

Full Paper

An Improved Particle Filter for Target Tracking in Sensor Systems

Xue Wang *, **Sheng Wang [†]** and **Jun-Jie Ma [‡]**

State Key Laboratory of Precision Measurement Technology and Instrument, Tsinghua University, Beijing 100084, P. R. China; E-mails: [†] wang_sheng00@mails.tsinghua.edu.cn; [‡] mjj@mails.tsinghua.edu.cn

* Author to whom correspondence should be addressed; E-mail: wangxue@mail.tsinghua.edu.cn

Received: 23 December 2006 / Accepted: 27 January 2007 / Published: 29 January 2007

Abstract: Sensor systems are not always equipped with the ability to track targets. Sudden maneuvers of a target can have a great impact on the sensor system, which will increase the miss rate and rate of false target detection. The use of the generic particle filter (PF) algorithm is well known for target tracking, but it can not overcome the degeneracy of particles and cumulation of estimation errors. In this paper, we propose an improved PF algorithm called PF-RBF. This algorithm uses the radial-basis function network (RBFN) in the sampling step for dynamically constructing the process model from observations and updating the value of each particle. With the RBFN sampling step, PF-RBF can give an accurate proposal distribution and maintain the convergence of a sensor system. Simulation results verify that PF-RBF performs better than the Unscented Kalman Filter (UKF), PF and Unscented Particle Filter (UPF) in both robustness and accuracy whether the observation model used for the sensor system is linear or nonlinear. Moreover, the intrinsic property of PF-RBF determines that, when the particle number exceeds a certain amount, the execution time of PF-RBF is less than UPF. This makes PF-RBF a better candidate for the sensor systems which need many particles for target tracking.

Keywords: Particle filter, radial-basis function network, target tracking, sensor system

1. Introduction

For surveillance, a sensor system is installed to search for targets and provide reliable detection within the given region. The sensor system can measure the range and bearing of the targets, but it can not track them. In spite of the recent advances in sensor technology, there are no devices that can detect the manned maneuvers of a tracked target in surveillance and guidance systems [1]. Filtering is used for estimating and tracking the state of a target as a set of observations becomes available online [2]. When a target moves, acceleration may be unexpected, varying over time and follow an unknown profile, which has great impact on the sensor system. Even a short-term acceleration will cause an error in the measurement sequence and result in divergence, if compensations are not used in time.

For implementing target tracking in a sensor system, the extended Kalman filter (EKF) was introduced [3], but because EKF only uses the first order terms of the Taylor series expansion of nonlinear functions, it often introduces large errors in the estimated statistics of the posterior distributions of the states. This is especially evident when the models are highly nonlinear and the local linearity assumption breaks down [2]. In these cases the Unscented Kalman filter (UKF) [4] with true nonlinear models was proposed. Unlike EKF, UKF uses true nonlinear models and instead approximates the distribution of the state random variable. In UKF, the state distribution is still represented by a Gaussian random variable, but it is specified using a minimal set of deterministically chosen sample points. These sample points completely capture the true mean and covariance of the Gaussian random variable, and when propagated through the true nonlinear system, captures the posterior mean and covariance accurately to the 2nd order for any nonlinearity, with errors only introduced in the 3rd and higher orders. However, UKF is limited in that it does not apply to general non-Gaussian distributions [2]. Then generic PF [5] was presented for handling multimodal probability density functions [6] and solving nonlinear non-Gaussian problems [7], which are widely adopted in maneuvering target tracking [15,16,17,18]. However, degeneracy will limit the ability of generic PF to search for lower minima in other regions of the error surface [2]. If the process noise increases, the estimation error will accumulate, which will lead to the divergence of the sensor system [1]. To solve this problem, the improved particle filters have always used various methods to propagate the mean and covariance of the Gaussian approximation to the state distribution, such as unscented particle filter (UPF) which resulted from using a UKF and Markov chain Monte Carlo (MCMC) step within a particle filter framework, but they also increased the execution time sharply [2]. They are computationally intensive and difficult to implement in real time in a sensor system.

