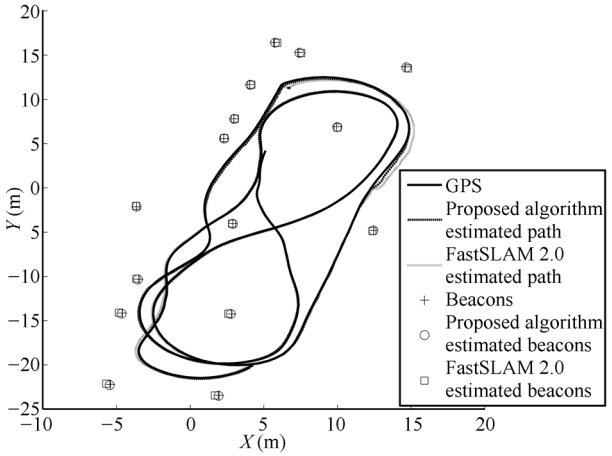


Fig. 9 Comparison between FastSLAM 2.0 and the proposed algorithm based on "Car Park Dataset"

5 Conclusions

This paper proposes an improved FastSLAM algorithm based on revised genetic resample and SR-UPF. In the proposed algorithm, 2 FastSLAM drawbacks are addressed. One problem is the linear approximations of the nonlinear functions, and the other is the particle depletion which is mainly caused by resampling process. Instead of approximating the motion nonlinear function by Taylor series expansion in FastSLAM 2.0, SR-UPF can decrease the linearization errors by passing sigma points through the nonlinear function. Moreover, SR-UPF has better numerical properties and guaranteed positive semi-definiteness by directly propagating the square root of the state covariance matrix. For the depletion problem, the revised genetic resampling algorithm is designed with attention to both the particle diversity and amending the particle degeneracy. By using double roulette wheels as the selection operation, not only the higher weight particles have more opportunity to breed the new generation, but many lower weight particles transfer their state to the new generation. So the particle diversity is improved. As fast MH developed for mutation operation can produce the particles converge to the target distribution, the new particle set is more accurate. The experiment results show that the proposed algorithm has more accurate estimations and less errors than FastSLAM 2.0.

Acknowledgments

Tai-Zhi Lv would like to thank Jiangsu Overseas Research & Training Program for University Prominent Young & Middle-aged Teachers and Presidents for financial support.

References

[1] C. L. Wang, T. M. Wang, J. H. Liang, Y. C. Zhang, Y. Zhou. Bearing-only visual SLAM for small unmanned aerial vehicles in GPS-denied environments. *International Journal of Automation and Computing*, vol. 10, no. 5, pp. 387–396, 2013.

[2] S. Thrun, W. Burgard, D. Fox. *Probabilistic Robotics*, Cambridge, USA: The MIT Press, 2005.

[3] M. Kaess, A. Ranganathan, F. Dellaert, iSAM: Incremental smoothing and mapping, *Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[4] S. Thrun, M. Montemerlo. The GraphSLAM Algorithm with Application to Large-Scale Mapping of Urban Structures, *The International Journal of Robotics Research*, vol. 25, no. 5–6, pp. 403–429, 2006.

[5] H. Wang, W. Mou, G. Seet, M. H. Li, M. W. S. Lau, D. W. Wang. Real-time visual odometry estimation based on principal direction detection on ceiling vision. *International Journal of Automation and Computing*, vol. 10, no. 5, pp. 397–404, 2013.

[6] M. Montemerlo. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association, Ph. D. dissertation, Carnegie Mellon University, USA, 2003.

[7] C. Gamallo, M. Mucientes, C. V. Regueiro. A FastSLAM-based Algorithm for Omni directional Cameras, *Journal of Physical Agents*, vol. 7, no. 1, pp. 12–21, 2012.

[8] S. M. Chen, J. F. Yuan, F. Zhang, H. J. Fang. Multi-robot FastSLAM Algorithm Based on Landmark Consistency Correction, *Mathematical Problems in Engineering*, vol. 2014, pp. 1–7, 2014.

[9] R. Martinez-Cantin, J. A. Castellanos. Unscented SLAM for large-scale outdoor environments. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Edmonton, Canada, pp. 3427–3432, 2005.

[10] D. B. Liu, G. R. Liu, M. H. Yu. An improved FastSLAM framework based on particle swarm optimization and unscented particle filter. *Journal of Computational Information Systems*, vol. 8, no. 7, pp. 2859–2866, 2012.

[11] K. Chanki, R. Sakthivel, W. K. Chung. Unscented FastSLAM: A robust algorithm for the simultaneous localization and mapping problem. In *Proceedings of International Conference on Robotics and Automation*, IEEE, Roma, Italy, pp. 2439–2445, 2007.

[12] J. H. Zhu, N. N. Zheng, Z. J. Yuan, Q. Zhang. A SLAM algorithm based on central difference particle filter. *Acta Automatica Sinica*, vol. 36, no. 2, pp. 249–257, 2010. (in Chinese)

[13] Y. Song, Y. D. Song, Q. L. Li. Robust iterated sigma point FastSLAM algorithm for mobile robot simultaneous localization and mapping. *Chinese Journal of Mechanical Engineering*, vol. 24, no. 4, pp. 693–700, 2011.

