Research Article

A Strong Tracking SLAM Algorithm Based on the Suboptimal Fading Factor

Yunpei Liang ¹, Jiahui Dai ^{1,2}, Kequan Wang,² Xiaobo Li,² and Pengcheng Xu²

¹State Key Laboratory of Coal Mine Disaster Dynamics and Control, Chongqing University, Chongqing 400044, China

²China Coal Technology Engineering Group Chongqing Research Institute, Chongqing 400037, China

Correspondence should be addressed to Jiahui Dai; wjj-123-321@163.com

Received 17 August 2017; Revised 2 March 2018; Accepted 19 March 2018; Published 3 May 2018

Academic Editor: Pietro Siciliano

Copyright © 2018 Yunpei Liang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper proposes an innovative simultaneous localization and mapping (SLAM) algorithm which combines a strong tracking filter (STF), an unscented Kalman filter (UKF), and a particle filter (PF) to deal with the low accuracy of unscented FastSLAM (UFastSLAM). UFastSLAM lacks the capacity for online self-adaptive adjustment, and it is easily influenced by uncertain noise. The new algorithm updates each Sigma point in UFastSLAM by an adaptive algorithm and obtains optimized filter gain by the STF adjustment factor. It restrains the influence of uncertain noise and initial selection. Therefore, the state estimation would converge to the true value rapidly and the accuracy of system state estimation would be improved eventually. The results of simulations and practical tests show that strong tracking unscented FastSLAM (STUFastSLAM) has a significant improvement in accuracy and robustness.

1. Introduction

SLAM can solve a problem that a mobile robot can build a global map and estimate its own pose [1, 2] in an unknown environment. The basic principle is to use sensor information and odometer data with multifeature matching to build an environment map while determining the location of the robot in the map. However, because of the complicated external environment, unpredictable system noise, and weak sensor perception problems, the accuracy of SLAM is usually lower than expected [3].

Nowadays, many solutions are used to solve the problem of SLAM's low accuracy. The classic SLAM algorithm is based on an extended Kalman filter (EKF) [4]. In essence, EKF is based on a first-order Taylor formula, and it can solve the problem that the Kalman filter (KF) is not suitable for nonlinear systems [5]. However, there are also some fatal flaws [6], such as its high computational complexity.

Moreover, under the condition of high nonlinearity of the system, since EKF is adopted in the SLAM model, there are some problems of lower accuracy, even divergence and nonconvergence. So, based on the EKF-SLAM, some new algorithms have been proposed.

Aimed at the disadvantages of computational complexity and the erroneous data associations in EKF-SLAM, the Rao-Blackwellized particle filter was used in the SLAM algorithm by Murphy [7] and Doucet et al. [8]. After that, Montemerlo et al. improved the SLAM algorithm and proposed FastSLAM [9].

Based on the FastSLAM algorithm, Kim et al. proposed the UFastSLAM algorithm [10]. The core of the algorithm uses an unscented transform [11, 12] to estimate the transition probability density of SLAM [13]. It cancels the restriction condition of the system model. In other words, the system is not required to be linear. Meanwhile, there is no need to calculate the Jacobian matrix, so the state function

and the measurement function are not required to be continuously differentiable. An unscented transform can even be applied to discontinuous systems.

UFastSLAM uses an unscented transform to estimate the robot's pose and uses UKF instead of EKF to estimate the position of environmental features. In practical application, the presence of system noise among relevant information may influence the accuracy of UKF. Meanwhile, the main parameters and the filter gain cannot be adjusted online. It also lacks adaptive ability. In order to improve adaptability and robustness over a rough terrain, we add a strong tracking filter (STF) [14] on the basis of UFastSLAM. STF can adjust filter gain online to achieve the goal of an algorithm self-adapting. STF not only has strong model mismatch robustness but also has the advantages of clear conception and simple calculation. However, because it adds a strong tracking factor, which increases the amount of calculation, the computation time increases slightly with the same number of particles.

This paper presents a strong tracking adaptive UFastSLAM (STUFastSLAM) algorithm which is based on the suboptimal fading factor. The new algorithm updates every sigma point by using an adaptive algorithm. It gets the optimized filter gain by utilizing an ST online regulatory factor which can restrain the influence brought by uncertain noise and initial selection on state estimation. STUFastSLAM combines the adaptability of STF with the nonlinear high approximation of UKF. In addition, it also introduces a particle filter which is not influenced by non-Gaussian and nonlinear models. This algorithm has better adaptive ability, accuracy, and robustness [15]. The simulation and experiment demonstrate that the accuracy of the new algorithm is much better than other algorithms in complex environment.

2. Related Work

2.1. FastSLAM Algorithm Framework. The FastSLAM algorithm combines a particle filter with EKF and develops a new solution of localization and mapping simultaneously. FastSLAM estimates the robot's pose and environmental features by using PF and EKF. In this way, the computational complexity of FastSLAM is turned into $O(M \times N)$. Here, M is the number of particles and N is the features of environment.

FastSLAM divides the whole progress of localization and mapping into separate parts. Because the conditions of each environmental landmark are independent, it can also divide the whole mapping progress into mapping on each landmark. If the moving trajectories of the robot $X_t = \{x_1, x_2, \dots, x_t\}$ are known and the position of environmental landmarks $M_n = \{m_1, m_2, \dots, m_n\}$ is relatively independent, the joint posterior distribution $p(X_t, M | z_t, u_t, n_t)$ can be represented as follows:

$$p(X_t, M | z_t, u_t, n_t) = p(X_t | z_t, u_t, n_t) \prod_{i=1}^N p(m_i | X_t, z_t, u_t, n_t), \quad (1)$$

where $Z_t = \{z_1, z_2, \dots, z_t\}$ is a sequence of observed data and $U_t = \{u_1, u_2, \dots, u_t\}$ is a sequence of control information. Both of them are used to estimate the moving trajectories of the robot $X_t = \{x_1, x_2, \dots, x_t\}$, not just its position. n_t is the correlation variable.

In (1), PF is used to estimate the path and EKF is used to estimate landmarks. The FastSLAM method combines PF and EKF [9].

