Robotica (2017) volume 35, pp. 1795–1815. © Cambridge University Press 2016
doi:10.1017/S0263574716000527

An improved FastSLAM 2.0 algorithm based on FC&ASD-PSO

Taizhi Lv†,* and Maoyan Feng‡

†*School of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing, Jiangsu 210094, P. R. China*
‡*School of Information Technology, Jiangsu Maritime Institute, Nanjing, Jiangsu 211170, P. R. China. E-mail: fmy999@163.com*

(Accepted July 4, 2016. First published online: August 9, 2016)

SUMMARY
FastSLAM 2.0 is a popular framework which uses a Rao-Blackwellized particle filter to solve the simultaneous localization and mapping problem. The sampling process is one of the most important phases in the FastSLAM 2.0 framework. Its estimation accuracy depends heavily on a correct prior knowledge about the control and observation noise statistics (the covariance matrices **Q** and **R**). Without the correct prior knowledge about these matrices, the estimation accuracy of the robot path and landmark positions may degrade seriously. However in many applications, the prior knowledge is unknown, or these noises are non-stationary. In this paper, these covariance matrices are supposed to be dynamic and denoted as $\mathbf{Q}_t$ and R_t. Since there are noises, time-adjacent observations are inconsistent with each other. This inconsistency can reflect the real value of the covariance matrices. By the inconsistency, an extra step is introduced to the FastSLAM 2.0 framework. This step makes $\mathbf{Q}_t$ and $\mathbf{R}_t$ match with their real value by using a particle swarm optimization method based on fractional calculus and alpha stable distribution (FC&ASD-PSO). Both simulation and experimental results show that the proposed algorithm improves the accuracy by the more accurate estimation on the noise covariance matrices.

KEYWORDS: Simultaneous localization and mapping, FastSLAM 2.0, Particle swarm optimization, Fractional calculus, Alpha stable distribution, Mobile robot

1. Introduction

Simultaneous localization and mapping (SLAM) is a process by which a mobile robot can build a map and locate itself at the same time in an unknown environment. SLAM is also known as concurrent mapping and localization, or 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.[1] SLAM has been applied to a number of different applications, stretching from search and rescue, over reconnaissance to commercial products.[1–3] However, the SLAM research still faces many challenging problems, such as large-scale and complex environments, reliable data association, non-linearity, and unknown priori knowledge.

From a probabilistic perspective, SLAM approaches can be divided into two groups: offline and online. Offline method optimizes the complete trajectory estimation and map after all data has been recorded. Batch algorithms such as smoothing and mapping (SAM)[6] and GraphSLAM[7] are offline methods. SAM involves not just the most current robot position, but the entire robot trajectory up to the current time.[6] GraphSLAM transforms the SLAM posterior into a graphical network, representing the log-likelihood of the data.[7] Online SLAM involves estimating the posterior over the momentary pose along with the map. The two key computational solutions to the online SLAM problem are extended Kalman filter SLAM (EKF-SLAM) and FastSLAM.[3,4] EKF-SLAM employs an extended Kalman filter (EKF) to represent the joint state space of a robot

* Corresponding author. E-mail: lvtaizhi@163.com

pose and all identified landmarks. Since EKF-SLAM has two major well-known shortcomings: quadratic computational complexity and sensitivity to failures in data association, it is difficult to be applied in real and large environments.[3,4,25] 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.[3−5] Some newer methods like Marginal-SLAM,[8] GMapping,[9] cubature Kalman filter SLAM (CKF-SLAM)[10] and unscented Kalman filter SLAM (UKF-SLAM)[4,11] are also recommendable to solve the online SLAM problem.

Because FastSLAM has two advantages: multi-hypothesis data association and the ability to cope with nonlinear and non-Gaussian robot motion models, it is believed to be an effective mean to solve the SLAM problem by a lot of scholars.[1,3−5] There are many successful implementations of FastSLAM to solve different SLAM applications. In ref. [12], FastSLAM was chosen to solve the visual SLAM application. Chen *et al.*[13] extended FastSLAM from the single-robot SLAM to the multi-robot case. FastSLAM, however, has some drawbacks as well. In ref. [14], it was noted that FastSLAM will degenerate over time, regardless of the number of particles and the density of landmarks within an environment. Resampling process aims for amending the particle degeneracy, but it causes particles to lose their diversity, the so called particle depletion. When particles lose their diversity, the estimation soon becomes optimistic, and it tends to underestimate its own uncertainty. Liu[15] introduced an 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. In ref. [16], some well-known resampling algorithms were analyzed by computer simulations. Although all resampling algorithms could not resolve the particle depletion problem, a suitable resampling method can improve the FastSLAM performance. Some biological evolution algorithms have been newly proposed to keep particle diversity as long as possible.[17,18] On the other hand, FastSLAM linearizes the motion model in the same manner as EKF-SLAM. Inaccurate approximation of the nonlinear function leads to filter divergence.[3,19] Chanki Kim[19] proposed Unscented FastSLAM to overcome this important limitation of the FastSLAM framework. This method has also been used in some improved FastSLAM algorithms.[20,21] Recently, the central difference particle filter and robust iterated sigma point methods were introduced, and they were both similar to Unscented FastSLAM.[22−24]

These improved FastSLAM algorithms mentioned above were based on the assumption that the control and observation noise statistics would be completely known and correct. Because of the complexity of the real world, this assumption is hard to be tenable. Incorrect priori knowledge about the control and observation noise matrices would seriously degrade the accuracy of the FastSLAM algorithm.[25−27] Havangi[25] proposed an adaptive Neuro-Fuzzy method to dynamically adjust the noise statistics. This method adjusts only the observation noise statistic, and this adjustment is influenced by the cumulative errors tremendously. Particle swarm optimization (PSO) is a very useful algorithm to find approximate solutions to extremely difficult or impossible numeric maximization and minimization problems.[28] There were some attempts which used the classical PSO to solve the SLAM problem, and these attempts were focused on the sampling process. Two weaknesses restrict its applications in SLAM problems. One is slow convergence, and the other is to get trapped into local optima easily.[29,30]

To solve the incorrect or unknown priori knowledge problem, this paper supposes the matrices $\mathbf{Q}$ and $\mathbf{R}$ to be dynamic. The time subscript is introduced to the covariance matrices, and they are rewritten as $\mathbf{Q}_t$ and $\mathbf{R}_t$. To make $\mathbf{Q}_t$ and $\mathbf{R}_t$ match with their real value, an extra adjusting step is introduced to the FastSLAM 2.0 framework. To decrease the cumulative errors, the proposed algorithm uses the inconsistency between the time-adjacent observations to adjust these matrices rather than the difference between the predicated observations and the observations. According to the inconsistency, a complex and multi-modal fitness function is defined. Comparing the classical PSO algorithm, fractional calculus and alpha stable distribution particle swarm optimization (FC& ASD-PSO) features better search efficiency and accuracy.

The main contributions of this paper are stated as follows:

(1) The control and observation covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ are supposed to be dynamic, and denoted as $\mathbf{Q}_t$ and $\mathbf{R}_t$. An extra $\mathbf{Q}_t$ and $\mathbf{R}_t$ adjusting step is introduced to the FastSLAM 2.0 framework.

https://doi.org/10.1017/S0263574716000527 Published online by Cambridge University Press

(2) The inconsistency between time-adjacent observations is used to adjust $\mathbf{Q}_t$ and $\mathbf{R}_t$. Without cumulative errors, the inconsistency can reflect the real values of $\mathbf{Q}_t$ and $\mathbf{R}_t$ more accurately.
(3) A novel PSO algorithm based on FC&ASD is proposed. With the faster convergence and better performance, more suitable values of $\mathbf{Q}_t$ and $\mathbf{R}_t$ are obtained quickly.

2. Background

2.1. Online SLAM problem

In SLAM problems, a robot does not have access to its own poses and a map of the environment. Instead, all which can be acquired are measurements and controls.[1] To describe the SLAM problem, the robot pose at time t is denoted as x_t, and the entire map is written as Θ. The map consists of N landmarks which are written as $\{m_1, m_2, \ldots, m_N\}$. The control and the observation at time t are denoted as u_t and z_t, respectively. The standard motion and observation models as nonlinear functions with independent Gaussian noise are as follows:

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

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

where g and h are nonlinear functions, and ε_t and δ_t are the Gaussian noise variables with the covariance matrices $\mathbf{Q}_t$ and $\mathbf{R}_t$, 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

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

where $u_{1:t} = [u_1, u_2, \ldots u_t]$ represents the set of all controls executed by the robot, and $z_{1:t} = [z_1, z_2, \ldots z_t]$ represents the set of all observations collected by the robot.

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

$$\underbrace{p(x_t, \Theta|z_{1:t}, u_{1:t})}_{\text{Posterior distribution at } t} \propto \eta \ \underbrace{p(z_t|x_t, \Theta)}_{\text{Observation model}} \int \underbrace{p(x_t|x_{t-1}, u_t)}_{\text{Motion model}} \underbrace{p(x_{t-1}, \Theta|z_{1:t-1}, u_{1:t-1}, x_0)}_{\text{Posterior distribution at } t-1} dx_{t-1} \tag{4}$$

In general, the integral in Eq. (4) cannot be evaluated in closed form. The extend Kalman filter and FastSLAM are simply approximations of the general Bayes filter.[5]

2.2. FastSLAM 2.0 algorithm

The posterior distribution in Eq. (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[5]

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

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

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

https://doi.org/10.1017/S0263574716000527 Published online by Cambridge University Press

