2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

A UPF-UKF Framework For SLAM

Xiang Wang
Department of Electrical and Computer Engineering,
University of Alberta,
Edmonton, AB, T6G 2V4, Canada
xiang@cs.ualberta.ca

Hong Zhang
Department of Computing Science,
University of Alberta,
Edmonton, AB, T6G 2E8, Canada
zhang@cs.ualberta.ca

***Abstract*— In this paper we propose a SLAM framework which is based on an algorithm that combines an Unscented Particle Filter (UPF) and Unscented Kalman Filters (UKFs). A UPF is used to estimate robot's poses and the UKFs are used to represent landmark positions. UPF can estimate robot poses more consistently and accurately than generic Particle Filters (PFs), especially when models are highly non-linear or noises are not Gaussian. UKF can update landmarks more accurately compared to popular EKF's when highly non-linear observation models are used. In addition, our algorithm avoids the calculation of the Jacobian for both motion model and the observation model, which could be extremely difficult for high order systems. The calculation cost of a UPF is on the same order of magnitude as a Particle Filter (PF), which uses Kalman filters to generate proposal distributions, and the calculation cost of a UKF is equivalent to an EKF. As a result, our SLAM framework is more accurate than other popular SLAM frameworks while its efficiency is maintained. Simulation results are shown to validate the performance goals.**

I. INTRODUCTION

SLAM solves the problem of building a map and then recovering the robot pose from observations obtained from sensors mounted on the robot. The robot senses its own motion and at the same time identifies nearby landmarks. Because these two types of measurements are both subject to noise, they are essentially a probabilistic estimate of the map along with the robot's momentary pose.

The map can be denoted by Θ, which consists of a collection of landmarks, each of which is denoted θ_n. The total number of landmarks is denoted N. The robot pose is s_t, where t is a discrete time index. In a 2D environment, robot poses are typically expressed by its two-dimensional Cartesian coordinates and its orientation. The sequence $s^t = s_1, s_2, \ldots, s_t$ denotes the path of the robot until time t. The measurement at time t is denoted y_t. Without loss of generality , the assumption of observing only one feature at any time is adopted for convenience and the data association is assumed to be known. For each measurement y_t, m_t specifies the identity of the observed feature, where m_t belongs to the set $1, \ldots, N$.

The observation model is then written in the following form:

$$y_t = h(\theta_{m_t}, s_t) + \varepsilon_t$$

The measurement model is governed by a function h and disturbed by a random noise. The noise at time t is the random variable ε_t, which is usually approximated by a Gaussian distribution.

Control commands of a robot are the other information used to solve SLAM problems. Control commands are denoted u_t, which are the collective motor commands carried out in the time interval $[t-1, t)$. The motion model is then written in the following form:

$$s_t = f(u_t, s_{t-1}) + \delta_t$$

The goal of SLAM is to recover the map from measurements y_t and controls u_t. The Bayes filter is the core of many popular SLAM algorithms. If both f and h are linear and all random variables are Gaussian, the Bayes filter is equivalent to the well-known Kalman filter.

II. RELATED WORK

The SLAM problem was introduced in a seminal paper [6] and was first developed into an implemented system using an Extended Kalman Filter (EKF) [5]. In the EKF framework, a high-dimensional Gaussian is used to represent the robot pose and landmark position estimates. Because the correlations between errors in the robot pose and landmarks are stored in the covariance matrix of the Gaussian , EKF can accommodate the correlated nature of errors in the map. Based on the EKF framework, many recent achievements have been developed ([3], [4]).

There are two limitations of the EKF approach. The first one is the high computational cost. It is clear that maintaining a multivariate Gaussian requires time quadratic in the dimension of the map. Therefore, it is more efficient to build a set of smaller maps than build a large one. Based on this idea, several approaches have been proposed ([7],[8]) to overcome this limitation. The second limitation is related to the data association problem. It is critical to choose the correct data association hypotheses because different data association hypotheses lead to different maps. As shown empirically, maintaining posteriors over multiple data associations make the SLAM algorithms robust. However, Gaussians cannot represent multi-modal distributions so only the most likely data association can be incorporated. As a result, the approach tends to fail catastrophically when the incorporated data association is incorrect.