There are two versions of FastSLAM. Based on FastSLAM 1.0, the latest FastSLAM 2.0 introduces observation information Z_t . FastSLAM 2.0 improves the accuracy of the algorithm in the process of localization and mapping [16]. However, FastSLAM 2.0 also has some inherent defects. Firstly, FastSLAM 2.0 uses first-order linearization during the process of new position sampling. At this point, it will bring relatively large errors if there is an equation with a high degree of nonlinearity. Secondly, FastSLAM 2.0 uses standard EKF during the process of the landmarks' observation updating. There are some disadvantages when the FastSLAM 2.0 algorithm deals with equations with higher degrees of nonlinearity. Therefore, in order to solve these problems, some researchers have improved this algorithm under the frame of FastSLAM 2.0.

Under the frame of FastSLAM, Grisetti et al. proposed an improved FastSLAM for grid mapping with a Rao-Blackwellized particle filter [17]. This algorithm discretizes the map and uses lasers and odometers to design an important sampling function. However, the accuracy of the grid will affect FastSLAM greatly. In 2009, Arasaratna and Haykin proposed the cubature Kalman filter (CKF) [18]. CKF has a greater performance in regards to nonlinear approximation, numerical accuracy, and filter stability. However, there are some inherent drawbacks of this algorithm. For instance, it has to calculate the tedious covariance matrix square root and reconstruct the covariance matrix during the iteration of SLAM. Kang et al. proposed the neuroevolutionary optimization SLAM (Neo-SLAM) algorithm [19]. Neo-SLAM is a kind of SLAM algorithm which uses neural networks to learn the nonlinear dynamics model and noise statistics data autonomously. Neo-SLAM is insensitive to the changes of system and sensor noise compared with the other algorithms, so it can be used to reduce the robot motion errors [20]. In order to reduce the linearization error of the Kalman filter family, Yadkuri and Khosrowjerdi proposed a slam algorithm based on improved weighted mean extended Kalman filter (IWMEKF) [21].

3. Theoretical Background

3.1. Strong Tracking Filter. According to the literature [22], when the theoretical model matches the actual system exactly, the output residual sequence of the KF is nonautocorrelation or a mutually orthogonal Gaussian white noise sequence. However, EKF is a suboptimal filter. It is impossible that the residual sequence is completely uncorrelated. Therefore, as long as the EKF residual sequences are weakly autocorrelated, it can be considered that EKF has good filtering effect.

In regards to the STF online real-time adjustment of variable gain matrix K_k , the theoretical basis (orthogonality principle) is as follows:

Condition 1

$$E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] = \min. \quad (2)$$

Condition 2

$$E(\varepsilon_{k+j}\varepsilon_k^T) = 0, \quad k = 1, 2, \dots, \quad j = 1, 2, \dots \quad (3)$$

Condition 1 is the performance index of EKF.

Condition 2 requires that the output residual sequences in different time remain orthogonal.

When there is uncertainty in the system model, if the EKF state estimation value deviates from the true state of the system, it will inevitably show up on the mean and magnitude of the output residual sequence. In this case, if the gain matrix K_k is adjusted online, condition 2 is forcibly established so that the residual sequences can still be orthogonal to each other and always have Gaussian white noise-like properties. Therefore, when the system model is uncertain, the STF can still keep the capability of effective tracking of status.

STF adjusts the gain matrix in real time online and forces the output residual sequence to keep orthogonality by introducing a fading factor λ_{k+1} in the state prediction covariance [22]. Therefore, STF can keep the traceability for the system state even if the system model is uncertain. The fading factor λ_{k+1} is calculated as follows [23]:

$$\lambda_{k+1} = \begin{cases} \lambda_0 & \lambda_0 \geq 1, \\ 1 & \lambda_0 < 1, \end{cases} \quad \lambda_0 = \frac{\text{tr}[N_{k+1}]}{\text{tr}[M_{k+1}]}, \quad (4)$$

where $\text{tr}(\cdot)$ is the operator of trace of the matrix.

$$N_{k+1} = V_{k+1} - H_{k+1} Q_k H_{k+1}^T - R_{k+1}, \quad (5)$$

$$M_{k+1} = H_{k+1} \Phi_{k+1,k} P_k \Phi_{k+1,k}^T H_{k+1}^T, \quad (6)$$

where H_{k+1} is the $m \times n$ dimension measurement matrix, Q_k is the covariance matrix of control noise, R_{k+1} is the covariance matrix of observation noise, and $\Phi_{k+1,k}$ is the $n \times n$ dimension state-transition matrix. P_k is the state prediction covariance.

V_{k+1} is the covariance matrix of the actual output residual sequence, which is estimated by

$$V_{k+1} = \begin{cases} \varepsilon_1 \varepsilon_1^T, & k = 0 \\ \frac{\rho V_k + \varepsilon_{k+1} \varepsilon_{k+1}^T}{1 + \rho}, & k \geq 1, \end{cases} \quad (7)$$

where $0 < \rho \leq 1$ is the forgetting factor; the value is set to 0.95 usually.

The derivation process is shown in the appendix.

4. Description of STUFastSLAM

STUFastSLAM is a kind of process which is based on FastSLAM 2.0 as a frame and uses UKF instead of first-order linearization and EKF. Besides, we introduce STF to solve the problem of the poor robustness of UKF when the model is uncertain. The process of STUFastSLAM algorithm is as follows [24]:

4.1. Robot State Estimation. At first, the robot's pose, the control input, and the observation are estimated as a whole. The state matrix after augmentation is

$$\begin{aligned} x_{t-1}^{a[k]} &= \begin{bmatrix} x_{t-1}^{[k]} & 0 & 0 \end{bmatrix}^T, \\ P_{t-1}^{a[k]} &= \begin{bmatrix} P_{t-1}^{[k]} & 0 & 0 \\ 0 & R_t & 0 \\ 0 & 0 & Q_t \end{bmatrix}, \end{aligned} \quad (8)$$

where $x_{t-1}^{a[k]}$ is an augmented matrix which contains the robot's pose, the control input, and the observation, and its dimension is 7×1 . $P_{t-1}^{a[k]}$ is an augmented covariance matrix which contains robot covariance, control noise, and observation noise, and its dimension is 7×7 . R_t and Q_t are the observation noise covariance and control noise covariance, respectively [25].