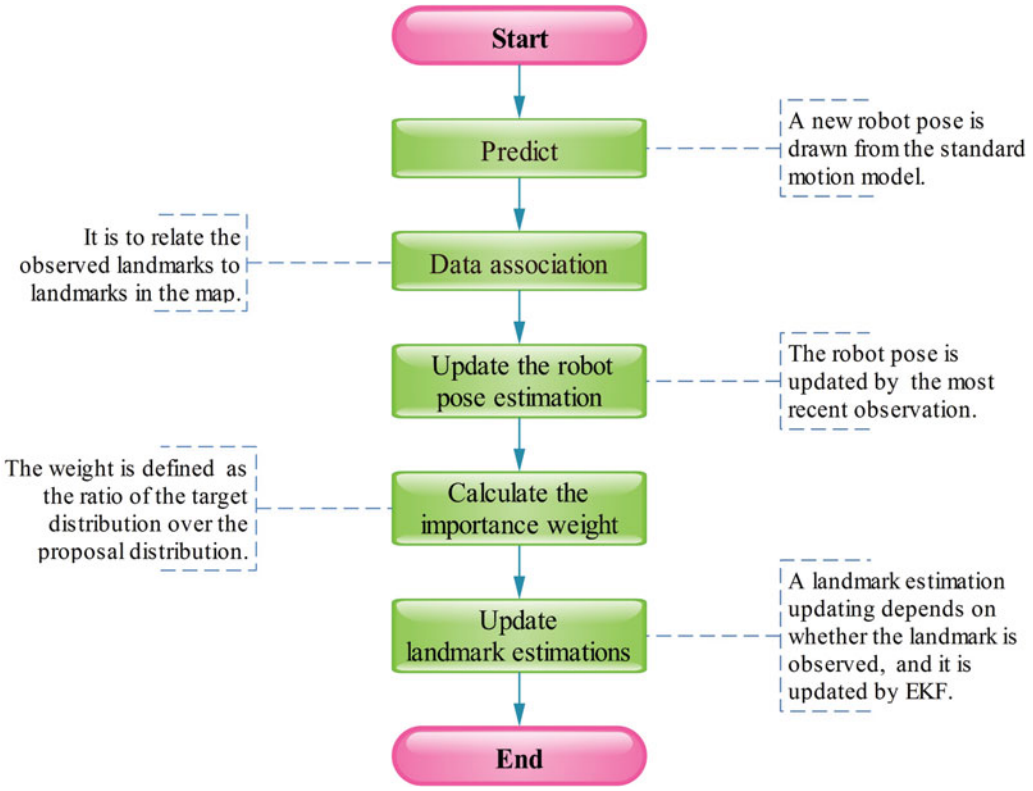

Fig. 1. The flow of each particle updating in FastSLAM 2.0.

following equation:

$$\mathrm{X}_t^{[i]} = \left\langle \left(x_t^{[i]}, \omega_t^{[i]}\right), \left(\mu_t^{[i,1]}, \sum_t^{[i,1]}\right), \left(\mu_t^{[i,2]}, \sum_t^{[i,2]}\right), \ldots, \left(\mu_t^{[i,j]}, \sum_t^{[i,j]}\right), \ldots, \left(\mu_t^{[i,N]}, \sum_t^{[i,N]}\right) \right\rangle \quad (6)$$

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

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

(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[5]

$$x_t^{[i]} \sim \mathrm{p}\left(x_t | x_{t-1}^{[i]}, u_t, z_t\right) \quad (7)$$

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

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

where $\hat{x}_t^{[i]}$ is the predicted pose of the ith particle from the motion model at time t. By incorporating the current observation into the proposal distribution, FastSLAM 2.0 get better match in the posterior and it is superior to FastSLAM 1.0 in nearly all respects.[33]

https://doi.org/10.1017/S0263574716000527 Published online by Cambridge University Press

(2) Data association

Data association is the process of relating landmarks observed in the environment to landmarks in the map. This paper adopts the probabilistic multi-hypothesis tracker (PMHT) method for data association.[31]

(3) Update the robot pose estimation

In this step, the robot pose is updated by the most recent observation

$$\hat{z}_t^{[i]} = \mathrm{h}\left(\hat{x}_t^{[i]}, \Theta\right) \tag{9}$$

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

where $\hat{z}_t^{[i]}$ is the predicted observation, and β is calculated from the landmark estimation covariance matrices, $\mathbf{Q}_t$, $\mathbf{R}_t$, 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

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

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

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

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

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

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[32]

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

The proposal distribution in FastSLAM 2.0 is as follows:[33]

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

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

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

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. A landmark estimation updating depends on whether the landmark is observed at time

https://doi.org/10.1017/S0263574716000527 Published online by Cambridge University Press

t.[5,32,33] If a landmark is newly observed, its mean and covariance are initialized as follows:

$$\mu_t^{[i,N+1]} = \mathrm{h}^{-1}\left(z_t, x_t^{[i]}\right) \sum_t^{[i,N+1]} = \mathbf{H}_{rz}\mathbf{R}_t\mathbf{H}_{rz}^{\mathrm{T}} \tag{17}$$

where the matrix $\mathbf{H}_{rz}$ is the Jacobian of h^{-1}.

If the jth landmark in the map is not observed, the estimation remains unchanged:

$$\mu_t^{[i,j]} = \mu_{t-1}^{[i,j]} \sum_t^{[i,j]} = \sum_{t-1}^{[i,j]} \tag{18}$$

If the jth landmark in the map is observed, the updating is specified through the following equations:

$$\mu_t^{[i,N+1]} = \mu_{t-1}^{[i,j]} + \mathbf{K}_{\mathrm{t}}\left(z_{\mathrm{t}} - \hat{z}_t^{[i]}\right) \sum_t^{[i,j]} = (\mathbf{I} - \mathbf{K}_{\mathrm{t}}\mathbf{H}_z) \sum_{t-1}^{[i,j]} \tag{19}$$

where $\mathbf{K}_t$ is the Kalman gain coefficient, and the matrix $\mathbf{H}_z$ 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.[15,16] The number is defined as

$$N_{\mathrm{eff}} = \frac{1}{\sum_{k=1}^{M} \left(\tilde{\omega}_t^k\right)^2} \tag{20}$$

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

$$\omega_t^{[j]} = \frac{1}{M} \tag{21}$$

2.3. Particle swarm optimization

PSO was published by Kennedy and Eberhart in 1995.[34] This algorithm imitates human (or insects) social behavior. For its efficiency in solving challenging optimization problems with minor implementation effort, it has been applied to a plethora of fields such as social biomedicine, finance, engineering design, automation, robot, signal process, computer graphic, and pattern recognition.[28–30]

In the PSO algorithm, each particle represents one potential solution. Each particle has memory storing its location, speed, and personal best solution. During the evolutionary process, the location and speed of each particle are updated iteratively by the following equations:[34]

$$\mathbf{V}_i(k+1) = \underbrace{\omega\mathbf{V}_i(k)}_{\text{Momentum component}} + \underbrace{c_1 r_1(p_{ib}(k) - \mathbf{X}_i(k))}_{\text{Cognitive component}} + \underbrace{c_2 r_2(p_{gb}(k) - \mathbf{X}_i(k))}_{\text{Social component}} \tag{22}$$

$$\mathbf{X}_i(k) = \mathbf{X}_i(k) + \mathbf{V}_i(k+1) \tag{23}$$

where $\mathbf{X}_i(k)$ is denoted as the ith particle location at the kth iteration. This particle will fly to the next location $\mathbf{X}_i(k+1)$ by the velocity $\mathbf{V}_i(k+1)$. $\mathbf{V}_i(k+1)$ is determined by three components, namely, momentum, cognitive, and social. The momentum component simulates the inertial behavior of the bird flying, and ω is the inertia weight that exerts the effect of the old velocity on the new one.[35] The cognitive component models the memory of the bird about its visited best position so far, and the social component attracts birds to fly towards the best position found in their neighborhood (interaction inside the swarm).[36] c_1 and c_2 are called the cognitive and social acceleration coefficients, respectively. r_1 and r_2 are random numbers. $p_{ib}(k)$ and $p_{gb}(k)$ are the best position of the ith particle and the global best position of the entire swarm, respectively.

https://doi.org/10.1017/S0263574716000527 Published online by Cambridge University Press

2.4. Fractional calculus and alpha stable distribution

There are several definitions of fractional calculus. Grünwald–Letnikov is one of the most popular definitions[37,38]

$$\alpha D_t^a f(t) = \lim_{h\to 0} \frac{1}{h^\alpha} \sum_{k=0}^{[\frac{t-a}{h}]} (-1)^k \binom{\alpha}{k} f(t - k \times h) \tag{24}$$

$$\binom{\alpha}{k} = \frac{\Gamma(\alpha + 1)}{\Gamma(k + 1)\Gamma(\alpha - k + 1)} \tag{25}$$

where α is the fractional order, and h is the step size. Often, in discrete time implementations expression (23) is approximated by

$$\alpha D_t^a f(t) = \frac{1}{T^\alpha} \sum_{k=0}^{[\frac{t-a}{T}]} (-1)^k \binom{\alpha}{k} f(t - k \times T) \tag{26}$$

Alpha stable distribution is parameterized by the two shape parameters α and β, the location parameter μ, and the scale parameter σ.[40] A random variable x is called stable if its characteristic function can be written as

$$f(x; \alpha, \beta, \sigma, \mu) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \exp[it\mu - |\sigma t|^\alpha (1 - i\beta \mathrm{sgn}(t)\Phi)] e^{-itx} \mathrm{d}t \tag{27}$$

where $\mathrm{sgn}(t)$ is just the sign of t, and Φ is given by

$$\Phi = \begin{cases} \tan(\frac{\pi\alpha}{2})\alpha \neq 1 \\ -(\frac{2}{\pi}) \log |t| \alpha = 1 \end{cases} \tag{28}$$

3. An Improved FastSLAM 2.0 Algorithm Based on FC&ASD-PSO

As already mentioned above, a correct priori knowledge about the control and observation noise statistics (covariance matrices $\mathbf{Q}_t$ and $\mathbf{R}_t$) is assumed in the FastSLAM 2.0 algorithm. However, it is very difficult to get the real values of $\mathbf{Q}_t$ and $\mathbf{R}_t$. Many sensors are sensitive to complex and dynamic environments. For example, the road surface, temperature, tire status, crawler belt tightening and driving speed can all affect the sensor sensitivity in an inertial navigation system (INS).[41] Ultrasound sensors are very sensitive to the angle of an object's surface relative to sensors.[42] In the FastSLAM 2.0 algorithm, the estimations of the robot pose and landmark positions are updated by the difference between the observation z_t and the predicted observation $\hat{z}_t$. This updating depends on $\mathbf{Q}_t$ and $\mathbf{R}_t$. If $\mathbf{R}_t$ is more than $\mathbf{Q}_t$, it means that we trust the control data more than the observation data, and the updated pose would be closer to the predicted pose. Otherwise it means that we trust the observation data more than the control data, and the updated pose would be closer to the observation. An incorrect priori knowledge about $\mathbf{Q}_t$ and $\mathbf{R}_t$ would cause the estimation to be the result of an error trend to the control data or the observation data.