This paper proposes a PF-RBF algorithm for target tracking in sensor systems. It is an improved PF algorithm which uses the radial-basis function network (RBFN) in sampling. According to the observations, PF-RBF uses RBFN to approximate the moving trajectory, construct the process model, perform sampling and decrease the cumulated effect of errors. We compare PF-RBF with UKF, PF and UPF. The results show that PF-RBF can track targets effectively, especially when the observation model of the sensor system is nonlinear, and it also displays good real time performance.

The remainder of this paper is organized as follows: target motion and observation model of sensor systems are introduced in section 2. The proposed PF-RBF algorithm is presented in section 3. The experimental results are described in section 4. Finally, the conclusions are given in section 5.

2. Target Motion and Observation Model of Sensor Systems

In a sensor system, the measurement sequences are in polar or Cartesian coordinates, but the target dynamics are best described in Cartesian coordinates, so we assume that measurement sequences described in Cartesian coordinates are available. The target tracking is always formulated as a dynamic state space model. The general state space model can be broken down into a state transition and state measurement model:

$$x_k = f(x_{k-1}, u_k) \quad y_k = h(x_k, r_k) \quad (1)$$

where k is the time index. The state variable x_k which refers to the position and velocity of the target is propagated by the process model f over time. The observation model h is used to map the state variable x_k to the corresponding observation y_k ; u_k and r_k are the process noise and the observation noise respectively. It is assumed that u_k and r_k are white Gaussian noise with zero means and uncorrelated with each other.

It is always assumed that the states follow a first order Markov process and the observations are assumed to be independent given the states

$$p(x_k | x_{1:k-1}) \quad (2)$$

$$p(y_k | x_k) \quad (3)$$

In target tracking, the posterior density $p(x_k | y_{1:k})$, where $y_{1:k} = \{y_1, y_2, \dots, y_k\}$, constitutes the complete solution to the sequential estimation problem [2]. So the estimation accuracy of posterior density is used as the metric in tracking. Because of the measured data and computational loading, most sensor systems use a position and velocity two-state process model [2], a maneuver-driving input model [1, 8] or use two process models and switch between them in the system [9]. However, these models are only appropriate for one of two situations, the presence or absence of target maneuvers. For robust target tracking, a dynamic process model should be constructed.

3. Principle of the PF-RBF Algorithm

The generic PF algorithm is a sequential importance sampling method which is based on Monte Carlo simulation and Bayesian sampling estimation theories. Various PFs all contain three important steps: sampling current value of each particle, evaluation of the recursive important weights and resampling. Recent research has focused almost exclusively on the weighting and resampling for improving the tracking accuracy [10], [11], while PF-RBF algorithm focuses on the sampling step.

3.1. Generic Particle Filter

In Bayesian sampling estimation theory, the posterior density $p(x_k | y_{1:k})$ can be inferred from prior density $p(x_k | y_{1:k-1})$

$$p(x_k | y_{1:k}) = \frac{p(y_k | x_k) p(x_k | y_{1:k-1})}{p(y_k | y_{1:k-1})} \quad (4)$$

where

$$p(y_k | y_{1:k-1}) = \int p(y_k | x_k) p(x_k | y_{1:k-1}) dx_k \quad (5)$$

and

$$p(x_k | y_{1:k-1}) = \int p(x_k | x_{k-1}) p(x_{k-1} | y_{1:k-1}) dx_{k-1} \quad (6)$$

Then PF uses the Monte Carlo simulation method to approximate the posterior density by N particles with the associated weight

$$p(x_{k-1} | y_{1:k-1}) \approx \sum_{i=1}^N \omega_{k-1}^i \delta(x_{k-1} - x_{k-1}^i) \quad (7)$$

For solving the difficulty of sampling from the posterior density function, the sequential importance sampling method is used, which samples from a known, easy-to-sample, proposal distribution $q(x_{0:k} | y_{1:k})$, where $x_{0:k}$ is the historical state variables and $y_{1:k}$ is the corresponding observation.