Step 1. $2N + 1$ sigma points are extracted around the mean point deterministically. N is the dimension of the mean vector. Here $N = 7$. So, the set of extracted sigma points are [26]

$$\chi_{t-1}^{a[0][k]} = x_{t-1}^{a[k]}, \quad (9)$$

$$\chi_{t-1}^{a[i][k]} = x_{t-1}^{a[k]} + \left(\sqrt{(N + \lambda) P_{t-1}^{a[k]}} \right)_i \quad (i = 1, \dots, N), \quad (10)$$

$$\chi_{t-1}^{a[i][k]} = x_{t-1}^{a[k]} - \left(\sqrt{(N + \lambda) P_{t-1}^{a[k]}} \right)_{i-N} \quad (i = 1, \dots, 2N), \quad (11)$$

where, at this moment, each sigma point contains robot state, control input, and observation information, so that we can calculate the robot state information by the sigma point which is transformed by the nonlinear kinematic model. We can get the robot state information as follows.

Step 2. Get the prediction equation

$$\bar{\chi}_t^{[i][k]} = f(u_t, \chi_{t-1}^{[i][k]}). \quad (12)$$

The robot state mean and the prediction of the covariance can be calculated as follows:

$$x_{t|t-1}^k = \sum_{i=0}^{2N} \omega_m^{[i]} \bar{\chi}_t^{[i][k]}, \quad (13)$$

$$P_{t|t-1}^{k(l)} = \sum_{i=0}^{2N} \omega_c^{[i]} \left(\bar{\chi}_t^{[i][k]} - x_{t|t-1}^k \right) \left(\bar{\chi}_t^{[i][k]} - x_{t|t-1}^k \right)^T. \quad (14)$$

In order to make the filter more robust when the system model is uncertain, the fading factor λ_t is introduced into the state prediction covariance matrix. $P_{t|t-1}^k$ is calculated as follows:

$$P_{t|t-1}^k = \lambda_t \sum_{i=0}^{2N} \omega_c^{[i]} \left(\bar{\chi}_t^{[i][k]} - x_{t|t-1}^k \right) \left(\bar{\chi}_t^{[i][k]} - x_{t|t-1}^k \right)^T. \quad (15)$$

Here λ_t is unknown; we will calculate the value of λ_t in the latter part.

$\omega_m^{[i]}$ is used for calculating the weight of the mean. $\omega_c^{[i]}$ is used for calculating the weight of the covariance. The formulas are as follows.

$$\omega_m^{[0]} = \frac{\lambda}{n + \lambda}, \quad (16)$$

$$\omega_c^{[0]} = \frac{\lambda}{n + \lambda} + (1 - \alpha^2 + \beta), \quad (17)$$

$$\omega_c^{[i]} = \omega_m^{[i]} = \frac{1}{2(n + \lambda)} \quad \text{for } i = 1, \dots, 2n, \quad (18)$$

where $\beta = 2$. In order to obtain more accuracy on the robot's state, observation information must be considered. We update the prediction by using the observed features. The formulas are as follows.

$$\bar{N}_t^{[i][k](l)} = h \left(\bar{\chi}_t^{[i][k](l)}, \mu_{k,t-1}^k \right), \quad (19)$$

$$\hat{n}_t^{[k](l)} = \sum_{i=0}^{2N} \omega_m^{[i]} \bar{N}_t^{[i][k](l)}, \quad (20)$$

$$S_t^{[k](l)} = \sum_{i=0}^{2N} \omega_c^{[i]} \left(\bar{N}_t^{[i][k](l)} - \hat{n}_t^{[k](l)} \right) \left(\bar{N}_t^{[i][k](l)} - \hat{n}_t^{[k](l)} \right)^T. \quad (21)$$

Utilizing $x_{t|t-1}^k$ and $P_{t|t-1}^{k(l)}$, the sigma point is calculated according to the sampling method selected by step 1. Based on the nonlinear function and (19), (20), and (21), we can get the sigma point $\bar{N}_t^{[i][k](l)}$. From $\bar{N}_t^{[i][k](l)}$, we can get the output prediction $\hat{n}_t^{[k](l)}$, the autocovariance $S_t^{[k](l)}$, and the cross covariance $\Sigma_t^{x,n[k](l)}$.

$\bar{N}_t^{[i][k](l)}$ is the sigma point which is calculated via the nonlinear observation equation. $\hat{n}_t^{[k](l)}$ is the prediction of observation, and $S_t^{[k](l)}$ is the updating vector (autocovariance). Thus, the cross covariance matrix $\Sigma_t^{x,n[k](l)}$ can be obtained.

$$\sum_t^{x,n[k](l)} = \sum_{i=0}^{2N} \omega_c^{[i]} \left(\bar{\chi}_t^{[i][k](l)} - x_{t|t-1}^k \right) \left(\bar{N}_t^{[i][k](l)} - \hat{n}_t^{[k](l)} \right)^T. \quad (22)$$

$P_{t|t-1}^{k(l)}$, $S_t^{[k](l)}$, and $\Sigma_t^{x,n[k](l)}$ are known. The fading factor λ_t in (15) can be calculated by (4), (5), (6), and (7).

The fading factor λ_t has already been obtained. So we use $x_{t|t-1}^k$ and $P_{t|t-1}^k$ to calculate the sigma point $\chi_{t-1}^{a[i][k]}$ in accordance with the sample strategy in step 1. The sigma point $\chi_{t-1}^{a[i][k]}$ transforms into $\bar{N}_t^{[i][k]}$ through the nonlinear

measurement function. We can calculate the output prediction $\hat{n}_t^{[k]}$, the autocovariance $S_t^{[k]}$, and the cross-covariance $\Sigma_t^{x,n[k]}$ from $\bar{N}_t^{[i][k]}$.

$$\bar{N}_t^{[i][k]} = h \left(\bar{\chi}_t^{[i][k]}, \mu_{k,t-1}^k \right), \quad (23)$$

$$\hat{n}_t^{[k]} = \sum_{i=0}^{2N} \omega_m^{[i]} \bar{N}_t^{[i][k]}, \quad (24)$$

$$S_t^{[k]} = \sum_{i=0}^{2N} \omega_c^{[i]} \left(\bar{N}_t^{[i][k]} - \hat{n}_t^{[k]} \right) \left(\bar{N}_t^{[i][k]} - \hat{n}_t^{[k]} \right)^T, \quad (25)$$

$$\sum_t^{x,n[k]} = \sum_{i=0}^{2N} \omega_c^{[i]} \left(\bar{\chi}_t^{[i][k]} - x_{t|t-1}^k \right) \left(\bar{N}_t^{[i][k]} - \hat{n}_t^{[k]} \right)^T. \quad (26)$$