Figure 2 shows the comparison between the three estimations of the robot pose and map. These estimations use the same FastSLAM 2.0 algorithm, but they are based on the different priori knowledge. The blue estimation is based on the correct priori knowledge about $\mathbf{Q}$ and $\mathbf{R}$. $\mathbf{R}$ is amplified mistakenly by a factor of 3 on the red estimation. $\mathbf{Q}$ is amplified mistakenly by a factor of 3 on the green estimation. The black arrow and plus signs show the real robot pose and landmark positions. Since the sensed velocity and angle are more than real value in this experiment, the predicted robot pose is the upper-left of the real pose. Because $\mathbf{R}$ is amplified in the red estimation, it leads this estimation over-trust the control data. The red estimated robot pose is closed to the predicted pose which is calculated from the motion model. All red landmark estimations are the upper-left of their real positions. Because $\mathbf{Q}$ is amplified mistakenly in the green estimation, it leads this estimation

https://doi.org/10.1017/S0263574716000527 Published online by Cambridge University Press

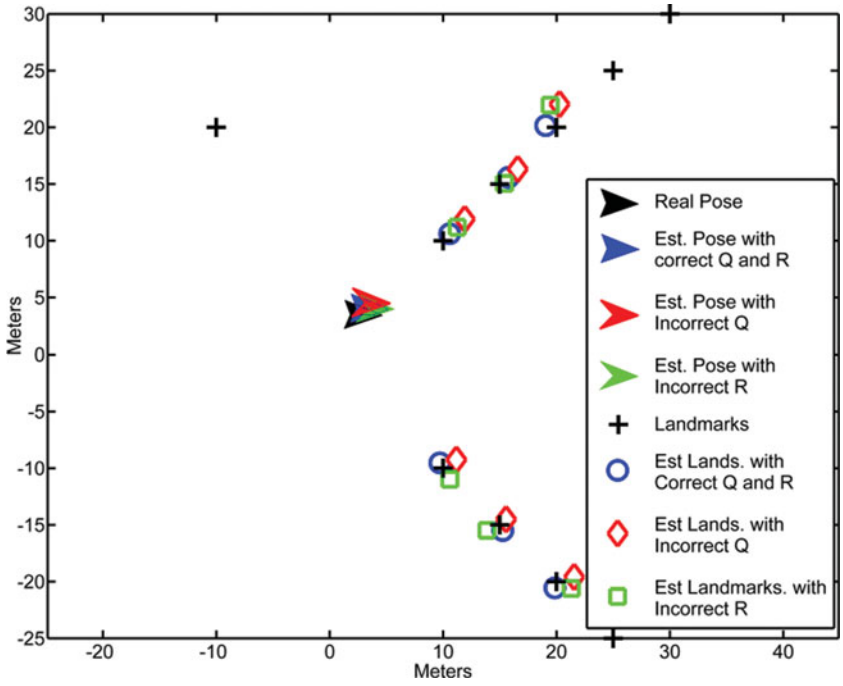

Fig. 2. The comparison between the three estimations based on different priori knowledge about **Q** and **R**.

over-trust the observation data. This amplification leads the robot pose estimation and landmark estimations to have an error trend to the observation data. It can be seen that the algorithm based on the correct priori knowledge gets more accuracy estimations comparing the same algorithm based on the incorrect priori knowledge.

To solve the incorrect priori knowledge problem, an adjusting step is introduced to the FastSLAM 2.0 framework. The suitable fitness function and time complexity are important for this step. Because the inconsistency between the observation z_{t-1} and z_t can reflect the real values of $\mathbf{Q}_t$ and $\mathbf{R}_t$, the adjusting step uses it as the fitness function. To improve the compute efficiency and accuracy, this step uses FC&ASD-PSO rather than the classical PSO algorithm to search the approximate solutions.

3.1. Fitness function

Although the difference between the observation z_t and predicted observation $\hat{z}_t$ can reflect the real values of $\mathbf{Q}_t$ and $\mathbf{R}_t$ to some extent, this reflection is influenced by the cumulative errors tremendously. To avoid the cumulative errors influence, the time-adjacent observations z_{t-1} and z_t are compared. In the adjusting step, it is assumed that the robot is at the original point at time $t-1$, and the pose is denoted as $\bar{x}_{t-1}$. The map, which uses the same coordinate system as observation, is denoted as $\bar{\Theta}$. Since noises exist in the controls and observations, there is an inconsistency between time-adjacent observations. According to this inconsistency, FC&ASD-PSO method searches the optimal $\tilde{\mathbf{Q}}_t$, $\tilde{\mathbf{R}}_t$, and $\tilde{x}_t$ which are used as the parameters of the fitness function. The fitness function is the product of the pose estimation and the observation estimation conditioned on the robot pose and the map

$$\text{argmax}|_{\tilde{\mathbf{Q}}_t,\tilde{\mathbf{R}}_t,\tilde{x}_t}\ \text{p}(z_t|\bar{x}_t,\bar{\Theta}_t)\text{p}(\bar{x}_t|u_t,\bar{x}_{t-1}) \tag{29}$$

where $\bar{x}_t$ is the robot pose with the same coordinate system as u_t. A special particle $X^{[\text{ex}]}$, which is used to adjust $\mathbf{Q}_t$ and $\mathbf{R}_t$, consists of $\bar{x}_t$ and $\bar{\Theta}$

$$\text{p}(\bar{x}_t|u_t,\bar{x}_{t-1}) \sim N\left(\bar{x}_t;\text{f}(u_t),\mathbf{F}_u\mathbf{Q}_t\mathbf{F}_u^{\text{T}}\right) \tag{30}$$

https://doi.org/10.1017/S0263574716000527 Published online by Cambridge University Press

where f is the transformation of the function g in Eq. (1) for the different coordinate system, and the matrix $\mathbf{F}_u$ is the Jacobian of the function f

$$\mathrm{p}(\bar{\Theta}) \sim N(\bar{\Theta}; z_{t-1}, \mathbf{R}_t) \tag{31}$$

$$\mathrm{p}(z_t|\bar{x}_t, \bar{\Theta}) \sim N(z_t; \mathrm{s}(\bar{x}_t, \bar{\Theta}), \mathbf{R}_t) \tag{32}$$

where s is the transformation of the function h in Eq. (2).

$\mathrm{p}(z_t|\bar{x}_t, \bar{\Theta}_t)\mathrm{p}(\bar{x}_t|u_t, \bar{x}_{t-1})$ can be written as a factored form

$$\mathrm{p}(z_t|\bar{x}_t, \bar{\Theta})\mathrm{p}(\bar{x}_t|u_t, \bar{x}_{t-1}) = \prod_{j=1}^{N} \mathrm{p}(z_t|\bar{x}_t, \bar{m}_j)\mathrm{p}(\bar{x}_t|u_t, \bar{x}_{t-1}) \tag{33}$$

The function s will be replaced by a linear approximation, and the approximation is obtained through a first-order Taylor expansion

$$\hat{x}_t = \mathrm{f}(u_t) \tag{34}$$

$$\hat{z}_t = \mathrm{s}(\hat{x}_t, z_{t-1}) \tag{35}$$

$$\mathbf{S}_m = \nabla_{\bar{m}_j} \mathrm{s}(x_t, \bar{m}_j)|\bar{x}_t = \hat{x}_t, \bar{m}_j = z_{t-1}(\bar{m}_j) \tag{36}$$

$$\mathbf{S}_x = \nabla_{\bar{x}_t} \mathrm{s}(x_t, \bar{m}_j)|\bar{x}_t = \hat{x}_t, \bar{m}_j = z_{t-1}(\bar{m}_j) \tag{37}$$

$$\mathrm{s}(\bar{x}_t, \bar{m}_j) \approx \hat{z}_t + \mathbf{S}_m(\bar{m}_j - z_{t-1}(\bar{m}_j)) + \mathbf{S}_x(\bar{x}_t - \hat{x}_t) \tag{38}$$

where $\hat{x}_t$ and $\hat{z}_t$ can be thought of as the predicted robot pose and the predicted observation at time t. The observation at time $t-1$ on the jth landmark $z_{t-1}(\bar{m}_j)$ is used to estimate this landmark position. The matrices $\mathbf{S}_m$ and $\mathbf{S}_x$ are the Jacobian of s. That is, they are the derivatives of s with respect to $\bar{m}_j$ and $\bar{x}_t$, respectively. An approximate form is determined by the linear approximation

$$\mathrm{p}(z_t|\bar{x}_t, \bar{\Theta}) \sim N(z_t; \hat{z}_t, \underbrace{\mathbf{R}_t + \mathbf{S}_m\mathbf{R}_t\mathbf{S}_m^{\mathrm{T}} + \mathbf{S}_x\mathbf{F}_u\mathbf{Q}_t\mathbf{F}_u^{\mathrm{T}}\mathbf{S}_x^{\mathrm{T}}}_{\mathbf{Z}_t}) \tag{39}$$

For brevity, the covariance of this Gaussian is written as $\mathbf{Z}_t$. The fitness function y_t is written as the following form:

$$y_t = \mathrm{p}(z_t|\bar{x}_t, \bar{\Theta})\mathrm{p}(\bar{x}_t|u_t, \bar{x}_{t-1}) = \frac{1}{\sqrt{2\pi\mathbf{Z}_t}} e^{-\frac{(z_t-\hat{z}_t)^2}{2\mathbf{Z}_t}} \frac{1}{\sqrt{2\pi\mathbf{F}_u\mathbf{Q}_t\mathbf{F}_u^{\mathrm{T}}}} e^{-\frac{(\bar{x}_t-\hat{x})^2}{2\mathbf{F}_u\mathbf{Q}_t\mathbf{F}_u^{\mathrm{T}}}} \tag{40}$$

3.2. FC&ASD-PSO implementation

3.2.1. Fractional-order velocity. Fractional calculus is used to update the velocity in this proposed algorithm. With memory and hereditary properties, this method has faster speed and more accurate solution than the classical PSO algorithm.