Another family of SLAM algorithms is called FastSLAM [9]. In [10], the author pointed out that the errors of the feature estimates would be independent if a robot path was given. FastSLAM algorithms were developed based on this property of the SLAM problems. Particle Filters (PFs) are

used to estimate robot path. Condition on these particles the mapping problem is factored into separate problems. Therefore, one EKF for each feature is used to update the feature estimate. The basic algorithm can be implemented in time logarithmic in the number of landmarks. Hence, FastSLAM offers computational advantages over plain EKF implementations and many of its descendants.

FastSLAM has two advantages over EKF-style approaches. The key one is that filter can maintain posteriors over multiple data associations and not just the most likely one. As a result, FastSLAM is significantly more robust to data association problems than EKF which is based on maximum likelihood data association. The other advantage is that FastSLAM can handle non-linear robot motion models while EKF-style algorithms can only approximate such models by linearization.

Since FastSLAM approximates non-linear observation models by linearizing functions, it can not handle severe non-linear camera models. Comparing with FastSLAM 1.0, FastSLAM 2.0 generates much less samples of robot poses with low likelihood. The idea is that robot poses are sampled using an improved proposal distribution which considers both the motion and the measurements. However, this method again can not handle highly non-linear observation models because EKF-style approximation is used to calculate the proposed distribution.

III. Background Knowledge

The Unscented Transform (UT) is a method to calculate the statistics of a random variable, which undergoes a nonlinear transformation ([1], [2]). UT is built on the idea that it is easier to approximate a probability distribution than an arbitrary nonlinear function. The Unscented Kalman Filter (UKF) is a straightforward application of the UT. UKF recursively minimizes mean-square-error estimations, where the state random variable (RV) is redefined as the concatenation of the original state and noise variables. Work in [2] has shown that the UKF leads to more accurate results than the EKF and that in particular it generates much better estimates of the covariance of the states where the EKF often seems to underestimate this quantity. The complete UKF algorithm that updates the mean $\bar{x}$ and covariance P of the states is as follows:

- Set the augmented state vector x_t^a and augmented covariance matrix P_t^a by incorporating the noise as follows:

$$x_t^a = [x_t^T, \delta_t^T, \varepsilon_t^T]^T \tag{1}$$

$$P_t^a = \begin{bmatrix} P_t & 0 & 0 \\ 0 & Q_t & 0 \\ 0 & 0 & R_t \end{bmatrix} \tag{2}$$

  where x_t^T is the original state vector, δ_t^T, is the motion noise, ε_t^T is the observation noise, P_t is the original covariance, Q_t is the motion noise covariance and R_t is the observation noise covariance.
- Initialize the mean and covariance of the states with:

$$\bar{x}_0 = E[x_0] \tag{3}$$

$$P_0 = E[(x_0 - \bar{x}_0)(x_0 - \bar{x}_0)^T] \tag{4}$$

$$\bar{x}_0^a = E[x^a] = [\bar{x}_0^T \ 0 \ 0]^T \tag{5}$$

$$P_0^a = \begin{bmatrix} P_0 & 0 & 0 \\ 0 & Q_0 & 0 \\ 0 & 0 & R_0 \end{bmatrix} \tag{6}$$

- For time $t \in 1...\infty$:
  (a) Computing Sigma Points and Weights:

$$\lambda = \alpha^2(n_x + \kappa) - n_a \tag{7}$$

$$\chi_{t-1}^a = [\bar{x}_{t-1}^a \quad \bar{x}_{t-1}^a \pm (\sqrt{(n_a + \lambda)P_{t-1}^a})_i] \tag{8}$$

$$W_{0(m)} = \frac{\lambda}{n_a + \lambda} \tag{9}$$

$$W_{0(c)} = \frac{\lambda}{n_a + \lambda} + (1 - \alpha^2 + \beta) \tag{10}$$

$$W_{i(m)} = W_{i(c)} = \frac{1}{2(n_a + \lambda)} \qquad i = 1, \cdots, 2n_a \tag{11}$$