Finally, the Kalman gain $K_t^{[k]}$ can be calculated as follows:

$$K_t^{[k]} = \sum_t^{x,n[k]} \left(S_t^{[k]} \right)^{-1}. \quad (27)$$

At time t , the robot state mean $x_t^{[k]}$ and the covariance estimation $P_t^{[k]}$ can be obtained as follows:

$$x_t^{[k]} = x_{t|t-1}^k + K_t^{[k]} \left(z_t - \hat{n}_t^{[k]} \right), \quad (28)$$

$$P_t^{[k]} = P_{t|t-1}^k - K_t^{[k]} S_t^{[k]} \left(K_t^{[k]} \right)^T. \quad (29)$$

We update the set of sigma points in (10) based on (28) and (29). The new sigma points set are as follows:

$$\chi_t^{a[i][k]} = \left[x_t^{a[k]}, x_t^{a[k]} \pm \sqrt{(N + \lambda) P_t^{a[k]}} \right]. \quad (30)$$

4.2. Environmental Feature Position Estimation. We then have to estimate and update the observed environmental features. Similar with the robot's pose estimation, we need to structure the sigma points set before we estimate the environmental features. As the SLAM is in 2D scene, the dimension of the environmental feature is 2. The sigma points are as follows:

$$\begin{aligned} \chi^{a[0][k]} &= \mu_{n,t-1}^{[k]}, \\ \chi^{a[i][k]} &= \mu_{n,t-1}^{[k]} + \left(\sqrt{(n + \lambda) \sum_{nt,t-1}^{[k]}} \right)_i \quad (i = 1, \dots, n), \\ \chi^{a[i][k]} &= \mu_{n,t-1}^{[k]} - \left(\sqrt{(n + \lambda) \sum_{nt,t-1}^{[k]}} \right)_{i-n} \quad (i = 1, \dots, 2n), \end{aligned} \quad (31)$$

where $\mu_{n,t-1}^{[k]}$ is the mean of the n th feature at $t - 1$, and $\Sigma_{n,t-1}^{[k]}$ is its covariance. The other parameters can be calculated like in the previous section. Then, we can get

$$\begin{aligned}
\bar{N}_t^{[i][k]} &= h\left(\chi^{[i][k]}, x_t^k\right) \quad (i = 0, \dots, 2n), \\
\hat{z}_t^{[k]} &= \sum_{i=0}^{2N} \omega_m^{[i]} \bar{z}_t^{[i][k]}, \\
\bar{S}_t^{[k]} &= \sum_{i=0}^{2N} \omega_c^{[i]} \left(\bar{z}_t^{[i][k]} - \hat{z}_t^{[k]}\right) \left(\bar{z}_t^{[i][k]} - \hat{z}_t^{[k]}\right)^T.
\end{aligned} \tag{32}$$

Here, $\bar{N}_t^{[i][k]}$ is the sigma point set which is calculated by nonlinear observation formulas. $\hat{z}_t^{[k]}$ and $\bar{S}_t^{[k]}$ are the predicted mean and covariance of observation, respectively. Then, we can calculate the Kalman gain $\bar{K}_t^{[k]}$ and update the mean $\mu_{nt,t}^{[k]}$ and covariance $\Sigma_{nt,t}^{[k]}$ of the environmental feature.

$$\bar{\Sigma}_t^{[k]} = \sum_{i=0}^{2N} \omega_c^{[i]} \left(X^{[i][k]} - \mu_{nt,t-1}^{[k]}\right) \left(\bar{z}_t^{[i][k]} - \hat{z}_t^{[k]}\right)^T, \tag{33}$$

$$\bar{K}_t^{[k]} = \bar{\Sigma}_t^{[k]} \left(\bar{S}_t^{[k]}\right)^{-1}, \tag{34}$$

$$\mu_{nt,t}^{[k]} = \mu_{nt,t-1}^{[k]} + \bar{K}_t^{[k]} \left(z_t - \hat{z}_t^{[k]}\right), \tag{35}$$

$$\Sigma_{nt,t}^{[k]} = \Sigma_{nt,t-1}^{[k]} - \bar{K}_t^{[k]} \bar{S}_t^{[k]} \left(\bar{K}_t^{[k]}\right)^T. \tag{36}$$

4.3. Resampling Strategy. In the last step, we have to make the judgment resample for the particles. The resampling method is exactly the same as that of FastSLAM 2.0.

$$\begin{aligned}
w_t^{[k]} &= \frac{\text{target distribution}}{\text{proposal distribution}} \\
&= \left|2\pi L_t^{[k]}\right|^{-1/2} \exp\left\{-\frac{1}{2}(z_t - \bar{z}_t)L_t^{[k-1]}(z_t - \bar{z}_t)\right\}.
\end{aligned} \tag{37}$$

Here,

$$L_t^{[k]} = \left(\sum_t^{x,n[k]}\right)^T \left(P_t^{[k]}\right)^{-1} \sum_t^{x,n[k]} + \bar{S}_t^{[k]}. \tag{38}$$

5. STUFastSLAM Overview

According to Section 4, the STUFastSLAM algorithm is composed of two parts. One is the estimation of robot pose, and the other is the estimation of the location of environmental features. We drew flow charts for these two parts separately. The flow charts of the algorithm are as follows:

Figure 1 is the flow chart of robot state estimation. It is divided into two parts: prediction and update.

The flow chart of environmental feature position estimation is shown as Figure 2.

The explanation of the STUFastSLAM algorithm is in the pseudocode. The pseudocode of the STUFastSLAM algorithm is as follows.

The covariance $P_{t|t-1}^{k(l)}$, the autocovariance $S_t^{[k](l)}$, and the cross covariance $\Sigma_t^{x,n[k](l)}$ indicate that the fading factor has not been introduced.

The covariance $P_{t|t-1}^k$, the autocovariance, and the cross-covariance indicate that the fading factor has been introduced.