Let the inertia weight ω is 1, and Eq. (22) can be rewritten as

$$\mathbf{V}_i(k+1) - \mathbf{V}_i(k) = \phi_1 + \phi_2 \tag{41}$$

where $\phi_1 = c_1 r_1(p_{ib}(k) - \mathbf{X}_i(k))$ and $\phi_2 = c_2 r_2(p_{gb}(k) - \mathbf{X}_i(k))$. The left side $\mathbf{V}_i(k+1) - \mathbf{V}_i(k)$ is the discrete version of the derivative of order α =1 (assuming $T = 1$), leading to the following expression:

$$\alpha D_{k+1}^{a}\mathbf{V}(k+1) = \phi_1 + \phi_2 \tag{42}$$

https://doi.org/10.1017/S0263574716000527 Published online by Cambridge University Press

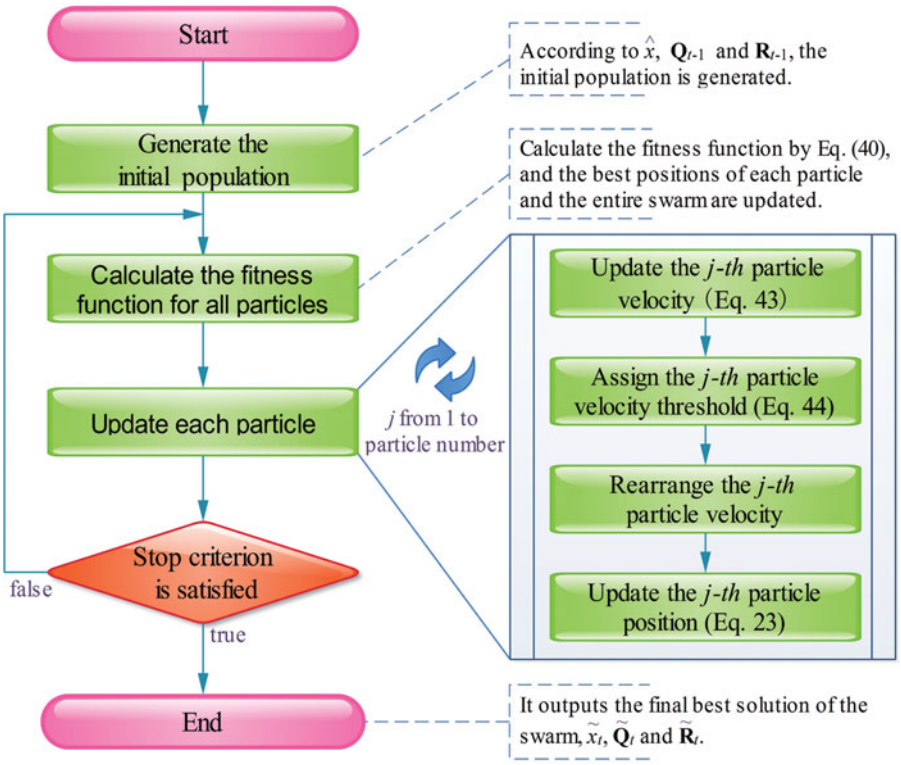

Fig. 3. The flow of FC&ASD-PSO.

Equation (22) can be written as Eq. (43) considering the first $a = k-3$ terms of differential derivative given by Eq. (26)

$$\mathbf{V}_i(k+1) = \alpha\mathbf{V}_i(k) + \frac{1}{2}\alpha\mathbf{V}_i(k-1) + \frac{1}{6}\alpha(1-\alpha)\mathbf{V}_i(k-2) + \frac{1}{24}\alpha(1-\alpha)(2-\alpha)\mathbf{V}_i(k-3) + \phi_1 + \phi_2 \tag{43}$$

where α varies in [0.2, 0.7] by the iteration number.

3.2.2. Velocity threshold based on alpha stable distribution. Velocity threshold plays an important role in the PSO algorithm. A dynamic velocity threshold can improve the PSO performance especially for multi-modal functions.[39,43] The fitness function y_t defined in Eq. (40) is a multi-modal function in whose solution space there are many local optima. In this paper, a stochastic velocity threshold automation strategy is proposed by incorporated the alpha stable distribution. This method has a large capability of the global exploration and big chances to jump the local optima.[43]

Let $v^j_{\max}(k)$ denoted as the jth particle velocity threshold at the iteration k. It is generated as follows:

$$v^j_{\max}(k) = |rand| \tag{44}$$

where $rand$ is a random number with the alpha stable probability distribution.

3.2.3. FC&ASD-PSO implementation. $[\tilde{x}_t, \tilde{\mathbf{Q}}_t, \tilde{\mathbf{R}}_t]$ is the output of the FC&ASD-PSO search, and they will be used in the adjusting step. The detail of FC&ASD-PSO is shown in Fig. 3.

https://doi.org/10.1017/S0263574716000527 Published online by Cambridge University Press

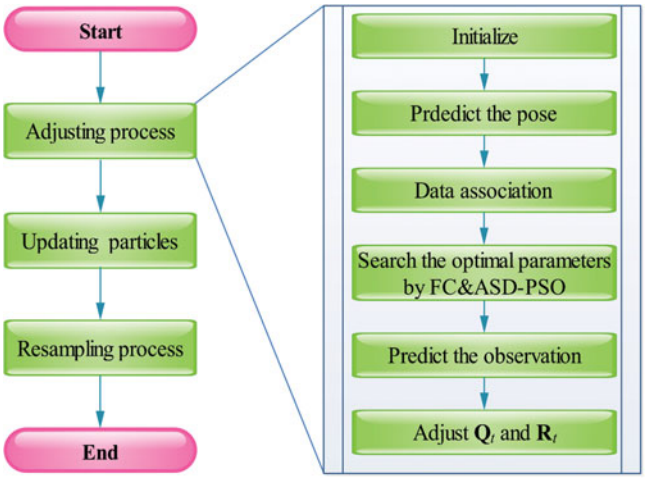

Fig. 4. The flow of the proposed algorithm.

3.3. An improved FastSLAM 2.0 algorithm based on FC&ASD-PSO

The proposed algorithm introduces an adjusting step before the updating particles process to the FastSLAM 2.0 framework. The detail of this algorithm is shown in Fig. 4. The updating particles process (as shown in Fig. 1) and resampling process are the same as FastSLAM 2.0.

In the adjusting process, $X_t^{[ex]}$ (as defined in Section 3.1) is supposed to be a special particle used for adjusting $\mathbf{Q}_t$ and $\mathbf{R}_t$. The adjusting process is as follows:

(1) Initialize

If there are a few observed landmarks, it is difficult to determine the respective impact of the control noise and the observation noise to the inconsistency. It is unnecessary to continue the adjusting process when the number of observed landmarks is less than 3.

The robot pose at time $t-1$ is supposed to be at the original point, and the map can be represented by the last observation z_{t-1}

$$X_t^{[ex]} = [\bar{x}_{t-1}, \bar{\Theta}] = [(0, 0, 0), z_{t-1}] \tag{45}$$

(2) Predict the pose

$$\hat{x}_t = \mathrm{f}(u_t) \tag{46}$$

(3) Data association

PMHT method is also used to implement the data association for this particle. The data association would be more accurate, for that there are no cumulative errors.

(4) Search the optimal parameters by the FC&ASD-PSO method

$$[\tilde{x}_t, \tilde{\mathbf{Q}}_t, \tilde{\mathbf{R}}_t] = \mathrm{PSO}(\hat{x}_t, \mathbf{Q}_{t-1}, \mathbf{R}_{t-1}, z_{t-1}, z_t) \tag{47}$$

(5) Predict the observation

$$\hat{z}_t = \mathrm{s}(\tilde{x}_t, \bar{\Theta}) \tag{48}$$

(6) Adjust Q_t and R_t

$$Dx_t = \frac{\tilde{x}_t - \hat{x}_t}{\mathbf{F}_u \mathbf{Q}_{t-1} \mathbf{F}_u^{\mathrm{T}}} \tag{49}$$

$$Dz_t(m_j) = \frac{z_t(\bar{m}_j) - \hat{z}_t(\bar{m}_j)}{\mathbf{S}_m \mathbf{R}_{t-1} \mathbf{S}_m^T} \quad j = 1, 2, \ldots, M_o \tag{50}$$

https://doi.org/10.1017/S0263574716000527 Published online by Cambridge University Press

where $z_t(\bar{m}_j)$ is the observation of the jth landmark, and $\hat{z}_t(\bar{m}_j)$ is the predicted observation of the jth landmark. M_o is the number of observed landmarks.

Let $\tilde{x}_t - \hat{x}_t$ represent the pose error, and $Dx_t \sim N(0, 1)$ obeys the normal distribution. In order to avoid frequent adjustment, $\mathbf{Q}_t$ will be adjusted when the relative small probability event happens. If $|Dx_t|$ is more than 1.4 or less than 0.2, $\mathbf{Q}_t$ is calculated as

$$\mathbf{Q}_t = \mathbf{Q}_{t-1} + \omega_q(\tilde{\mathbf{Q}}_t - \mathbf{Q}_{t-1}) \tag{51}$$

Otherwise $\mathbf{Q}_t$ remains unchanged

$$\mathbf{Q}_t = \mathbf{Q}_{t-1} \tag{52}$$

where ω_q is an adjustment weight and varied by the number of the observed landmarks.

Let $z_t(\bar{m}_j) - \hat{z}_t(\bar{m}_j)$ represent the observation noise of the jth landmark. Mn is denoted as the number of $|Dz_t(\bar{m}_j)|$ greater than 1.4, and Ln is denoted as the number of $|Dz_t(\bar{m}_j)|$ less than 0.2

$$Mn_p = \frac{Mn}{M_o}\ Ln_p = \frac{Ln}{M_o} \tag{53}$$

When Mn_p or Ln_p is more than a given threshold $N_{\max}$, $\mathbf{R}_t$ is adjusted by the following equation:

$$\mathbf{R}_t = \mathbf{R}_{t-1} + \omega_r(\tilde{R}_t - \mathbf{R}_{t-1}) \tag{54}$$

Otherwise, $\mathbf{R}_t$ remains unchanged

$$\mathbf{R}_t = \mathbf{R}_{t-1} \tag{55}$$

where ω_r is an adjustment weight. While M_o decreases, the threshold $N_{\max}$ increases, and ω_q and ω_r decrease, and *vice versa*.