[14] T. Bailey, J. Nieto, E. Nebot. Consistency of the FastSLAM algorithm. In *Proceedings of International Conference on Robotics and Automation*, IEEE, Orlando, USA, pp. 424–429, 2006.

[15] C. Stachniss, D. Hahnel, W. Burgard. Exploration with active loop-closing for FastSLAM. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Sendai, Japan, pp. 1505–1510, 2004.

[16] J. S. Liu. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. Statistics and Computing, vol. 6, no. 2, pp. 113–119, 1996.

[17] G. Grisetti, C. Stachniss, W. Burgard. Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[18] N. Nosan, I. K. Kim, H. C. Lee, B. H. Lee. Analysis of resampling process for the particle depletion problem in fastslam. In *Proceedings of the 16th International Symposium on Robot and Human Interactive Communication*, IEEE, Jeju Island, Korea, pp. 200–205, 2007.

[19] X. Z. Li, S. M. Jia, W. Cui. On sample diversity in particle filter based robot SLAM. In *Proceedings of International Conference on Robotics and Biomimetics*, IEEE, Phuket, Thailand, pp. 1072–1077, 2011.

[20] Y. M. Xia, Y. M. Yang. An improved FastSLAM algorithm based on genetic algorithms. In *Proceedings of International Symposium, Communications in Computer and Information Science*, Springer, Guangzhou, China, pp. 296–302, 2011.

[21] H. Wang, Y. Yan, D. W. Wang. A GA based SLAM with range sensors only. In *Proceedings of the 11th International Conference on Control Automation Robotics and Vision*, IEEE, Singapore, pp. 1796–1802, 2010.

[22] P. Li, S. M. Song, X. L. Chen, G. R. Duan. Square root unscented Kalman filter incorporating Gaussian process regression. *Systems Engineering and Electronics*, vol. 32, no. 6, pp. 1281–1285, 2010. (in Chinese)

[23] R. D. Merw, E. A. Wan. The square-root unscented kalman filter for state and parameter-estimation. *Systems Engineering and Electronics*, vol. 32, no. 6, pp. 1281–1285, 2010.

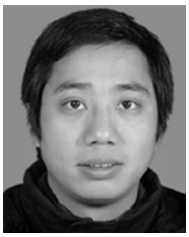
[24] P. Giordani, R. Kohn. Adaptive independent Metropolis-Hastings by fast estimation of mixtures of normals. *Journal of Computational and Graphical Statistics*, vol. 19, no. 2, pp. 243–259, 2010.

[25] J. E. Guivant, E. M. Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 242–257, 2001.

[26] T. Bailey. *SLAM simulations*, [Online], Available: http://www-personal.acfr.usyd.edu.au/tbailey/software/, March 10, 2011.

[27] Y. Bar-Shalom, X. R. Li, T. Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*, Hoboken, USA: John Wiley and Sons, 2004.

[28] E. Nebot, J, Guivant, J. Nieto. *ACFR, Experimental outdoor dataset*, [Online], Available: http://www.acfr.usyd.edu.au/home-pages/academic/enebot/dataset.htm, July 28, 2012.



Tai-Zhi Lv received the B. Sc. degree in computer science from Nanjing University, China in 2002, and the M. Sc. degree in computer software and theory from Nanjing University of Science and technology, China in 2006. He is currently a Ph. D. degree candidate of Nanjing University of Science and Technology, China.

His research interests include SLAM (Simultaneous localization and mapping) and robot navigation.

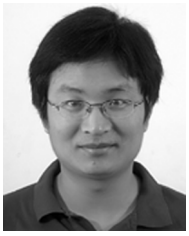
E-mail: lvtaizhi1@gmail.com (Corresponding author)

ORCID iD: 0000-0003-1994-9841

Chun-Xia Zhao received the B. Sc. degree in industrial automatization from Harbin Institute of Technology, China in 1985, the M. Sc. degree in pattern recognition and artificial intelligence control from Harbin Institute of Technology, China in 1988, and the Ph. D. degree in electromechanics and automatization from Harbin Institute of Technology, China in 1998. She is currently a professor at School of Computer Science and Technology, Nanjing University of Science and Technology, China. She has published above 100 refereed journal and conference papers.

Her research interests include underground robotics, computer vision and navigation.

E-mail: zhaochx@mail.njust.edu.cn



Hao-Feng Zhang received the B. Sc. in computer science and technology from Nanjing University of Science and Technology, China in 2003, and the Ph. D. degree in pattern recognition and intelligent systems from Nanjing University of Science and Technology, China in 2007. He is currently an associate professor at School of Computer Science and Technology, Nanjing University of Science and Technology, China.

His research interests include robotics, robot navigation and image process technology.

E-mail: zhanghf@njust.edu.cn