6. Simulation Result

In order to verify the reliability and validity of STUFastSLAM, we designed a simulation experiment and compared the results among STUFastSLAM, FastSLAM 2.0, and UFastSLAM. The simulation experiment was carried out on the Matlab2015 platform. The SLAM simulator was published by the University of Sydney robot center [27]. We use the car-like motion model, and the observation model is the Bearing-Rang radius scan model. The wheel base is 4 m, and the robot velocity is $V = 3$ m/s. The effective range of sensor is 40 m, and its front view is 180° . The noise is (0,1) distributional Gaussian white noise. The process noise is ($\sigma_V = 0.3$ m/s, $\sigma_G = 3^\circ$), and the observation noise is ($\sigma_r = 0.4$ m, $\sigma_\theta = 5^\circ$).

Figures 3 and 4 are simulation environment graphs. The green “×”s are the environmental features, and the “O”s are the waypoints of the robot’s moving path. The curve is the robot’s reference path. The robot moves from the point of origin and follows the reference path for 2 laps. In this experiment, we set 35 features and 17 waypoints. The robot starts from (0,0) point and moves along the path (Figure 3). All the tests were repeated 30 times, and we obtained the average value as the statistical results.

In Figure 4, the black curve is the estimated path trajectory of the robot. The red points show the estimated landmarks. Because of odometer and observation errors, there is some deviation between the estimated and true paths and between estimated and true landmarks. In the simulation experiment, we compare the true path with the estimated path of different algorithms and compare the location deviation of the true landmarks with the estimated ones to compare the accuracy of the algorithms. The smaller the deviation, the more accurately the robot draws the map. By comparing the time of the car run the whole distance, we can compare the calculation speed of the algorithms. The shorter the time, the lower the computational complexity of the algorithm, and the simpler the calculation.

Figures 5–7 show the comparison of the simulation path of FastSLAM 2.0, UFastSLAM, and STUFastSLAM. In the beginning, the estimation accuracy of each of these 3 algorithms is high. The estimated path is basically consistent with the true path, and the deviation between them is very small. With the increase of robot walking distance, there is some deviation between the estimated path and the true path. We find that the deviation of FastSLAM 2.0 is the greatest. The accuracy of UFastSLAM is higher than that of FastSLAM 2.0 in the latter period. Due to the addition of adaptive factors, STUFastSLAM not only has greatly improved the estimation accuracy but also can keep high position estimation in the process of robot exploration. The coincidence degree of the estimated path and the true path is the highest.

Figure 8 is the comparison of the position estimation error among standard FastSLAM 2.0, UFastSLAM, and

FIGURE 1: Flow chart of robot state estimation.

FIGURE 2: Flow chart of environmental feature position estimation.

STUFastSLAM. The system noise and the observation noise are in the condition of Gauss white noise. The x -axis presents the number of steps the experiment took, and each experiment takes 17,383 steps. The first lap is 1~8691 steps, and the second lap is 8692~17,383 steps. Firstly, the estimation accuracy is significantly improved in the second lap in Figure 8. This is because when the robot runs the second lap, it matches and corrects the path of the first lap. Secondly, the accuracy of FastSLAM 2.0 is lower than that of the other two algorithms, and the maximum location error is higher than in the other two. In STUFastSLAM, the introduction of an adaptive factor causes the error in the simulation state to be well controlled. In most cases, the error of STUFastSLAM is less than those of FastSLAM 2.0 and UFastSLAM. Especially in the simulation of the second lap, the error

during the overall process is in the minimum state, which is consistent with the previous theoretical analysis.

In order to verify the accuracy and stability of the three algorithms in different levels of environmental noise, we set five groups of environmental noise: $(\sigma_r = 0.1 \text{ m}, \sigma_b = 3^\circ)$, $(\sigma_r = 0.3 \text{ m}, \sigma_b = 3^\circ)$, $(\sigma_r = 0.4 \text{ m}, \sigma_b = 5^\circ)$, and $(\sigma_r = 0.4 \text{ m}, \sigma_b = 8^\circ)$. The particle number is set to 10, and the odometer noise is set to $(\sigma_v = 0.3 \text{ m/s}, \sigma_G = 3^\circ)$. We use the root mean square error (RMSE) to estimate the deviation between the observation value and the true value. RMSE is defined as

$$\text{RMSE} = \left(\frac{\sum_{k=0}^{Ns} |x_k - \hat{x}_k|^2}{Ns - 1} \right)^{1/2} \quad (39)$$

where x_k is the true state of the robot and $\hat{x}_k$ is the estimated state at time k .

From Figure 9, we can find that the RMSE of these three algorithms rise to different extents as the noise grows, which represents their accuracy decreasing. But in all the cases, the RMSE of STUFastSLAM is minimal. When the noise is $\sigma_r = 0.1$, $\sigma_b = 3$, the RMSE of UFastSLAM is very close to that of FastSLAM, and the RMSE of STUFastSLAM is significantly lower. With the increase in the noise level, the advantage of STUFastSLAM becomes more and more clear and the gap between the RMSE of STUFastSLAM and those of the other two algorithms grows. When the noise is $\sigma_r = 0.4$, $\sigma_b = 8$, the gap between the RMSEs of UFastSLAM and STUFastSLAM becomes small, but the gap with FastSLAM 2.0 is even more evident.