The recursive estimate for the importance weights of particle i can be derived as follows:

$$\omega_k^i = \omega_{k-1}^i \frac{p(y_k | x_k) p(x_k | x_{k-1})}{q(x_k | x_{0:k-1}, y_{1:k})} \quad (8)$$

Then the estimated state can be approximated by

$$\hat{x}_k \approx \sum_{i=1}^N \omega_k^i x_k^i \quad (9)$$

3.2. Trajectory Approximation with RBFN

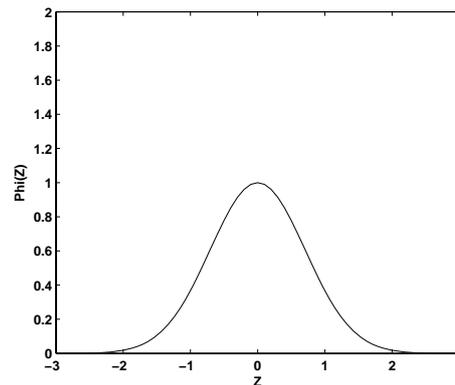
Radial basis functions are a special class of function. Their responses decrease (or increase) monotonically with the distance from a central point [12]. Centre, distance scale and precise shape of the radial function are parameters of the model, all fixed if it is linear [19]. In principle, they could be employed in any sort of linear or nonlinear model and single-layer or multi-layer network. RBFN is a three-layer feed-forward neural network which is embedded with several radial-basis functions. Such a network is characterized by an input layer, a single layer of nonlinear processing neurons, and an output layer. The output of RBFN [19] is calculated as:

$$y = \sum_{j=1}^M \tau_j \phi_j(\|z - c_j\|_2) \quad (10)$$

where z is an input vector, ϕ_j is a basis function, $\|\cdot\|_2$ denotes the Euclidean norm, τ_j is the weight in the output layer, M is the number of neurons in the hidden layer, and c_j is the center of RBF in the input vector space. The functional form of ϕ_j is always assumed as follows:

$$\phi(z) = \exp(-z^2 / \sigma^2) \quad (11)$$

Figure 1 illustrates the shape of function $\phi(z)$ with $\sigma = 1$.

Figure 1. The shape of function $\phi(z)$ with $\sigma = 1$.

RBFNs present good approximation properties. The RBFN family is broad enough to uniformly approximate any continuous function on a compact set. For any continuous input-output function $f(z)$ there is an RBFN with a set of centers $\{t_i\}_{i=1}^M$ and a common width $\sigma > 0$ such that the input-output mapping function $F(z)$ realized by the RBFN is close to $f(z)$ in the L_p norm, $p \in [1, \infty]$ [12]. Here the input-output mapping is represented by

$$F(z) = \sum_{i=1}^M \tau_i G\left(\frac{z-t_i}{\sigma}\right) \quad (12)$$

where $\sigma > 0$, $\omega_j \in R$, and G is a continuous integrable bounded function in a subset $R^{m_0} \in R$ and

$$\int_{R^{m_0}} G(z) dz \neq 0 \quad (13)$$

Because the trajectory of target is a typical continuous function, RBFN can approximate it and construct the dynamic process model for state estimation in PF algorithm well. In PF-RBF, we use previous observations and current prediction to train RBFN; the details will be discussed below.

3.3. Improved PF Algorithm Combined with RBFN

The process model of the generic PF algorithm always uses a single motion model which is static and cannot offer a dynamic and consistent approximation of state variables for target tracking [1,2,8]. The improved PF algorithm based on an interacting multi-model (IMM) filter [13] can solve it, which allows for several modes combined with a weighted estimate [14], but the state variable must be estimated and updated in each time step for each model separately, so the execution time will increase sharply.