4. Experimental Analysis

4.1. Simulation experiments

4.1.1. Simulation models. In the simulation environment, the SLAM state is described by the robot pose (position and heading) and landmark positions in the Cartesian coordinate system. The state at time t is represented by a joint state-vector $\mathbf{X}_t$

$$\mathbf{X}_t = [x_t, \Theta] = [(x_{x_t}, y_{x_t}, \theta_{x_t}), (x_{m_1}, y_{m_1}), \ldots, (x_{m_j}, y_{m_j}), \ldots, (x_{m_N}, y_{m_N})] \tag{56}$$

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

$$x_t = \mathrm{g}(x_{t-1}, u_t) = \begin{bmatrix} x_{x_{t-1}} + \Delta T V_t \cos(\theta_{x_{t-1}} + \alpha_t) \\ y_{x_{t-1}} + \Delta T V_t \sin(\theta_{x_{t-1}} + \alpha_t) \\ \theta_{x_{t-1}} + \Delta T V_t \sin(\alpha_t)/\mathrm{L} \end{bmatrix} + \varepsilon \tag{57}$$

$$z_t(m_j) = \mathrm{h}(x_t, m_j) = \begin{bmatrix} l_t(m_j) \\ \beta_t(m_j) \end{bmatrix} = \begin{bmatrix} \sqrt{(x_{m_j} - x_{x_t})^2 + (y_{m_j} - y_{x_t})^2} \\ \arctan\left(\frac{y_{m_j} - y_{x_t}}{x_{m_j} - x_{x_{t_t}}}\right) - \theta_{x_t} \end{bmatrix} + \delta \tag{58}$$

The control and the observation are denoted as ut and z_t, respectively. u_t includes the velocity V_t and steering angle α_t. $z_t(m_j)$ represents the range-bearing observation from the robot to the jth 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 special particle $\mathbf{X}_t^{[ex]}$ uses the coordinate system of the control and observation. Since the control and observation data is determined by the distance and angle in the simulation environment,

https://doi.org/10.1017/S0263574716000527 Published online by Cambridge University Press

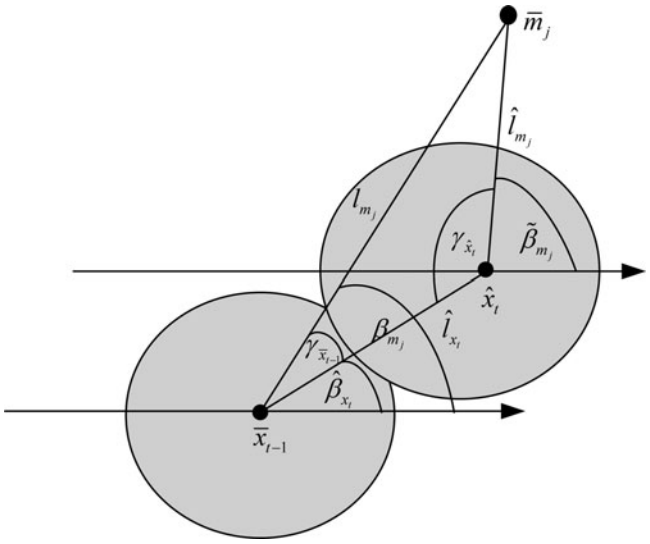

Fig. 5. The particle $\mathbf{X}_t^{[ex]}$ observation model based on the polar coordinate system. $\gamma_{\bar{x}_{t-1}}$ and $\gamma_{\hat{x}_t}$ are the angles at the points $\bar{x}_{t-1}$ and $\hat{x}_t$. l_{m_j}, $\hat{l}_{x_t}$, and $\hat{l}_{m_j}$ are the lengths of edges.

the estimation of the robot pose and landmark positions for the special particle is modeled in the polar coordinate system

$$\mathbf{X}_t^{[ex]} = [\bar{x}_t, \bar{\Theta}_t] = [(l_{x_t}, \beta_{x_t}, \theta_{x_t}), (l_{m_1}, \beta_{m_1}), \ \ldots, (l_{m_j}, \beta_{m_j}), \ \ldots, (l_{m_N}, \beta_{m_N})] \tag{59}$$

$\bar{x}_{t-1}$ is set to the original point

$$\bar{x}_{t-1} = [0, 0, 0] \tag{60}$$

According to Eqs. (57) and (58), the motion and observation models in the polar coordinate system are as follows:

$$\hat{x}_t = \mathrm{f}(\bar{x}_{t-1}, u_t) = \begin{bmatrix} \hat{l}_{x_t} \\ \hat{\beta}_{x_t} \\ \widehat{\theta}_{x_t} \end{bmatrix} = \begin{bmatrix} \Delta T V_t \\ \alpha_t \\ \Delta T V_t \sin(\alpha_t)/\mathrm{L} \end{bmatrix} \tag{61}$$

$$\hat{z}_t = [\hat{l}_{m_j}, \hat{\beta}_{m_j}]^{\mathrm{T}} = \mathrm{s}(\hat{x}_t, \bar{\Theta}_t) = \mathrm{s}(\hat{x}_t, z_{t-1}) \tag{62}$$

The predicted pose $\hat{x}_t$, the estimated position of the jth landmark $\bar{m}_j$, and $\bar{x}_{t-1}$ construct a triangle which is shown in Fig. 5.

The prediction of the observation $\hat{z}_{m_j} = [\hat{l}_{m_j}, \hat{\beta}_{m_j}]$ for the jth landmark is calculated from this model. First, $\gamma_{\bar{x}_{t-1}}$ is calculated

$$\gamma_o = \beta_{m_j} - \hat{\beta}_{x_t} \tag{63}$$

$$\gamma_{\bar{x}_{t-1}} = \begin{cases} |\gamma_o||\gamma_o & | \le \pi \\ 2\pi - |\gamma_o & ||\gamma_o| > \pi \end{cases} \tag{64}$$

By using the cosine theorem, the distance $\hat{l}_{m_j}$ between $\hat{x}_t$ and $\bar{m}_j$ can be calculated as

$$\hat{l}_{m_j} = \sqrt{\hat{l}_{x_t}^2 + l_{m_j}^2 - 2\hat{l}_{x_t} l_{m_j} \cos(\gamma_{\bar{x}_{t-1}})} \tag{65}$$

https://doi.org/10.1017/S0263574716000527 Published online by Cambridge University Press

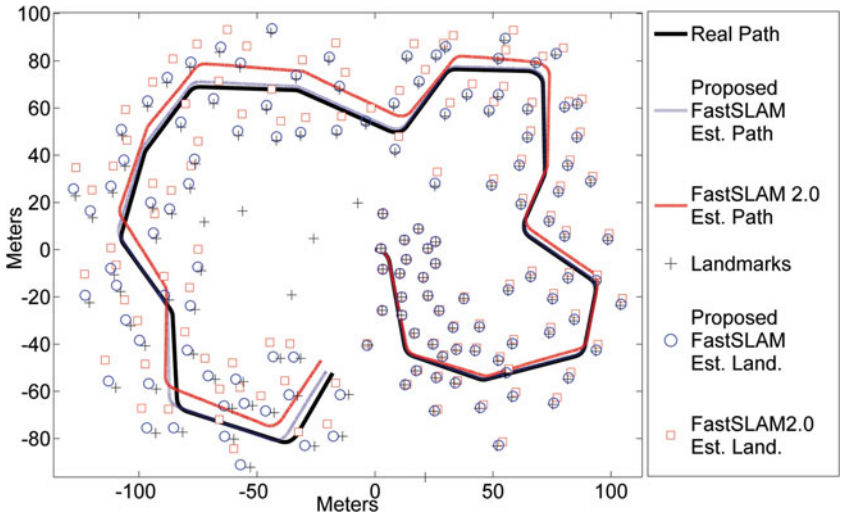

Fig. 6. The comparison between the two algorithms based on an incorrect **Q**. The control and observation noise covariance matrices are static. **Q** is amplified mistakenly by a factor of 3, and **R** is correct.

By using the cosine theorem, the angle $\gamma_{\hat{x}_t}$ can be calculated as

$$\gamma_{\hat{x}_t} = \arccos\left(\frac{\hat{l}_{x_t}^2 + \hat{l}_{m_j}^2 - l_{m_j}^2}{2\hat{l}_{x_t}\hat{l}(m_j)}\right) \tag{66}$$

If γ_o is in $[0, \pi]$ or $[-2\pi, -\pi]$, the predicted robot pose $\hat{x}_t$ is under the line which starts from $\bar{x}_{t-1}$ to $\bar{m}_j$, and the angle $\tilde{\beta}_{m_j}$ is in $[\beta_{m_j}, \pi + \beta_{m_j}]$

$$\tilde{\beta}_{m_j} = \pi + \hat{\beta}_{x_t} - \gamma_{\hat{x}_t} \tag{67}$$

If γ_o is between $[\pi, 2\pi]$ or $[-\pi, 0]$, $\hat{x}_t$ is above the line which starts from $\hat{x}_{t-1}$ to $\bar{m}_j$, and $\tilde{\beta}_{m_j}$ is in $[\pi + \beta_{m_j}, 2\pi + \beta_{m_j}]$

$$\tilde{\beta}_{m_j} = \pi + \hat{\beta}_{x_t} + \gamma_{\hat{x}_t} \tag{68}$$

Considering the heading of the robot, the predicted observation angle $\hat{\beta}_{m_j}$ is calculated as

$$\hat{\beta}_{m_j} = \tilde{\beta}_{m_j} - \hat{\theta}_{x_t} \tag{69}$$

4.1.2. Simulation results and analysis. The comparisons between the proposed algorithm and FastSLAM 2.0 are based on a Matlab SLAM simulator which was implemented by Tim Bailey.[44] 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 $\frac{\pi}{6}$ rad. The speed noise ϵ_v is 0.3 m/s, and the angle noise ϵ_β is $\frac{\pi}{60}$ rad. It equips a ranger bearing sensor with a π rad frontal view and a maximum range of 30 m. The range noise of the observation δ_l is 0.2 m, and the angle noise δ_β is $\frac{\pi}{180}$ rad.