  where κ is a scaling parameter, α is a positive scaling parameter which could be made as small as possible to minimize higher order effects, β is a parameter which minimizes the effects from high order terms, n_a is the dimension of the x_t^a and $\chi^a = [(\chi^x)^T, (\chi^v)^T, (\chi^n)^T]^T$, $W_{0,i(m)}$ are weights to calculate the mean, $W_{0,i(c)}$ are weights to calculate the covariance and $(\sqrt{(n_a + \lambda)P_{t-1}^a})_i$ is the ith row or column of the matrix square root of $(n_a + \lambda)P_{t-1}^a$.
  (b) States Prediction:

$$\chi_{t|t-1}^x = f(\chi_{t-1}^x, \ \chi_{t-1}^v) \tag{12}$$

$$\bar{x}_{t|t-1} = \sum_{i=0}^{2n_a} W_{i(m)} \chi_{i,t|t-1}^x \tag{13}$$

$$P_{t|t-1} = \sum_{i=0}^{2n_a} W_{i(c)} [\chi_{i,t|t-1}^x - \bar{x}_{t|t-1}][\chi_{i,t|t-1}^x - \bar{x}_{t|t-1}]^T \tag{14}$$

$$Y_{t|t-1} = h(\chi_{t|t-1}^x, \ \chi_{t-1}^n) \tag{15}$$

$$\bar{y}_{t|t-1} = \sum_{i=0}^{2n_a} W_{i(m)} Y_{i,t|t-1} \tag{16}$$

  where $f(.)$ is the motion model and the $h(.)$ is the observation model.
  (c) States Update

$$P_{\bar{y}_t, \bar{y}_t} = \sum_{i=0}^{2n_a} W_{i(c)} [Y_{i,t|t-1} - \bar{y}_{t|t-1}][Y_{i,t|t-1} - \bar{y}_{t|t-1}]^T \tag{17}$$

$$P_{x_t, y_t} = \sum_{i=0}^{2n_a} W_{i(c)} [\chi_{i,t|t-1} - \bar{x}_{t|t-1}][Y_{i,t|t-1} - \bar{y}_{t|t-1}]^T \tag{18}$$

$$K_t = P_{x_t,y_t} P^{-1}_{\tilde{y}_t,\tilde{y}} \tag{19}$$

$$\bar{x}_t = \bar{x}_{t|t-1} + K_t(y_t - \bar{y}_{t|t-1}) \tag{20}$$

$$P_t = P_{t|t-1} - K_t P_{\tilde{y}_t,\tilde{y}} K_t^T \tag{21}$$

where $n_a = n_x + n_v + n_n$ and K is gain.

PFs require the design of proposal distributions approximate the posterior distribution reasonably well. In general, it is hard to design such a proposal. The most common strategy is to sample from the probabilistic model of the state evolution. This strategy can fail if the new measurements appear in the tail of the prior or if the likelihood is too peaked in comparison to the prior. To overcome this problem, several techniques based on linearization have been proposed. For example, FastSLAM 2.0 used linearization to include new measurements for a better proposal distribution as well.

As mentioned above, the UKF is capable of more accurately propagating the mean and covariance than the EKF. Distributions generated by the UKF have a larger support overlap with the true posterior distribution than the overlap achieved by the EKF estimates. For these reasons, UKF is a better candidate for more accurate proposal distribution generation for particle filters. The generated proposal distribution will have larger higher order moments and with means that are close to the true mean of the target distribution. Therefore, it has been proven theoretically and empirically that replacing EKF proposal by UKF proposal results in an Unscented Particle Filter (UPF) which performs better than other PFs, especially when the system is highly nonlinear [11]. Work in [11] also showed that under very loose assumptions, convergence of the UPF is ensured and that the convergence rate of the method is independent of the dimension of the state-space. The only crucial assumption is to ensure that the weights, W_t, are upper-bounded.

An UPF and UKF combined framework will be proposed in the next section. This framework updates robot poses using UPF and updates landmark locations using UKF. UPF can estimate the state more consistently and accurately. UKF can update feature positions accurately to the third order for Gaussians and higher order errors scaled by choice of transform parameters. In addition, the calculation of the Jacobian for both the motion model and the observation model is avoided. The calculation cost of the proposed framework is on the same order of magnitude as FastSLAM 2.0, so improved performance can be obtained from using this new framework.