The objective of our algorithm is to perform a robust and accurate approximation of state variables and decrease the execution time. Our algorithm contains five steps: (1) constructing RBFN based on previous observations $y_{0:k-1}$ and current prediction \bar{x}_k for approximating the trajectory of target. It should be noted that the observations $y_{0:k-1}$ do not equal to the states of target, which just have relationship to the state variable $x_{0:k-1}$ according to the observation model h . So, for training the RBFN, the state variable $x_{0:k-1}$ should be inferred from observations $y_{0:k-1}$. Because of the observation noise, we can only get the approximate state variable $\bar{x}_{0:k-1}$ instead of $x_{0:k-1}$; (2) sampling new value of each particle based on RBFN; (3) evaluating the recursive importance weights; (4) resampling; (5) output. The pseudo-code for our algorithm is outlined in Algorithm 1:

Algorithm 1.

Algorithm 1

1. Initialization: $k = 0$

For $i = 1, \dots, N$

- $x_0^i \sim \text{sample}(p(x_0))$ / draw the states from the prior

2. **For** $k = 1, 2, \dots$

(a) Constructing RBFN

- Use the previous observations $y_{0:k-1}$ to infer the previous approximate state variable $\bar{x}_{0:k-1}$;
- Predict the current state by kinematic theory with the given time interval T :

$$\bar{x}_k = \bar{x}_{k-1} + \bar{v}_{k-1}T + 0.5\bar{a}_{k-1}T^2 \quad (14)$$

$$\bar{v}_{k-1} = \frac{\bar{x}_{k-1} - \bar{x}_{k-2}}{T} \quad (15)$$

$$\bar{a}_{k-1} = \bar{v}_{k-1} - \bar{v}_{k-2} = \frac{\bar{x}_{k-1} + \bar{x}_{k-3} - 2\bar{x}_{k-2}}{T} \quad (16)$$

- Construct RBFN with the previous real state of target $\bar{x}_{0:k-1}$ and current state prediction \bar{x}_k .

(b) Sampling

For $i = 1, \dots, N$

- Sample $\hat{x}_k^i \sim q(x_k | x_{0:k-1}^i, y_{1:k})$ by the constructed RBFN

$$\hat{x}_k^i = \sum_{j=1}^M \tau_j \phi_j(\|\hat{x}_{0:k-1}^i - c_j\|_2)$$

(c) Evaluating

For $i = 1, \dots, N$

- Evaluate the recursive importance weights by (8)

For $i = 1, \dots, N$

- Normalize the importance weights

$$\tilde{\omega}_k^i = \frac{\omega_k^i}{\sum_{j=1}^N \omega_k^j} \quad (17)$$

(d) Resampling

- Multiply / suppress samples \hat{x}_k^i with high / low importance weights $\tilde{\omega}_k^i$, respectively, to obtain N random samples $x_{0:k}^i$ approximately distributed according to $p(x_{0:k}^i | y_{1:k})$.

(e) Output

- Output the estimated state

$$\hat{x}_k = \sum_{i=1}^N \omega_k^i \hat{x}_k^i$$

- Set $\omega_k^i = \tilde{\omega}_k^i = \frac{1}{N}$, $x_k^i = \hat{x}_k^i$ $i = 1, \dots, N$
-

As illustrated in Algorithm 1, in the PF-RBF algorithm, the RBFN is dynamically trained as the process model for the sampling step. For approximating the state of the target, we use the previous

observations and current prediction based on observations as the samples to train the RBFN. Then, each particle uses the estimated state sequence $\hat{x}_{0:k-1}^i$ as input vectors to sample the new value of the state variable \hat{x}_k^i . Then PF-RBF continues the process of evaluating and normalizing importance weights, resampling, and output as generic PF algorithm.

Because RBFN is trained with the observations and prediction based on observations, it will not be impacted by the posterior distribution error of each particle, and can approximate the real trajectory of target effectively. With the guidance of RBFN, each particle can estimate the probability distribution more effectively and keep the multimodality at the same time.