Figure 10 shows the computation time of these three algorithms under the same RMSE. From Figure 10, the computation time of STUFastSLAM is least, and the computation time of UFastSLAM is slightly more than that of STUFastSLAM. The computation time of FastSLAM 2.0 is far greater than those of the other two. It indicates that STUFastSLAM

```

Initialization parameters
for k = 1 to M
  % Robot state estimation
  Extract the robot position  $x_{t-1}^{[k]}$  from sigma points set  $X_{t-1}$  (10)
  Predict mean  $x_{t|t-1}^k$  (13) and covariance  $P_{t|t-1}^{k(l)}$  (14) of robot
  Associate observation information data
  % The calculation of fading factor
  Calculate the fading factor  $\lambda_{k+1}$  (4) from the prediction of the covariance  $P_{t|t-1}^{k(l)}$ , the autocovariance  $S_t^{[k](l)}$  (21) and the cross
  covariance  $\Sigma_t^{x,n[k](l)}$  (22).
  % Introduce the fading factor  $\lambda_t$ 
  Obtain the predicted covariance of the robot after the introduction fading factor (15)
  Obtain the autocovariance  $S_t^{[k]}$  (25) and the cross-covariance  $\Sigma_t^{x,n[k]}$  (26) of the robot after the introduction fading factor
  for  $\hat{k}$ =known feature
    Update mean  $x_t^{[k]}$  (28) and covariance  $P_t^{[k]}$  (29) of the robot
    Update sigma points (30)
    Calculate importance weight (17)
  end for
  % Environmental features position estimation
  if  $\hat{k}$ =new feature
    Initialize new feature mean  $\mu_{n,t-1}^{[k]}$  and covariance  $\Sigma_{n,t-1}^{[k]}$ 
  else
    Update mean (35) and covariance (36) of features
  end if
  for unobserved features
     $\mu_{n,t}^{[k]} = \mu_{n,t-1}^{[k]}$ ,  $\Sigma_{n,t}^{[k]} = \Sigma_{n,t-1}^{[k]}$ 
  end for
  Add updated  $\{x_t^{[k]}, N_t^{[k]}, \mu_{N_t^{[k]},t}^k, \Sigma_{N_t^{[k]},t}^k\}$  points set  $X_{aux}$ 
end for
% Resampling strategy
for k = 1 to M
  Normalize weight and calculate  $\omega_{neff}$  (37)
  if  $\omega_{neff} < \omega_\lambda$ 
    Resample
  else
    Maintain the original particle weight
  end for
Add new particles to  $X_t$ 
Return  $X_t$ .

```

PSEUDOCODE 1: STUFastSLAM algorithm.

FIGURE 3: Simulation environment.

FIGURE 4: Robot path graph.

FIGURE 5: FastSLAM 2.0 simulation graph.

FIGURE 6: UFastSLAM 2.0 simulation graph.

FIGURE 7: STUFastSLAM simulation graph.

still has the time advantage compared with the other two algorithms under the same accuracy.

Figure 11 shows the comparison of RMSE in the case of different quantities of particles when STUFastSLAM has the

FIGURE 8: Estimated position error in different algorithms.

FIGURE 9: RMSE of different algorithms in different levels of measurement noise.

same noise level. When the number of particles is less than or equal to 20, the RMSE value of this algorithm shows a linear downtrend basically. When the number of particles is greater than 20, the decrease in the RMSE value becomes gentler. It indicates that the accuracy of this algorithm is gradually improved with the increase in particle number. However, when the number of particles is greater than 20, the increase in accuracy is not obvious. Meanwhile, a great deal of computation time is also needed. Therefore, the STUFastSLAM algorithm essentially achieves its optimal level when the number of particles is 20.

FIGURE 10: Computation time under the same RMSE.

FIGURE 11: RMSE of STUFastSLAM by different numbers of particles.

FIGURE 12: Estimation position error of STUFastSLAM in different numbers of particles.

FIGURE 13: P3DX robot.

FIGURE 14: Experiment environment.

FIGURE 15: Office room environment CAD graph.

Figure 12 shows the position estimation error of STUFastSLAM in the case of different numbers of particles. Figure 12 corresponds to Figure 11; when the number of particles is between 5 and 20, the estimated accuracy improves obviously as the number of particles increases. When the number of particles is 40, the accuracy is only slightly higher than that of 20 particles.

7. Experiment Base on P3DX Robot

In order to verify the actual performance of the algorithm, we simulated robot real-time localization and mapping under rough road conditions in the office.

This experimental platform adopts the Pioneer-3DX (P3DX) autonomous mobile robot which is produced by American Active Media Company, as shown in Figure 13.

FIGURE 16: Office floor plan.

TABLE 1: Control point dimension.

Control points	Dimension (mm)
A-B	12,345
A-C	4979
A-D	5300
A-E	7825

It is a small intelligent robot driven by two wheels with differential drive. It has many advantages, such as flexibility, stability, generality, and easy operation. P3DX is the most mature experimental platform for the wheeled mobile robots in the world.

7.1. Experiment 1. Figure 14 is a real photo of our office. The experiment will build a 2D environment map of the office in this condition. In order to verify the SLAM performance under rough road conditions, we threw some gravel and sand on the ground to simulate the condition of rough roads and introduce skidding.

In order to test the performance of the algorithm quantitatively, we used the laser rangefinder to measure the office and drew the following environment map with SolidWorks as shown in Figure 15.

We chose 5 points A, B, C, D, and E in Figure 16 as the distance control points of indoor environment which can be used to compare the accuracy of algorithms. The dimensions are as shown in Table 1:

We compared the performance with UFastSLAM and STUFastSLAM in the same test experiment conditions. We controlled the robot to walk in the office to build an office map.

Figure 17 is the environment model which was built by UFastSLAM. Figure 17 is the somewhat distorted map due to the robot's wheels skidding. The map is too vague to show the real environment of the office.

Figure 18 is the environment model which was built by STUFastSLAM. As the reader can see in Figure 18, the map

FIGURE 17: Indoor environment scanning chart (UFastSLAM).

FIGURE 18: Indoor environment scanning chart (STUFastSLAM).

is much clearer than in Figure 17. In this map, the corners can be seen clearly and the wall is vertical. The entire map is seen in much sharper detail.

In order to quantify the gap between the two algorithms, we built the office map 10 times to obtain the average dimensions of the control points as follows:

Table 2 shows the average dimensions of the control points obtained by STUFastSLAM.

Table 3 shows the average dimensions of the control points obtained by UFastSLAM.

The resulting analysis table is shown below:

From Figure 19, we can see that in the same conditions, the average error of STUFastSLAM is much lower than the one of UFastSLAM when mapping an indoor environment.

TABLE 2: STUFastSLAM control point dimension.

Control points	Dimension (mm)
A-B	12,234
A-C	4872
A-D	5431
A-E	7715

TABLE 3: UFastSLAM control point dimension.

Control points	Dimension (mm)
A-B	11,873
A-C	4869
A-D	5325
A-E	7705

FIGURE 19: Average error of different algorithms in a real condition.

FIGURE 20: RMSE of different algorithms in a real condition.

Figure 20 shows the RMSE of different algorithms in a real environment. We can see that the RMSE of STUFastSLAM is much lower than that of UFastSLAM.

7.2. Experiment 2. In this experiment, we let the robot go from point A, through gate B, walk in a 13.36 m long,

FIGURE 21: Robot indoor walking track.

FIGURE 22: Comparison of maps drawn by STUFastSLAM and UFastSLAM.