Figure 6 shows the comparison of the robot path and landmark estimations between the proposed algorithm and FastSLAM 2.0 with 50 particles. This comparison is based on the incorrect priori knowledge about **Q** and the correct priori knowledge about **R**. Since **Q** is amplified mistakenly, the robot pose estimation would be closer to the observation in FastSLAM 2.0. There is no great impact on accuracy by the incorrect priori knowledge in the beginning. With the cumulative errors, the excessive trust for the observation will seriously decreases the accuracy of the estimation. The proposed algorithm starts with a wrongly known priori knowledge about $\mathbf{Q_0}$ and then adapts $\mathbf{Q}_t$. By the FC&ASD-PSO searching, the adjusting step attempts to minimize the mismatch between $\mathbf{Q}_t$ and the real values. As there are enough observed landmarks in the simulation environment, the

https://doi.org/10.1017/S0263574716000527 Published online by Cambridge University Press

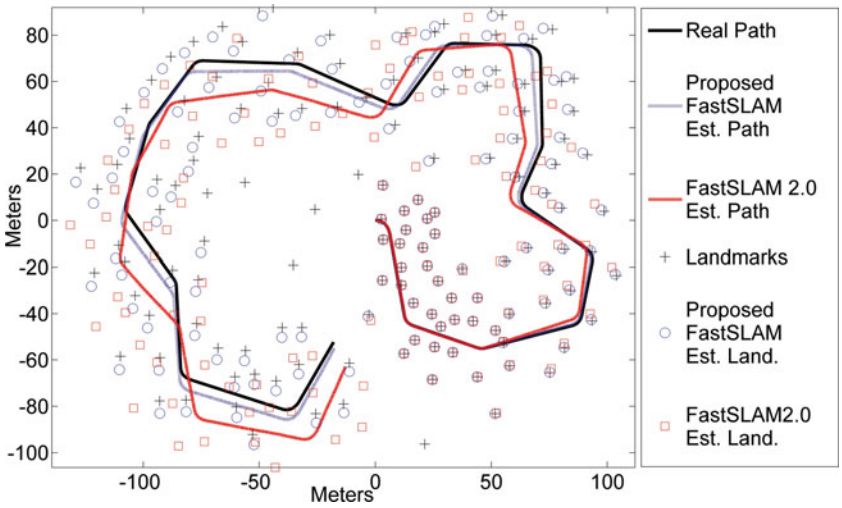

Fig. 7. The comparison between the two algorithms based on an incorrect **R**. The control and observation noise covariance matrix is static. **Q** is correct, and **R** is amplified mistakenly by a factor of 3.

estimations on the covariance matrices $\mathbf{R}_t$ and $\mathbf{Q}_t$ are very close to the real values in the proposed algorithm. In 50 experiments, the adjustments usually get 90% precision on $\mathbf{R}_t$ estimation and 80% precision on $\mathbf{Q}_t$ estimation before 32 iterations, and then the estimations fluctuate around the real value. For Q_t converges to the actual real control covariance, the proposed algorithm has higher precision than FastSLAM 2.0 in the estimation of the robot path and landmark positions while $\mathbf{Q}_t$ in FastSLAM 2.0 is kept fixed over time. From Fig. 6, it is noticeable that the estimated path by the proposed algorithm is almost the same with the real path whereas there are some difference between the path acquired by FastSLAM 2.0 and the real path.

Figure 7 shows the comparison between the proposed algorithm and FastSLAM 2.0 with 50 particles. This comparison is based on the incorrect priori knowledge about **R** and the correct priori knowledge about **Q**. Since **R** is amplified mistakenly, the robot pose estimation would be closer to the predicted pose in FastSLAM 2.0. The excessive trust for the control data decreases the accuracy of the estimation. The proposed algorithm starts with a wrongly known priori knowledge $\mathbf{R_0}$ and then adapts $\mathbf{R}_t$. The adjusting step attempts to minimize the mismatch between $\mathbf{R}_t$ and real values. Because there are enough observed landmarks, it takes a few iterations to get approximate estimations for the control and observation noise covariance matrices. Based on the approximate estimations of $\mathbf{Q}_t$ and $\mathbf{R}_t$, the estimated path by the proposed algorithm is more accurate than the estimation of FastSLAM 2.0 while these covariance matrices are kept fixed in FastSLAM 2.0.

Figure 8 shows the comparison between the proposed algorithm and FastSLAM 2.0 with 50 particles. This comparison is based on a temporary control error. In the real environments, it is likely that the robot motion changes drastically for some reasons, but the sensor does not sense it. For example, the phenomenon will occur when the robot is blocked by something. This experiment simulates this situation in which the robot is blocked from the 81th iteration to the 160th iteration. The velocity decreases to 0.1 m/s, and the sensed velocity is still 3 m/s in this interval. Although the robot pose can be revised by the observation, a good revision depends on how correct is the priori knowledge about $\mathbf{Q}_t$ and $\mathbf{R}_t$. The proposed algorithm attempts to adjust $\mathbf{Q}_t$ and make the robot pose estimation closer to the observation in this interval. In the adjusting step, there is a big difference between the updated pose and the predicted pose of the special particle. Since the big difference and enough observations, $\mathbf{Q}_t$ is rapidly increased. By this adjustment, the proposed algorithm can get more accurate estimation than FastSLAM 2.0. After the interval, the difference between the updated pose and the predicted pose of the special particle become smaller, $\mathbf{Q}_t$ is also decreased to the real value.

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 I shows the results of these experiments, and RMSE represents root mean square error. The results show that the proposed FastSLAM is more accurate

https://doi.org/10.1017/S0263574716000527 Published online by Cambridge University Press

Table I. The comparison of 50 experimental results between the proposed algorithm and FastSLAM 2.0.

Priori knowledge	Algorithm	RMSE in robot position estimation (m)	RMSE in landmark position estimation (m)	Average running time (s)
Incorrect **Q**	proposed FastSLAM	**2.598584**	**2.297926**	25.83
	FastSLAM 2.0	4.522,848	4.031649	**16.71**
Incorrect **R**	proposed FastSLAM	**2.814088**	**2.486992**	25.83
	FastSLAM 2.0	6.161934	5.441496	**16.71**

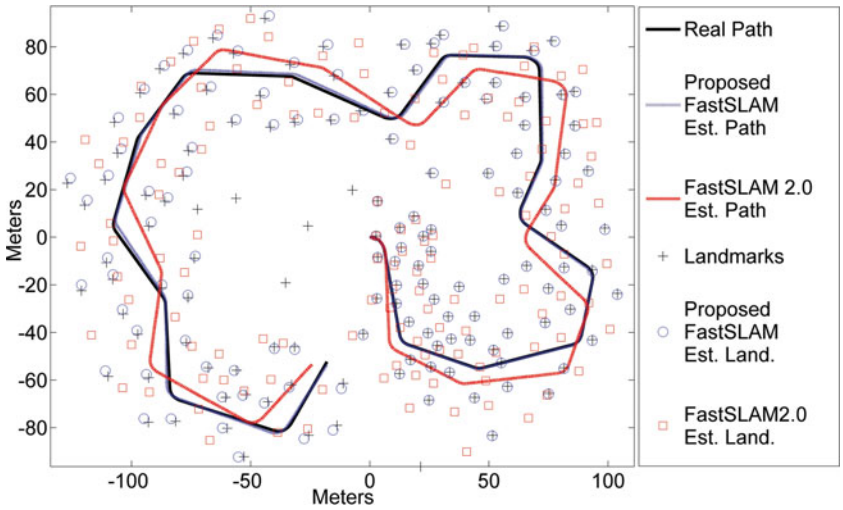

Fig. 8. The comparison between the two algorithms based on a temporary control error. In this simulation, the real robot velocity is 0.1 m/s from the 81th iteration to the 160th iteration, but the sensed velocity is still 3 m/s. After this interval, the robot motion returns to normal.

than FastSLAM 2.0 on the estimation of the robot pose and landmark positions. Because there is an extra adjusting process in the proposed algorithm, the running time of the proposed algorithm is more than that of the FastSLAM 2.0 algorithm.

In the 50 simulation experiments based on the incorrect priori knowledge about **Q** and the correct priori knowledge about **R**, the robot is assumed to run for about 200 s. Figure 9 shows the RMSE comparisons based on time series between two algorithms. Four sub-figures show the RMSE comparisons in x, y, orientation, and position, respectively. It can be seen that the robot position error of FastSLAM 2.0 is more than that of the proposed algorithm. Figure 10 shows the RMSE comparison in landmark positions between two algorithms after the loop is closed. It can be seen the landmark estimation errors of proposed algorithm is fewer than that of FastSLAM 2.0.

Figure 11 shows the RMSE comparisons based on time series. A total of 50 simulation experiments based on the incorrect priori knowledge about **R** were carried out for the comparisons. It can be seen that the robot position error of FastSLAM 2.0 is more than that of the proposed algorithm. For the map is dense with many landmarks in the simulation environment, the incorrect priori knowledge about **R** has more effect on the estimation accuracy, and RMSE is more than that of the incorrect priori knowledge about **Q** in FastSLAM 2.0. From Fig. 12, it can be seen the landmark position estimation error of proposed algorithm is fewer than that of FastSLAM 2.0.

FastSLAM in its current form cannot produce consistent estimates in the long-term.[14] By introducing the adjusting step, the proposed algorithm can get more accuracy than FastSLAM 2.0. Using more particles and better resample can prolong the consistency, but higher landmark density degrades consistency more quickly. In the simulation environment, the proposed algorithm becomes rapidly optimistic as same as FastSLAM 2.0.

https://doi.org/10.1017/S0263574716000527 Published online by Cambridge University Press

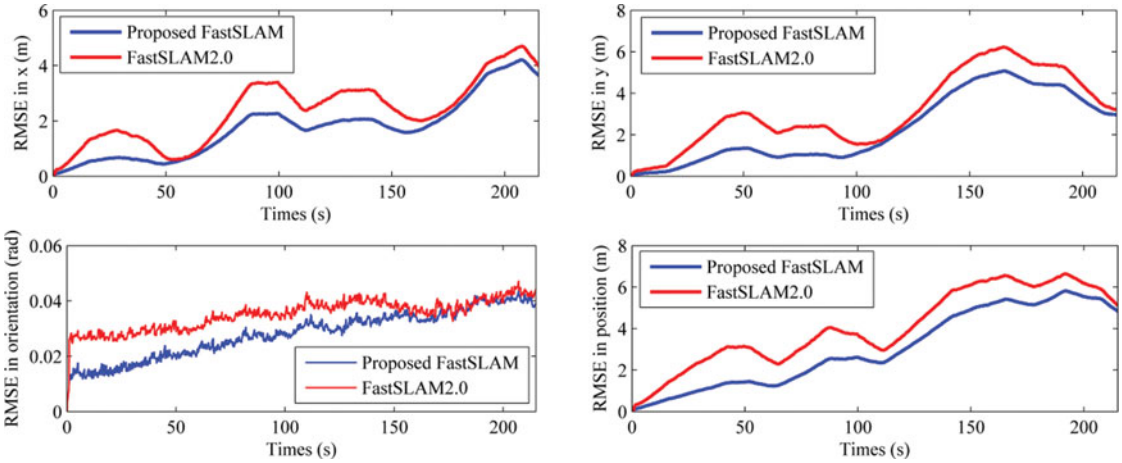

Fig. 9. The RMSE comparison in x, y, orientation, and position between two algorithms based on an incorrect **Q**.