Although the computation complexity of RBFN is $O(n^3)$ [20], where n is the number of training, at each time instant there is only one training process and the time is irrelative with the particle number, so the change rate of execution time of PF-RBF should nearly equals to PF. Furthermore, because the randomness of the movement of target, we can just adopt the nearest m samples of approximate state variable $\bar{x}_{k-m:k-1}$ to train the RBFN, which can significantly reduce the computation complexity.

4. Experimental Results

4.1. Experiment Setup

The estimation improvement obtained by PF-RBF algorithm is illustrated in this section. The UKF is more suitable than the EKF for proposal distribution generation within the particle filter framework [6], so we only compare the state tracking results of UKF, PF, UPF and PF-RBF. For simplifying and generalizing the problem, we just performed one-dimension tracking. Two-dimension tracking can be easily derived. The used assumptions and parameters are listed below.

For simulation, a time-series is generated by the following process model

$$x_k = x_{k-1} + v_{k-1}T + \frac{1}{2}a_{k-1}T^2 + u'_k \quad (18)$$

where $x_{k-1} + v_{k-1} \cdot T$ presents the state transition model, $\frac{1}{2}a_{k-1}T^2 + u'_k$ presents the process noise, and T is the time interval. Here, for simulating the circuitry of the maneuvering target, the related factors are given by

$$\begin{aligned} v_{k-1} &= \sin(\beta\pi(k-1)) \\ a_{k-1} &= \dot{v}_{k-1} = \beta\pi \cos(\beta\pi(k-1)) \end{aligned} \quad (19)$$

u'_k is a term of process noise which is drawn from a Gaussian distribution $\mathcal{N}(0,0.1)$. In target tracking, the whole process noise is considered as the degree of maneuvering. $\beta = 0.04$ is a scalar parameter.

For comparing tracking performance, a non-stationary observation model is used as in [2]:

$$y_k = \begin{cases} \phi_1 x_k^2 + r_k & k \leq 30 \\ \phi_2 x_k - 2 + r_k & k > 30 \end{cases} \quad (20)$$

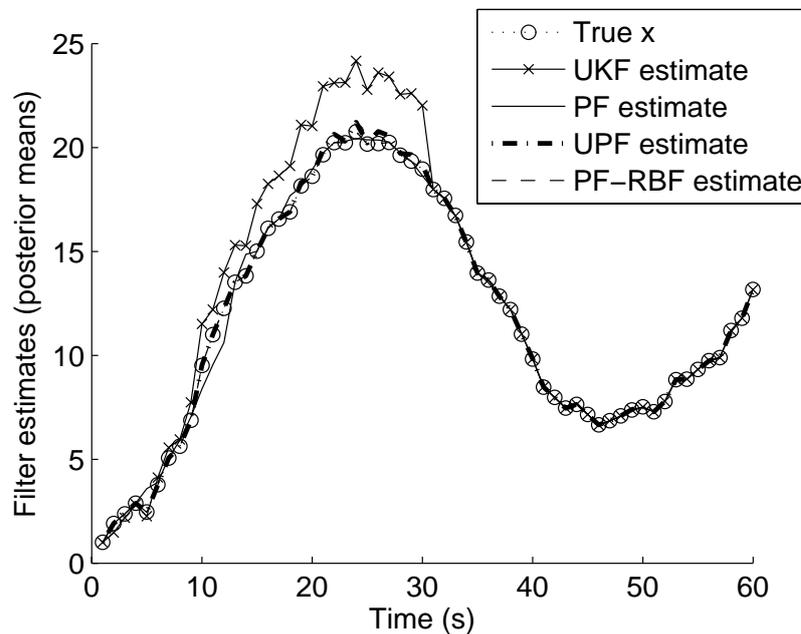
with $\phi_1 = 0.2$ and $\phi_2 = 0.5$. The observation noise, r_k is drawn from a Gaussian distribution $\mathcal{N}(0,0.00001)$, which is the same as the parameter proposed by [2]. Given only the noisy observations y_k , the different filters are used to estimate the underlying clean state sequence x_k for $k = 1, 2, \dots, 60$. The experiment is repeated 100 times with random re-initialization for each run. All of the particle filters use 10 to 200 particles and residual resampling. The point scaling parameter of UKF