IV. A UPF-UKF SLAM FRAMEWORK

Similar to the family of FastSLAM algorithms, the proposed SLAM framework exploits the factored posterior state estimates by maintaining $N + 1$ filters. One filter is over robot path, $p(s^t|m^t, y^t, u^t)$, and N separate filters are over feature locations, $p(\theta|s^t, m^t, y^t)$. All $N + 1$ filters are low dimensional. This factored representation is exact, rather than just an approximation. The UPF that is used to calculate the posterior over the robot path has the pleasing property that the amount of computation needed for each incremental update stays constant, regardless of the path length t. Additionally, it can cope well with non-linear robot motion models. The remaining N posteriors over feature locations $p(\theta|s^t, m^t, y^t)$ are calculated by UKF. Each UKF estimates a single landmark pose. The individual UKFs are conditioned on robot paths. As such, each particle possesses its own set of UKFs.

A. Sampling the Robot Pose

A new robot pose s_t for each particle in S_{t-1} is sampled using both control u_t and measurement y_t according to UPF as follows:

- For the ith particle, set the augmented state $x_t^{(i)a}$ and the augmented covariance matrix $P_t^{(i)a}$ using equations (1) and (2).
- Initialize the mean and covariance of the robot pose for i_{th} particle using equations (3) to (6).
- Sampling a new robot pose for the i_{th} particle with UKF:
  (a) Calculate sigma points $\chi_t^{(i)a}$and weights W using equations (7) to (11).
  (b) Propagate sigma points into the future (pose predict) using equations (12) to (16).
  (c) Incorporate the new observation (pose update) using equations (17) to (21).

B. Updating the Observed Landmark Position Estimate

For the ith particle, every landmark estimate is updated using the following update equations:

- For the ith particle, set the augmented state $x_t^{(i)a} = [(x_t^i)^T (\varepsilon_t^i)^T]$ and the augmented covariance matrix $P_t^{(i)a}$ by incorporating the observation noises as follows:

$$P_t^{(i)a} = \begin{bmatrix} P_t^i & 0 \\ 0 & R_t^i \end{bmatrix}$$

  where $(x_t^i)^T$ is one landmark position estimate, $(\varepsilon_t^i)^T$ is the observation noise, P_t^i is the original covariance and R_t^i is the observation noise covariance.
- Initialize the mean and covariance of the robot pose for ith particle using equations (3) to (5) and (22).

$$P_0^{(i)a} = \begin{bmatrix} P_0^i & 0 \\ 0 & R_0 \end{bmatrix} \tag{22}$$

- Landmark position estimate prediction:
  (a) Calculate sigma points $\chi_t^{(i)a}$and weights W using equations (7) to (11).
  (b) Calculate prediction using equations (15), (16), (23) and (24).

$$\bar{x}^i_{t|t-1} = x^i_{t-1} \tag{23}$$

$$P^i_{t|t-1} = P^i_{t-1} \tag{24}$$

  (c) Estimate update using equations (17) to (21).

C. The Pseudo Code of UPF-UKF SLAM Framework

The UPF-UKF SLAM frame work can be summarized using the pseudo-code as follows:

for $i = 1$ to M **do**
  Get observations at time t
  Do data association
  // processing N_t observed landmarks
  for $j = 1$ to N_t **do**
    if it is a new feature **then**
      Add the new feature into the map
    else
      1. Sampling a new pose for the ith particle and calculating the sample weight
      2. Updating the landmark position estimate
    end if
  end for
  Handle unobserved landmarks
end for
Resample to obtain a new set of M particles.

where M is the number of particles and N_t is the number of landmarks.

V. EXPERIMENTS

In order to verify the performance of the proposed framework, experiments were conducted using simulation data with 47 visual landmarks and known data association. The simulator was developed based on the work in [13]. The exploration area of the robot is 100 meters wide and 120 meters long, and the visual landmarks are randomly located in the area. The robot moves at a speed of $3\ m/s$ and with a maximum steering angle of $30°$. The motion noises are $0.3\ m/s$ for the velocity and $3°$ for the steering angle. The observation noise is set at 0.5 pixels. The robot will travel once along the planned loop.