4.77 m wide hut, and then walk out of gate C and enter a 2.26 m wide corridor and return to the starting point (as shown in Figure 21). The road was covered with gravel to simulate the situation of the robot walking on harsh road conditions. The accuracy of the drawn map is judged by comparing the map size measured by the slam algorithm with the actual layout of the building.

We used the STUFastSLAM algorithm and the UFastSLAM algorithm to create maps and compare the two maps.

In Figure 22, the red map is drawn by the STUFastSLAM algorithm, and the gray map is drawn by the UFastSLAM algorithm. As can be seen from the figure, the red map is obviously more regular than the gray map and can reflect various indoor details not mapped by UFastSLAM. In order to quantitatively compare the accuracy of the two algorithms, the distances between the points in the map are measured and compared with the actual measurements of the building. We got Figure 23.

From Figure 23, the error of STUFastSLAM is far lower than that of UFastSLAM in terms of maximum error, minimum error, average error, and standard deviation.

Figure 24 shows the RMSE of different algorithms in a real environment. We can see that the RMSE of STUFastSLAM is much lower than that of UFastSLAM. The RMSE value is slightly different from that of Figure 20 due to the differences in the road conditions and the indoor space. It still, however, certifies that the new algorithm can obtain better performance in practical applications.

To verify the effect of the number of particles on map accuracy, we set the number of particles in the STUFastSLAM algorithm to be 10, 20, and 40 particles, respectively.

FIGURE 23: Comparison of error in STUFastSLAM and UFastSLAM.

FIGURE 24: RMSE of different algorithms in a real condition.

As can be seen from Figure 25, as the number of particles increases, the quality of the map drawn by the robot progressively improves. We measured the size of the three maps and compared them with the actual maps, as shown in Figure 26.

Figure 26 and Figure 11 provide confirmation for one another. As the number of particles increases, the RMSE value of the map decreases, and thus the accuracy of the map is increased.

8. Conclusions

This article puts forth a solution for the problems of rescue robots' poor mapping accuracy and mapping maps easily diverging in complex terrain environments in the form of the STUFastSLAM algorithm. This article is based on the framework of UFastSLAM and innovatively introduces the strong tracking adaptive factor. For each sigma point in the UFastSLAM algorithm, adaptive algorithms are used to update them in real time. Optimized filter gain is obtained through the STF online adjustment factor, and the influence of uncertain noise and initial value selection on the state estimation is suppressed. By this, system state

FIGURE 25: Comparison of maps with different particle numbers.

FIGURE 26: RMSE of STUFastSLAM by different numbers of particles.

estimation accuracy is increased. The robustness of the system is enhanced, so the new algorithm is more suitable for environments with harsh terrain. In addition, we carried out simulation experiments to verify the effectiveness of the

algorithm. The experimental results show that the accuracy and stability of STUFastSLAM are better than those of the other two algorithms under different noise conditions. Even under the same accuracy, it has less computation time than the other two algorithms have. Finally, in the office, we simulated robot real-time localization and mapping under rough road conditions. The results show that the map drawn by the rescue robot much better reflects the surrounding environment, and the map has high accuracy and good reliability.

Through a series of simulations and experiments, the performance of STUFastSLAM is verified. The algorithm is superior to other similar algorithms in accuracy and reliability. It can successfully accomplish the task of mapping in complex terrain environments.

Appendix

Detailed Derivation and Calculation of Suboptimal Fading Factor λ_{k+1}

Given the nonlinear model

$$\begin{aligned} x_k &= f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1}), \\ z_k &= h_k(x_k, v_k), \end{aligned} \quad (\text{A.1})$$

where x_k is the state parameter of robot, z_k is the observation of environmental feature, f is the nonlinear kinematic model, and h is the environment observation model. u_{k-1} is the control action for the robot in the discrete time $[k, k-1]$, v_k is the control noise and its covariance matrix is Q , and w_{k-1} is the observation noise of sensor and its covariance matrix is R . w_{k-1} and v_k represent unrelated white Gaussian noise.

The recurrence formula of the strong tracking filter with fading factor is shown:

$$\begin{aligned} \hat{x}_{k+1|k} &= f_k(\hat{x}_k) + q_k, \\ P_{k+1|k} &= \lambda_{k+1} \Phi_{k+1,k} P_k \Phi_{k+1,k}^T + Q_k, \\ \hat{z}_{k+1|k} &= h_{k+1}(\hat{x}_{k+1|k}) + r_{k+1}, \\ \hat{x}_{k+1} &= \hat{x}_{k+1|k} + K_{k+1}(z_{k+1} - \hat{z}_{k+1|k}), \\ K_{k+1} &= P_{k+1|k} H_{k+1}^T (H_{k+1} P_{k+1|k} H_{k+1}^T + R_{k+1})^{-1}, \\ P_{k+1} &= (I - K_{k+1} H_{k+1}) P_{k+1|k}, \\ \Phi_{k+1,k} &= \frac{\partial f_k(x_k)}{\partial x_k} \bigg|_{x_k = \hat{x}_k}, \\ H_{k+1,k} &= \frac{\partial h_{k+1}(x_{k+1})}{\partial x_{k+1}} \bigg|_{x_{k+1} = \hat{x}_{k+1|k}}, \end{aligned} \quad (\text{A.2})$$

where $\lambda_{k+1} \geq 1$ is the suboptimal fading factor. $\hat{x}$ is the estimate, P is the prediction error covariance matrix, $\hat{z}$ is the prediction of measurement value, K is Kalman gain, and k is the k th time. Φ is the $n \times n$ dimension state-transition matrix, and H is the $m \times n$ dimension measurement matrix. The theoretical output residual sequence is assumed as $\varepsilon_{k+1} = z_{k+1} - \hat{z}_{k+1|k}$; the suboptimal fading factor λ_{k+1} is then calculated as follows:

$$\begin{aligned} \lambda_{k+1} &= \begin{cases} \lambda_0 & \lambda_0 \geq 1, \\ 1 & \lambda_0 < 1, \end{cases} \quad \lambda_0 = \frac{\text{tr}[N_{k+1}]}{\text{tr}[M_{k+1}]}, \\ N_{k+1} &= V_{k+1} - H_{k+1} Q_k H_{k+1}^T - R_{k+1}, \\ M_{k+1} &= H_{k+1} \Phi_{k+1,k} P_k \Phi_{k+1,k}^T H_{k+1}^T \\ &= H_{k+1} [P_{k+1}^{(l)} - Q_k] H_{k+1}^T \\ &= H_{k+1} P_{k+1|k}^{(l)} H_{k+1}^T - H_{k+1} Q_k H_{k+1}^T \\ &= H_{k+1} P_{k+1|k}^{(l)} H_{k+1}^T + R_{k+1} - V_{k+1} + N_{k+1}, \end{aligned} \quad (\text{A.3})$$