is set to $\alpha = 1$, which is used to control the size of the sigma point distribution and should ideally be a small number to avoid sampling non-local effects when the nonlinearities are strong, scaling parameter for higher order terms of Taylor series expansion is set to $\beta = 0$, which is a non-negative weighting term to be used to incorporate knowledge of the higher order moments of the distribution, and sigma point selection scaling parameter is set to $\kappa = 2$, this parameter is used to guarantee positive semi-definiteness of the covariance matrix. Three parameters are optimal for the scalar cases, and they are also adopted in [2]. The number of the samples of approximate state variable $\bar{x}_{k-m:k-1}$ used for training the RBFN is 20. The time interval is 1s. The root-mean-square-errors (RMSEs) of the estimation values are acquired for comparison.

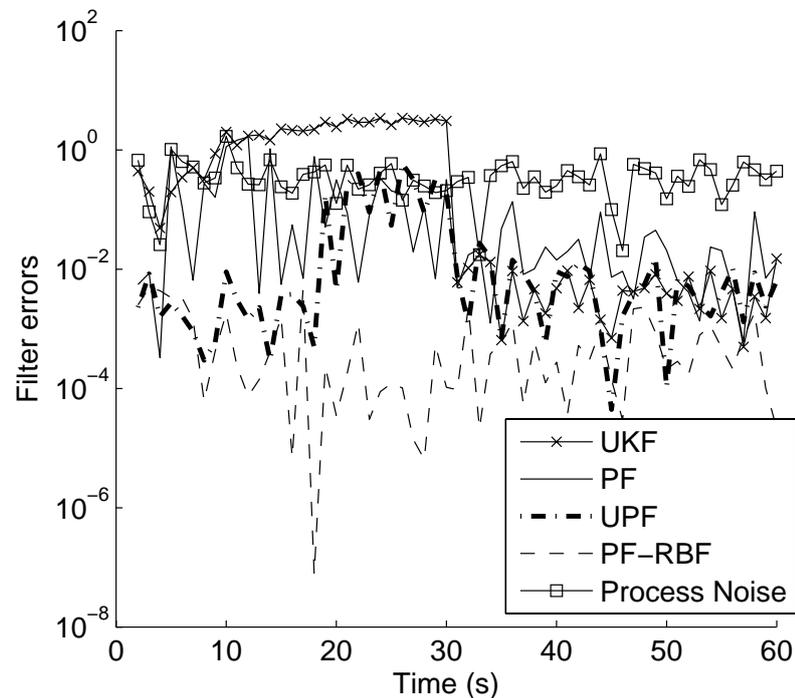
4.2. Tracking results

In Figure 2 the estimated results generated from a single run of the different filters are compared, where the particle number is 200 in each particle filter. Figure 3 shows the corresponding errors of different filters versus the process noises.

Figure 2. Plot of estimated results generated from a single run of the different filters with non-stationary observation model.



As illustrated in Figure 3, the errors generated by all algorithms are strongly correlative with process noise, i.e., the tendencies of the errors are coherent with the tendency of the process noise. The error of UKF algorithm increases sharply because the estimation error cumulates and results in system divergence. Although the errors of estimated states in four algorithms are nearly equal when the observation model is linear ($k > 30$), our algorithm stands out by its lower error performance compared to the other three algorithms when the observation model is nonlinear ($k \leq 30$).

Figure 3. Errors of different filters versus the process noises in a single run.

The RMSEs of PF-RBF are all small whether the observation model is linear or nonlinear. This means that the PF-RBF is robust and effective for both linear and nonlinear observation models. When a maneuver occurs ($20 < k < 30$), the RMSEs of PF-RBF are still smaller than 0.01, while the RMSEs of the other three algorithms increase sharply. This confirms verifies that PF-RBF is also suitable for tracking maneuvering targets.