As an improvement on PF based algorithms, a proposal distribution can be moved towards the real pose distribution by using EKF to incorporate measurements. However, it is assumed that the posterior pose distribution is Gaussian and consequently linearization inaccuracies will be introduced. In the real world, many robot systems are disturbed by non-Gaussian noise and many nonlinear systems can not be approximated well by linearization. Then if EKF is used to obtain the proposal distribution of the robot pose, sampled robot poses will fall into areas of low measurement likelihood. In this situation, the algorithms, which use EKF to calculate the proposal distribution, will be unable to converge when the set of particles is small. Although these algorithms can possibly converge with a large set of particles, the obtained results are usually poor. To prove the above statement, two sets of experiments were conducted. First, both FastSLAM 2.0 and UPF-UKF Framework were implemented on a highly nonlinear system disturbed by Gaussian noise and the experiment was repeated with the number of particles varying from one to 100. Second, both methods were implemented on a system that can be approximated properly by linearization and was disturbed by heavy-tailed noise. In order to verify the efficiency of the proposed framework, the runtime of both methods were compared. There are two reasons for exclusively comparing the proposed framework with FastSLAM 2.0. First, as discussed in [12], the FastSLAM family of algorithms outperformed other popular SLAM algorithms. Second, FastSLAM 2.0 significantly improved the results obtained from FastSLAM 1.0 as shown in [12].

A. Motion Model and Observation Model

The motion model and the observation model have a critical impact on the performance of SLAM algorithms. In our experiments, the assumption of the motion model is that the velocity at time $t+1$ is equal to the velocity at time t plus an arbitrary random noise. The motion model of the robot can be described as follows:

$$\begin{bmatrix} x_{r1(t)} \\ x_{r2(t)} \\ x_{r3(t)} \end{bmatrix} = \begin{bmatrix} x_{r1(t-1)} + l \cdot cos(g + x_{r3(t-1)}) \\ x_{r2(t-1)} + l \cdot sin(g + x_{r3(t-1)}) \\ x_{r3(t-1)} + l \cdot sin(g)/WB \end{bmatrix} \quad (25)$$

where x_r is the robot pose, $l = \bar{v} \cdot dt$ is the translation during one time step ($\bar{v}$ is the mean of the velocity and dt is the time duration), g is the steer angle during one time step and WB is the wheel base.

Cameras are used as observation sensors in our experiments. Instead of using a simple pinhole model, we chose to build a complete and accurate camera model to handle different types of cameras, especially cheap cameras which contain large image distortion. Our observation model can be described as follows:

$$u = f_1 \cdot (x_{d1} + \alpha \cdot x_{d2}) + c_1 \quad (26)$$

$$v = f_2 \cdot x_{d2} + c_2 \quad (27)$$

where (u, v) are the image coordinates, f_1 and f_2 are focal lengths, c_1 and c_2 are the principal points, α is the skew coefficient, and x_{d1} and x_{d2} can be calculated as follows:

$$x_{d1} = (1 + k_1 \cdot r^2 + k_2 \cdot r^4) \cdot \frac{X_c}{Z_c} \quad (28)$$

$$x_{d2} = (1 + k_1 \cdot r^2 + k_2 \cdot r^4) \cdot \frac{Y_c}{Z_c} \quad (29)$$

where k_1 and k_2 are image radial distortion coefficients, r and (X_c, Y_c, Z_c) are defined as:

$$r^2 = \frac{X_c^2 + Y_c^2}{Z_c^2} \quad (30)$$

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x_{r1} \\ r_{21} & r_{22} & r_{23} & x_{r2} \\ r_{31} & r_{32} & r_{33} & x_{r3} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} x_{f1} \\ x_{f2} \\ x_{f3} \\ 1 \end{bmatrix} \quad (31)$$

where
$r_{11} = c\phi c\psi,\ r_{12} = s\theta s\phi c\psi - c\theta s\psi,\ r_{13} = c\theta s\phi c\psi + s\theta s\psi,$
$r_{21} = c\phi s\psi,\ r_{22} = s\theta s\phi s\psi + c\theta c\psi,\ r_{23} = c\theta s\phi s\psi - s\theta c\psi,$
$r_{31} = -s\phi,\ r_{32} = s\theta c\phi,\ r_{33} = c\theta c\phi.$

where $c = cos$, $s = sin$, and θ, ϕ and ψ are rotation angles with respect to x, y and z axes. (x_{f1}, x_{f2}, x_{f3}) is a landmark position and $[x_{r1}, x_{r2}, x_{r3}]$ is a robot pose. If a robot moves on a plane, we have $x_{r2} = 0$, $\theta = 0$ and $\psi = 0$.