where $\text{tr}(\cdot)$ is the operator of trace of the matrix. $P_{k+1|k}^{(l)}$ represents the state prediction covariance without introducing fading factors, which is calculated by

$$P_{k+1|k}^{(l)} = \Phi_{k+1,k} P_k \Phi_{k+1,k}^T + Q_k \quad (\text{A.4})$$

V_{k+1} is the covariance matrix of the actual output residual sequence, which is estimated by

$$V_{k+1} = \begin{cases} \varepsilon_1 \varepsilon_1^T, & k = 0 \\ \frac{\rho V_k + \varepsilon_{k+1} \varepsilon_{k+1}^T}{1 + \rho}, & k \geq 1. \end{cases} \quad (\text{A.5})$$

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is financially supported by the State Key Research Development Program of China (Grant no. 2016YFC0801404), the National Science and Technology Major Project of China (Grant no. 2016ZX05043005), and the National Natural Science Foundation of China (51674050), which are gratefully acknowledged.

References

- [1] M. Bryson and S. Sukkarieh, "Active airborne localisation and exploration in unknown environments using inertial SLAM," in *2006 IEEE Aerospace Conference*, p. 13, Big Sky, MT, USA, March 2006.
- [2] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.
- [3] D. Xuemei, L. Lang, and C. Mengyuan, "Strong tracking square root cubature Kalman filter based on SLAM algorithm," *Journal of Electronic Measurement and Instrument*, vol. 10, pp. 1493–1499, 2015.
- [4] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the EKF-SLAM algorithm," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3562–3568, Beijing, China, October 2006.
- [5] K. S. Kumar, N. R. Dustakar, and R. K. Jatotoh, "Evolutionary computational tools aided extended Kalman filter for ballistic target tracking," in *2010 3rd International Conference on*

- Emerging Trends in Engineering and Technology*, pp. 588–593, Goa, India, November 2010.
- [6] L. M. Paz, J. D. Tardós, and J. Neira, “Divide and conquer: EKF SLAM in $O(n)$,” *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1107–1120, 2008.
 - [7] K. Murphy, “Bayesian map learning in dynamic environments,” *Advances in Neural Information Processing Systems*, vol. 12, pp. 1015–1021, 2000.
 - [8] A. Doucet, N. De Freitas, K. Murphy, and S. Russell, “Rao-Blackwellised particle filtering for dynamic Bayesian networks,” in *UAI’00 Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pp. 176–183, Stanford, CA, USA, June–July 2000.
 - [9] M. S. Montemerlo, *Fastslam: a factored solution to the simultaneous localization and mapping problem with unknown data association*, Carnegie Mellon University, Pittsburgh, PA, USA, 2003.
 - [10] C. Kim, R. Sakthivel, and W. K. Chung, “Unscented FastSLAM: a robust and efficient solution to the SLAM problem,” *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 808–820, 2008.
 - [11] S. J. Julier and J. K. Uhlmann, “A new method for the nonlinear transformation of means and covariance in filters and estimators,” *IEEE Transactions on Automatic Control*, vol. 45, no. 3, pp. 477–482, 2000.
 - [12] S. Julier and J. Uhlmann, “Unscented filtering and nonlinear estimation,” *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2005.
 - [13] E. Wan and R. Van Der Merwe, “The unscented Kalman filter for nonlinear estimation,” in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pp. 153–158, Lake Louise, AB, Canada, October 2000.
 - [14] D. H. Zhou and Q. L. Wang, “Strong tracking filtering of nonlinear systems with colored noise,” *Journal of Beijing Institute of Technology*, vol. 17, no. 3, pp. 321–326, 1997.
 - [15] Z. Hongren, J. Zhongliang, and P. Wang, *Tracking of Maneuvering Targets*, National Defense Industry Press, Beijing, 1991.
 - [16] 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,” in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pp. 1151–1156, Acapulco, Mexico, 2003.
 - [17] G. Grisetti, C. Stachniss, and 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] I. Arasaratnam and S. Haykin, “Cubature Kalman filters,” *IEEE Transactions on Automatic Control*, vol. 54, no. 6, pp. 1254–1269, 2009.
 - [19] J. G. Kang, S. Kim, S. Y. An, and S. Y. Oh, “A new approach to simultaneous localization and map building with implicit model learning using neuro evolutionary optimization,” *Applied Intelligence*, vol. 36, no. 1, pp. 242–269, 2012.
 - [20] A. Chatterjee and F. Matsuno, “A neuro-fuzzy assisted extended Kalman filter-based approach for simultaneous localization and mapping (SLAM) problems,” *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 5, pp. 984–997, 2007.
 - [21] F. F. Yadkuri and M. J. Khosrowjerdi, “Methods for improving the linearization problem of extended Kalman filter,” *Journal of Intelligent & Robotic Systems*, vol. 78, no. 3–4, pp. 485–497, 2015.
 - [22] P. S. MayBeck, *Stochastic Models, Estimation and Control*, Academic Press, New York, NY, USA, 1979.
 - [23] Z. Lin, *Filtering Theory of Nonlinear Systems*, National Defense Industry Press, Beijing, 2012.
 - [24] Z. Xiaobing, *Implementation of Unscented FastSLAM Algorithm for Mobile Robot Based on Vision*, Harbin Institute of Technology, Heilongjiang, 2010.
 - [25] G. Campion, G. Bastin, and B. D’Andréa-Novel, “Structural properties and classification of kinematic and dynamic models of wheeled mobile robots,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 1, pp. 47–62, 1996.
 - [26] S. J. Julier, “The scaled unscented transformation,” in *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, vol. 6, pp. 4555–4559, Anchorage, AK, USA, May 2002.
 - [27] T. Bailey, “SLAM simulations,” September 2012, <http://www.personal.acfr.usyd.edu.au/tbailey/software/index.html>.