Then we compared the mean and variance of RMSEs generated by each algorithm with 200 particles over 100 independent runs. As shown in Table 1, the superior performance of the PF-RBF is clearly evident for both linear and nonlinear observation models.

Table 1. Estimated results generated by each algorithm where the mean and variance of RMSEs are calculated over 100 independent runs. The particle number is 200.

Algorithm	Mean of RMSEs	
	Nonlinear	Linear
UKF	0.58461	0.0067026
Generic PF	0.47292	0.1485
UPF	0.085445	0.0080179
PF-RBF	0.0187598	0.0066431

4.3. Execution time and accuracy

In each PF algorithm, the execution time and accuracy are partially determined by the particle number. In this section, we compare the average RMSE and execution time of PF, UPF and PF-RBF. The particle number increases from 10 to 200.

The average RMSE calculated over 100 independent runs, which decreases when the particle number increases, are shown in Figure 4. When the observation model is linear, PF-RBF performs best. And for the nonlinear observation model, the average RMSE of PF-RBF is biggest when the particle number is 10, but the average RMSE of PF-RBF decreases rapidly and become lowest when the particle number is larger than 20. As a whole, PF-RBF has superior performance in target tracking and it is robust for both linear and nonlinear observation models.

Figure 4. The average RMSEs of three algorithms calculated over 100 independent runs with the linear and non-linear observation model.

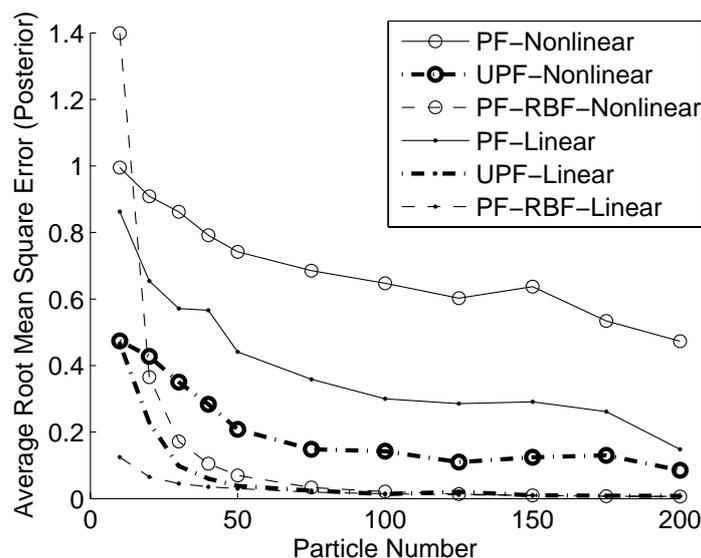
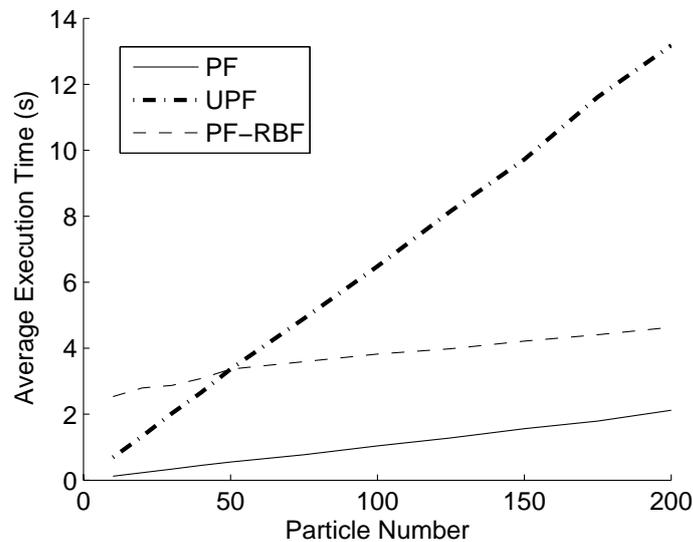


Figure 5 shows that the execution time of three algorithms for 60 step tracking all increase linearly with particle number and UPF takes more time than PF-RBF when the particle number exceeds 50. Although PF-RBF includes the RBFN training step which takes most time, for each time point there is only one training process and the time is independent of the particle number, so the rate of change of execution time of PF-RBF nearly equals the PF one, and because UPF contains UKF and MCMC steps for each particle, the change rate is higher than PF-RBF. As illustrated in Figure 5, the change rate of UPF is about 7 times greater than the one of PF-RBF. This makes the PF-RBF a better candidate for the scenarios which require many particles.