The model described above is a highly non-linear system and was used in the first set of experiments. In the second set of experiments, a regular system was represented by a simple pinhole camera model by setting α, k_1 and k_2 in equations (26), (28) and (29) to zeros.

B. Experimental Results

Fig. 1 to Fig. 4 show the estimated robot path and the landmark positions obtained respectively from FastSLAM 2.0 and UPF-UKF Framework in two sets of experiments. It is clear that in both sets of experiments, UPF-UKF Framework generated more accurate results than FastSLAM 2.0 did. When using 100 particles, both algorithms could converge. However, the results, obtained by using small number of particles, showed that FastSLAM 2.0 diverged.

The accuracy of the framework was verified by comparing the RMS errors obtained from FastSLAM 2.0 and UPF-UKF Framework. The experiment was repeated while the number of particles was varied. A comparison of the results is shown in Fig. 5. It is clear that the errors associated with each method drops sharply when the number of particles is greater than 10. It can be seen that the proposed framework constantly outperforms the FastSLAM 2.0. In [12], the authors claimed that FastSLAM performs equally well for varying number of particles. However, the conclusion does not hold in our experiment. The reason for this is that the linearized pose sampling and feature updating methods are not capable of handling highly non-linear observation models.

Fig. 6 shows the execution time for FastSLAM 2.0 and UPF-UKF Framework running with the number of particles varying from one to 100. We can see that though the runtime of UPF-UKF Framework is longer, they are still on the same order of magnitude. Therefore, it is clear that the proposed UPF-UKF Framework outperforms FastSLAM 2.0 when the observation system is highly non-linear and/or the noise is non-Gaussian while the runtime stays at the same order of magnitude.

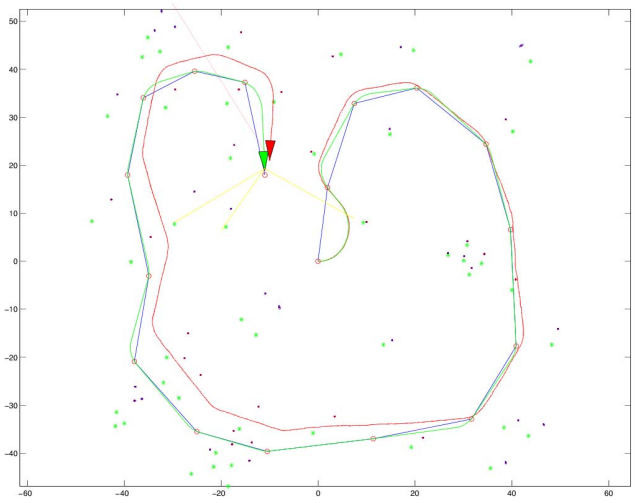

Fig. 1. Estimated robot path and landmark positions using FastSLAM 2.0 with the highly nonlinear system disturbed by Gaussian noise. The green, blue and red paths are the planned, the real and the estimated robot paths. The green and red dots are real and estimated landmark positions.

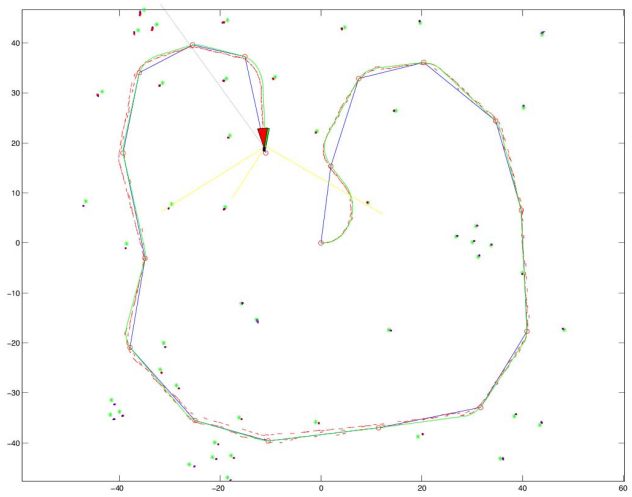

Fig. 2. Estimated robot path and landmark positions using UPF-UKF Framework with the highly nonlinear system disturbed by Gaussian noise. The green, blue and red paths are the planned, the real and the estimated robot paths. The green and red dots are real and estimated landmark positions.