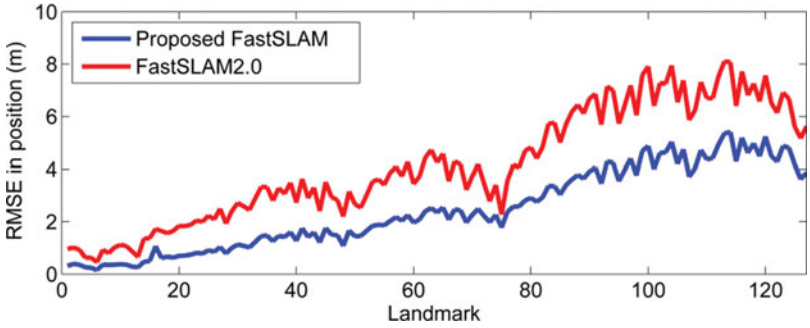

Fig. 10. The RMSE comparison in landmark positions between two algorithms based on an incorrect **Q**.

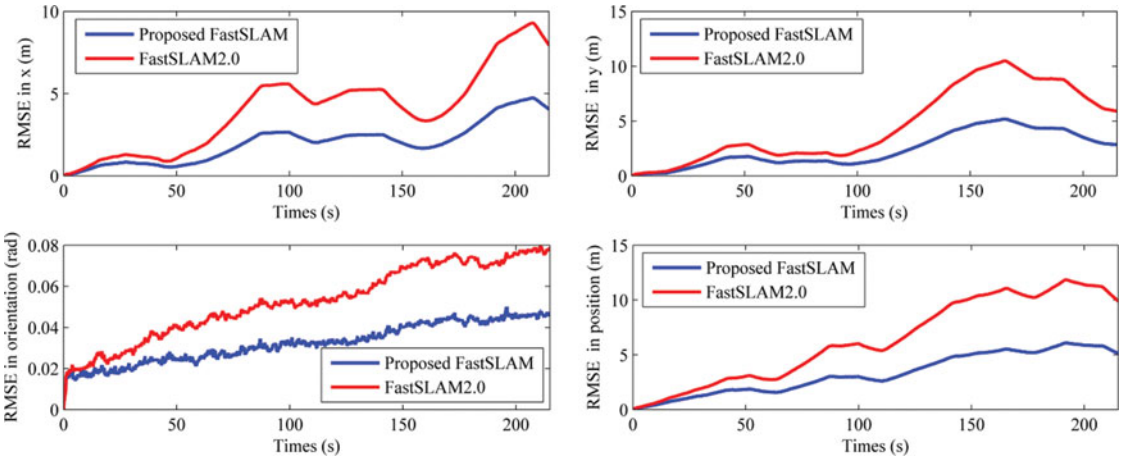

Fig. 11. The RMSE comparison in x, y, orientation, and position between two algorithms based on an incorrect **R**.

4.2. Experiments with "Car Park Dataset"

4.2.1. Experimental models. "Car Park Dataset", which was collected by Australian Centre for Field Robotics (ACFR) in Sydney,[45] is popular in the SLAM research community. A truck equipped with GPS, inertial and laser sensors was tested in an open parking area with artificial beacons. The motion model of the truck is shown in Fig. 13.

https://doi.org/10.1017/S0263574716000527 Published online by Cambridge University Press

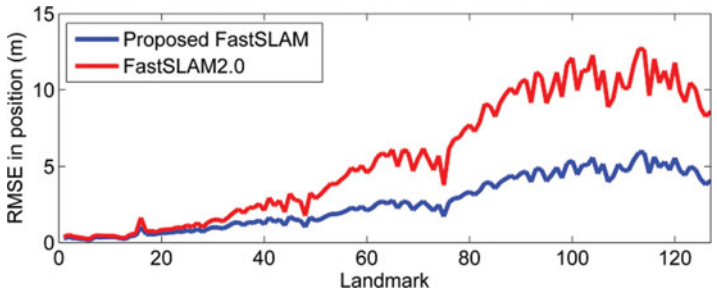

Fig. 12. The RMSE comparison in landmark positions between two algorithms based on an incorrect **R**.

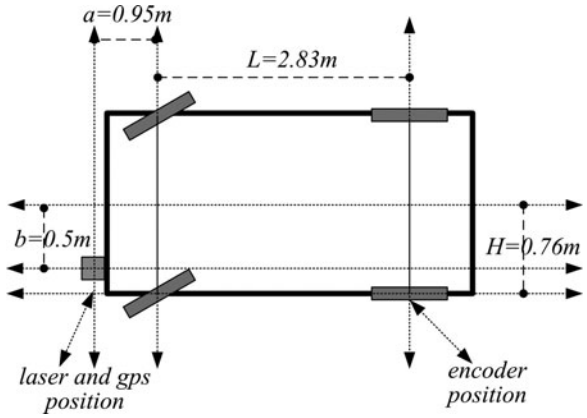

Fig. 13. The motion model of the truck.

The motion and observation model are as follows:

$$x_t = g(x_{t-1}, u_t) = \begin{bmatrix} x_{x_{t-1}} + \Delta T V_t(\cos(\theta_{x_{t-1}} + \alpha_t) - \frac{\tan(\alpha_t)}{L}(a\sin(\theta_{x_{t-1}}) + b\cos(\theta_{x_{t-1}}))) \\ y_{x_{t-1}} + \Delta T V_t(sin(\theta_{x_{t-1}} + \alpha_t) - \frac{\tan(\alpha_t)}{L}(b\sin(\theta_{x_{t-1}}) - a\cos(\theta_{x_{t-1}}))) \\ \theta_{x_{t-1}} + \frac{\Delta T V_t \tan(\alpha_t)}{L} \end{bmatrix} + \varepsilon \quad (70)$$

$$V_t = \frac{V_{et}}{1 - \frac{H}{L} \times \tan(\alpha_t)} \quad (71)$$

$$z_t(m_j) = h(x_t) = \begin{bmatrix} l_t(m_j) \\ \beta_t(m_j) \end{bmatrix} = \begin{bmatrix} \sqrt{(x_{m_j} - x_{x_t})^2 + (y_{m_j} - y_{x_t})^2} \\ \arctan\left(\frac{y_{m_j} - y_{x_t}}{x_{m_j} - x_{x_t}}\right) - \theta_{x_t} + \frac{\pi}{2} \end{bmatrix} + \delta \quad (72)$$

where V_t is the velocity of the center of the axle, V_{et} is the velocity of the back left wheel, ΔT is the sampling interval, and αt is the steering angle.

4.2.2. Experimental results and analysis. Figures 14 and 15 show the comparisons between the proposed algorithm and FastSLAM 2.0. These comparisons are based on incorrect **Q** and **R**, respectively. The estimated path and landmark positions in proposed algorithm are closer to the GPS path and landmark positions than the estimations in FastSLAM 2.0. Table II shows the map estimation in proposed algorithm is more accurate than the estimation in FastSLAM 2.0, and the running time is also more than that of FastSLAM 2.0. Because there are fewer observed landmarks in this environment, the adjusting process is only executed 57 times. The running time of each adjusting process is 0.0142 s, and the radar sampling interval is 0.2136 s. The proposed algorithm can meet the requirement of the real-time application.

https://doi.org/10.1017/S0263574716000527 Published online by Cambridge University Press

Table II. The comparison between the proposed algorithm and FastSLAM 2.0.

Priori knowledge	Algorithm	RMSE in landmark position estimation (m)	Running time (s)
Incorrect **Q**	proposed FastSLAM	**0.1049**	10.27
	FastSLAM 2.0	0.1233	**9.48**
Incorrect **R**	proposed FastSLAM	**0.1318**	10.27
	FastSLAM 2.0	0.1689	**9.48**

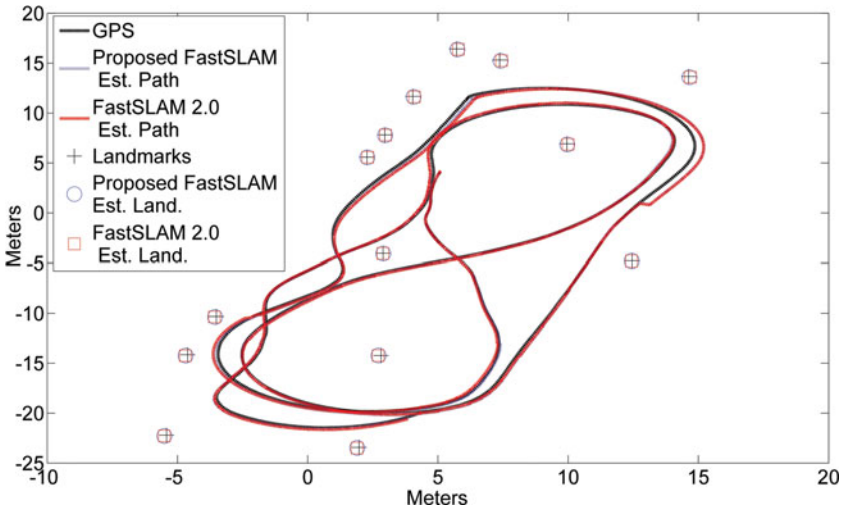

Fig. 14. The comparison between the proposed algorithm and FastSLAM 2.0 based on the incorrect priori knowledge about **Q**. The covariance matrix **Q** is amplified mistakenly by a factor of 3.

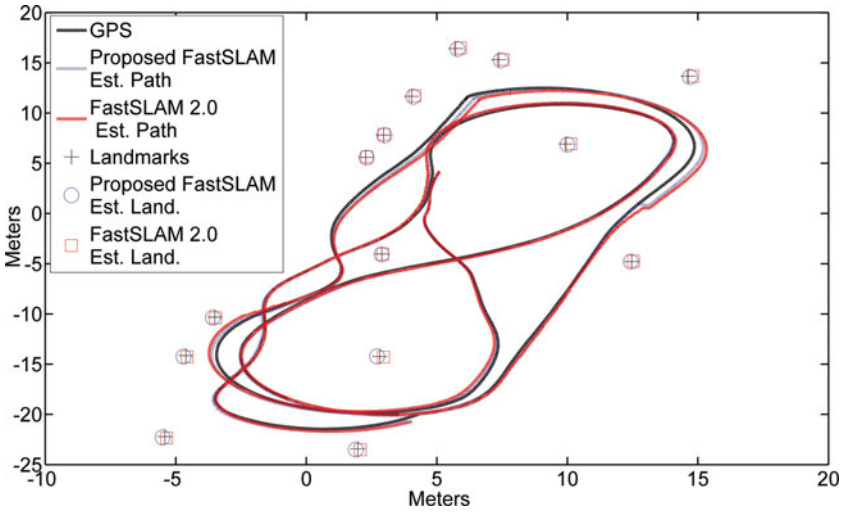

Fig. 15. The comparison between the proposed algorithm and FastSLAM 2.0 based on the incorrect priori knowledge about **R**. The covariance matrix R is amplified mistakenly by a factor of 3.