VI. CONCLUSIONS

In this paper, a UPF-UKF SLAM Framework was proposed. A UPF is used to estimate the robot's poses and the UKFs are used to update the landmark positions. When the observation model is highly non-linear or the noises are non-Gaussian, UPFs can more accurately estimate the robot poses than generic PFs or PFs, which use a linearized method to calculate a proposal distribution. Because of the advantages discussed previously, when the observation model is highly non-linear, the UKF can more accurately update landmarks than the widely used EKF. In addition, the calculation of the Jacobian for both the motion model and the observation model is avoided in the proposed framework. From the discussion of the experimental results, it is clear that the UPF-UKF Framework outperforms other popular SLAM algorithms while maintaining the same high level of efficiency.

REFERENCES

[1] Simon J. Julier and Jeffrey K. Uhlmann, "Unscented filtering and nonlinear estimation," Proceedings of the IEEE, Vol. 92, No. 3, March 2004, pp401-422.

[2] Simon J. Julier and Jeffrey K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," Proceedings of Eroses: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Multi Sensor Fusion, Tracking and Resource Management II, SPIE, 1997.

[3] G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localisation and map building (SLAM) problem," IEEE Transactions of Robotics and Automation, vol. 17, no. 3, pp. 229C241, 2001.

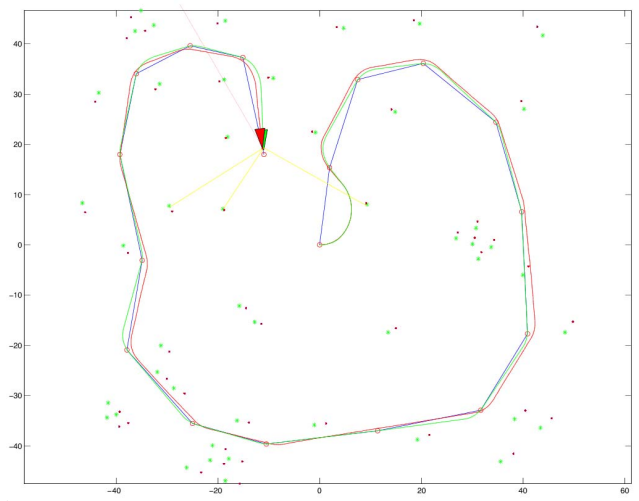

Fig. 3. Estimated robot path and landmark positions using FastSLAM 2.0 with the regular system disturbed by heavy-tailed noise. The green, blue and red paths are the planned, the real and the estimated robot paths. The green and red dots are real and estimated landmark positions.

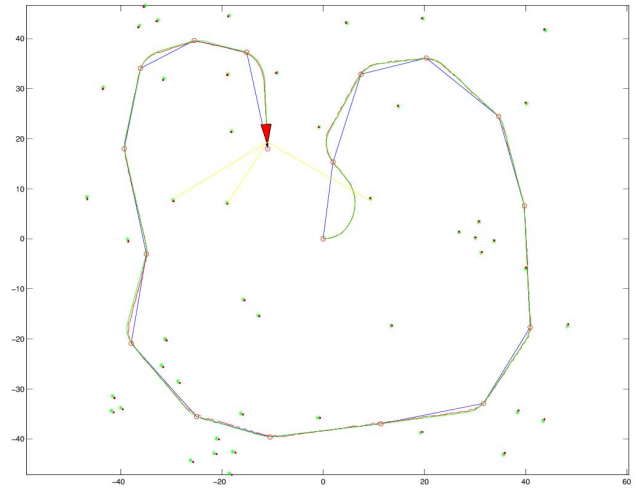

Fig. 4. Estimated robot path and landmark positions using UPF-UKF Framework with the regular system disturbed by heavy-tailed noise. The green, blue and red paths are the planned, the real and the estimated robot paths. The green and red dots are real and estimated landmark positions.