5. Conclusions

In the FastSLAM 2.0 framework, it is assumed that the priori knowledge about the control and observation noise matrices is known and correct. Since sensors are usually sensitive to environments, this assumption is not tenable in many applications. The incorrect priori knowledge about $\mathbf{Q}_t$ and $\mathbf{R}_t$ may seriously degrade the performance of FastSLAM 2.0. This paper proposes an improved FastSLAM algorithm based on FC&ASD-PSO. An extra step for adjusting the noise covariance matrices $\mathbf{Q}_t$ and $\mathbf{R}_t$ is introduced in this algorithm. In order to reduce the influence of the cumulative

https://doi.org/10.1017/S0263574716000527 Published online by Cambridge University Press

errors, this step uses a special particle. This particle, which uses the same coordinate system of control and observation, is used to compare time-adjacent observations. FC&ASD-PSO searches for the optimal solutions $\tilde{\mathbf{Q}}_t$ and $\tilde{\mathbf{R}}_t$ from time-adjacent observations. $\mathbf{Q}_t$ and $\mathbf{R}_t$ are updated by $\tilde{\mathbf{Q}}_t$ and $\tilde{\mathbf{R}}_t$. The experiment results and theoretical analysis show that compared to FastSLAM 2.0, the proposed algorithm has higher estimation precision and lower RMSE.

6. Supplementary Material

To view supplementary material for this article, please visit http://dx.doi.org/10.1017/S0263574716000527.

References

1. J. A. Fernández-Madrigal and J. B. Claraco, *Simultaneous Localization and Mapping for Mobile Robots: Introduction and Methods* (IGI Global, Hershey, 2013).
2. C. Fischer, Localisation, Tracking, and Navigation Support for Pedestrians in Uninstrumented and Unknown Environments *Ph.D. Thesis* (Lancaster University, Lancaster, UK, 2012).
3. H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *Robot. Autom. Mag.* **24**(2), 99–110 (2006).
4. Z. Kurt-Yavuz and S. Yavuz, "A Comparison of EKF, UKF, FastSLAM2.0, and UKF-Based FastSLAM Algorithms," *Proceedings of the IEEE International Conference on Intelligent Engineering Systems*, Lisboa, Portugal (2012) pp. 37–43.
5. M. Montemerlo, FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association *Ph.D. Thesis* (Carnegie Mellon University, Pittsburgh, USA, 2003).
6. F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *Int. J. Robot. Res.* **25**(12), 1181–1203 (2006).
7. S. Thrun and M. Montemerlo, "The GraphSLAM algorithm with application to large-scale mapping of urban structures," *Int. J. Robot. Res.* **25**(5-6), 403–429 (2006).
8. R. Martinez-Cantin, N. D. Freitas and J. A. Castellanos, "Analysis of Particle Methods for Simultaneous Robot Localization and Mapping and a New Algorithm: Marginal-SLAM," *Proceedings of the IEEE International Conference on Robotics and Automation*, Roma, Italy (2007) pp. 2415–2420.
9. G. Grisetti, C. Stachniss and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *Robotics*, **32**(1), 34–46 (2007).
10. Z. Chen, X. Dai, L. H. Jiang, C. Yang and B. Cai, "Adaptive iterated square-root cubature Kalman filter and its application to SLAM of a mobile robot," *Telkomnika Indonesian J. Electr. Eng.* **11**(12), 7213–7221 (2013).
11. M. C. Ruben and J. A. Castellanos, "Unscented SLAM for Large-Scale Outdoor Environments," *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Victoria, Canada (2005) pp. 3427–3432.
12. C. Gamallo, M. Mucientes and C. V. Regueiro, "A FastSLAM-based algorithm for omni directional cameras," *J. Phys. Agents* **7**(1), 12–21 (2012).
13. S. M. Chen, J. F. Yuan, F. Zhang and H. J. Fang, "Multirobot FastSLAM algorithm based on landmark consistency correction," *Math. Probl. Eng.* **2014**, 1–7 (2014).
14. T. Bailey, J. Nieto and E. Nebot, "Consistency of the FastSLAM Algorithm," *Proceedings of the IEEE International Conference on Robotics and Automation*, Orlando, Florida (2006) pp. 424–429.
15. C. Stachniss, G. Grisetti, D. Hhnel and W. Burgard, "Improved Rao-Blackwellized Mapping by Adaptive Sampling and Active Loop-closure," *Proceedings of the Workshop on Self-Organization of Adaptive Behavior*, Ilmenau, Germany (2004) pp. 1–15.
16. N. Nosan, I. K. Kim, H. C. Lee and B. H. Lee, "Analysis of Resampling Process for the Particle Depletion Problem in FastSLAM," *Proceedings of the IEEE International Conference on Robot and Human interactive Communication*, Jeju, Korea (2007) pp. 200–205.
17. W. Zhou and C. X. Zhao, "A FastSLAM 2.0 algorithm based on genetic algorithm," *Robotics*, **31**(1), 25–32 (2009).
18. H. Wang, Y. Yan and D. W. Wang, "A GA Based SLAM with Range Sensors Only," *Proceedings of the IEEE International Conference on Control, Automation, Robotics and Vision*, Singapore (2010) pp. 1796–1803.
19. C. Kim, R. Sakthivel and W. K. Chung, "Unscented FastSLAM: A robust and efficient solution to the SLAM problem," *Robotics*, **24**(4), 808–820 (2008).
20. D. Liu, G. R. Liu and M. H. Yu, "An improved FastSLAM framework based on particle swarm optimization and unscented particle filter," *J. Comput. Inf. Syst.* **8**(7), 2859–2866 (2012).
21. R. Havangi, H. D. Taghirad, M. A. Nekoui and M. Teshnehlab, "A square root unscented FastSLAM with improved proposal distribution and resampling," *Ind. Electron.* **61**(5), 2334–2345 (2014).
22. J. H. Zhu, N. N. Zhang and Z. J. Yang, "A SLAM algorithm based on central difference particle filter," *Acta Autom. Sin.* **36**(2), 249–257 (2010).

23. Y. Song, Y. D. Song and Q. L. Li, "Robust iterated sigma point FastSLAM algorithm for mobile robot simultaneous localization and mapping," *Chin. J. Mech. Eng.* **24**(4), 1–8 (2011).
24. X. J. Yan, C. X. Zhao and J. Z. Xiao, "A Novel FastSLAM Algorithm Based on Iterated Unscented Kalman Filter," *Proceedings of the IEEE International Conference on Robotics and Biometrics*, Phuket, Thailand (2011) pp. 1906–1911.
25. R. Havangi, M. Teshnelab, M. A. Nekoui and H. Taghirad, "An Adaptive Neuro-Fuzzy Rao-Blackwellized Particle Filter for SLAM," *Proceedings of the IEEE International Conference on Mechatronics*, Istanbul, Turkey (2011) pp. 487–492.
26. T. Z. Lv, "A novel EKF-SLAM algorithm against outlier disturbance," *Comput. Eng.* **38**(21), 1–4 (2012).
27. R. J. Fitzgerald, "Divergence of the Kalman filter," *Autom. Control*, **16**(6), 736–747 (1971).
28. X. D. Li and A. P. Engelbrecht, "Particle Swarm Optimization: An Introduction and its Recent Developments," *Proceedings of the Genetic Evolutionary Computation Conference*, London, England (2007) pp. 3391–3414.
29. R. Havangi, M. A. Nekoui and M. Teshnehlab, "An improved FastSLAM framework using soft computing," *Turk. J. Electr. Eng. Comput. Sci.* **20**(1), 25–46 (2012).
30. H. C. Lee, S. K. Park and J. S. Choi, "PSO-FastSLAM: An Improved FastSLAM Framework Using Particle Swarm Optimization," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, Texas (2009) pp. 2763–2768.
31. S. J. Davey, "Extensions to the Probabilistic Multi-Hypothesis Tracker for Improved Data Association," *Ph.D. Thesis* (University of Adelaide, Lancaster, UK, 2003).
32. M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem," *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada (2002) pp. 593–598.
33. M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, "FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges," *Proceedings of the International Joint Conference on Artificial Intelligence*, Acapulco, Mexico (2003) pp. 1151–1156.
34. J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Proceedings of the IEEE International Conference on Neural Networks*, Perth, Australia (1995) pp. 1942–1948.
35. N. Hamta, S. F. Ghomi, F. Jolai and M. A. Shirazi, "A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect," *Int. J. Prod. Econ.* **141**, 99–111 (2013).
36. A. I. Selvakumar and K. Thanushkodi, "A new particle swarm optimization solution to nonconvex economic dispatch problems," *IEEE Trans. Power Syst.* **22**(1), 42–51 (2007).
37. J. T. Machado, V. Kiryakova and F. Mainardi, "Recent history of fractional calculus," *Commun. Nonlinear Sci. Numer. Simul.* **16**(3), 1140–1153 (2011).
38. S. Das, *Functional Fractional Calculus* (Berlin, Springer, 2011).
39. P. C. Fourie and A. A. Groenwold, "The particle swarm optimization algorithm in size and shape optimization," *Struct. Multidiscip. Optim.* **23**, 259–267 (2002).
40. A. Weron and R. Weron, *Computer Simulation of Lévy α-stable Variables and Processes* (Berlin, Springer, 1995).
41. O. J. Woodman, "An Introduction to Inertial Navigation," *Technical Report* UCAM-CL-TR-696 (University of Cambridge, Cambridge, UK, 2007).
42. J. Nordh and K. Berntorp, "Extending the Occupancy Grid Concept for Low-Cost Sensor Based SLAM," *Proceedings of the* 10^{th} *IFAC Symposium on Robot Control*, Dubrovnik, Croatia (2012) pp. 151–156.
43. Z. Cui, J. Zeng and G. Sun, "Lévy velocity threshold particle swarm optimization," *ICIC Express Lett.* **2**(1), 23–28 (2008).
44. T. Bailey, "FastSLAM 2.0 source code," http://www.personal.acfr.usyd.edu.au/tbailey/software/index.html.
45. E. Nebot, J. Guivant and J. Nieto, "ACFR, experimental outdoor dataset," http://www.acfr.usyd.edu.au/homepages/academic/enebot/dataset.htm.