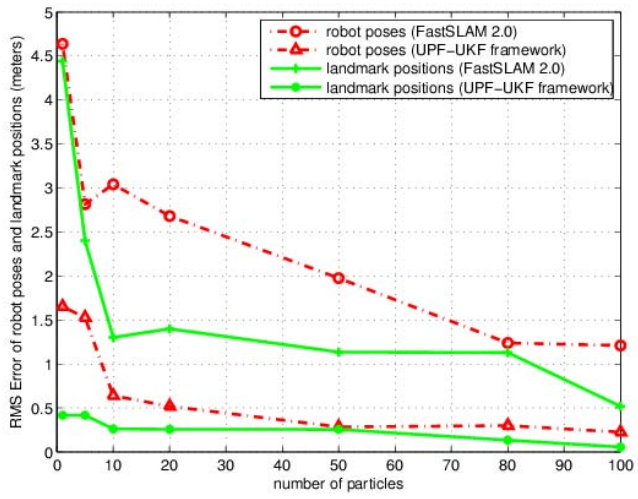

Fig. 5. A comparison of RMS error of robot poses and landmark positions using UPF-UKF Framework and FastSLAM 2.0 with various number of particles.

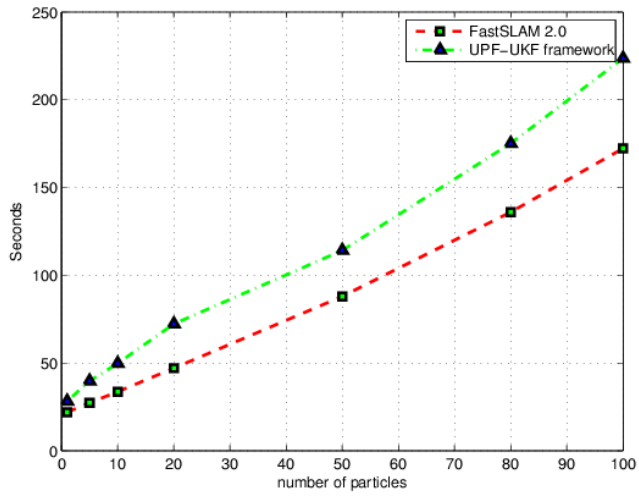

Fig. 6. The run time of UPF-UKF Framework and FastSLAM 2.0 with various number of particles.

[4] J. Leonard, J.D. Tardos, S. Thrun, and H. Choset, editors, Workshop Notes of the ICRA Workshop on Concurrent Mapping and Localization for Autonomous Mobile Robots (W4). ICRA Conference, Washington, DC, 2002.

[5] P. Moutarlier and R. Chatila, "Stochastic multisensory data fusion for mobile robot location and environment modeling," in 5th Int. Symposium on Robotics Research, Tokyo, 1989.

[6] R.C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," International Journal of Robotics Research, 5(4):56 68, 1986.

[7] J.J. Leonard and H.J.S. Feder, "A computationally efficient method for large-scale concurrent mapping and localization," In J. Hollerbach and D. Koditschek, editors, Proceedings of the Ninth International Symposium on Robotics Research, Salt Lake City, Utah, 1999.

[8] S. Thrun, D. Koller, Z. Ghahramani, H. Durrant-Whyte, and A.Y. Ng, "Simultaneous mapping and localization with sparse extended information filters," In J.-D. Boissonnat, J. Burdick, K. Goldberg, and S. Hutchinson, editors, Proceedings of the Fifth International Workshop on Algorithmic Foundations of Robotics, Nice, France, 2002.

[9] M. Montemerlo and S. Thrun, "Simultaneous localization and mapping with unknown data association using FastSLAM," In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2003.

[10] K. Murphy, "Bayesian map learning in dynamic environments," In Advances in Neural Information Processing Systems (NIPS). MIT Press, 1999.

[11] Rudolph van der Merwe, Arnaud Doucet, Nando de Freitast and Eric Wan, "The unscented particle filter," Technical Report CUED/F-INFENG/TR 380, Cambridge University Engineering Department. Available at : www.cs.ubc.ca/ nando/papers/upf.ps.gz

[12] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto and E. Nebot, "FastSLAM: An Efficient Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association," available at: robots.stanford.edu/papers/Thrun03g.pdf

[13] Tim Bailey, "Source code for SLAM simulations," available at: www.acfr.usyd.edu.au/homepages/academic/tbailey/software