Figure 5. The average execution time of three particle filters for 60 steps tracking over 100 independent runs.



5. Conclusions

In this paper, we focused on target tracking in sensor systems. The PF-RBF algorithm was proposed, which trains RBFN to approximate the trajectory of target and constructs dynamic process model according to the previous observations and current predictions. The trained RBFN is used to perform sampling steps for each particle, instead of the classical process model. With the guidance of RBFN, each particle can give an accurate proposal distribution, the cumulated effect of errors can be decreased, and the sensor system remains convergent even if the target maneuvers. The target tracking experiment results verify that when the observation model is linear, PF-RBF perform better than UKF, PF and UPF, and for a nonlinear observation model, PF-RBF is most robust and accurate, except when the particle number is less than 20. Moreover, the rate of change of execution time of PF-RBF is about seven times less than that of UPF, it this makes the execution time of PF-RBF less than that of UPF when the particle number exceeds a certain number, which in this paper is 50. It should be noted that the critical particle number thresholds depend on the given problem, and may be different in other problems. However, it still implies that PF-RBF is suitable for dealing with sensor systems which need many particles for multi-target tracking.

Acknowledgements

This paper is sponsored National Grand Research 973 Program of China (No. 2006CB303000) and National Natural Science Foundation of China (No. 60673176, No.60373014, No. 50175056).

References and Notes

1. Duh, F.; Lin, C. Tracking a maneuvering target using neural fuzzy network. *IEEE Trans. Sys. Man, Cyber.*, **2004**, *34*, 16-33.
2. Van der Merwe, R.; Doucet, A.; de Freitas, N.; Wan, E. The unscented particle filter. *Technical Report CUED/ F-INPENG/TR 380*; Department of Engineering, Cambridge University: Cambridge, UK, **2000**.
3. Anderson, B. D.; Moore, J. B. *Optimal Filtering*; Prentice Hall: NJ, U.S.A., **1979**.
4. Julier, S.; Uhlmann, J.K. A general method for approximating nonlinear transformations of probability distributions. *Technical Report, Department of Engineering Science*; Oxford University: Oxford, UK, **1996**.
5. Doucet, A. On sequential simulation-based methods for Bayesian filtering. *Technical Report, CUED/F-INFENG/TR 310*; Department of Engineering, Cambridge University: Cambridge, UK, **1998**.
6. Gordon, N.J.; Salmond, D.J.; Smith, A.F.M. Novel approach to non-linear/non-Gaussian Bayesian state estimation. *Proc. Inst. Elect. Eng. F*, **1993**, *140*, 107-113.
7. Doucet, A.; Godsill, S.; Andrieu, C. On sequential Monte Carlo sampling methods for Bayesian filtering. *Stat. Comput.* **2000**, *10*, 197-208.
8. Chan, Y.T.; Hu, A.G.C.; Plant, J.B. A Kalman filter based tracking scheme with input estimation. *IEEE Trans. Aerosp. Electron. Syst.* **1979**, *AES-15*, 237-244.
9. Bar-Shalom, Y.; Birmiwala, K. Variable dimension filter for maneuvering target tracking. *IEEE Trans. Aerosp. Electron. Syst.* **1982**, *AES-18*, 621-629.
10. Hong, S.; Bolić, M.; Djurić, P.M. An efficient fixed-point implementation of residual resampling scheme for high-speed particle filters. *IEEE Sig. Process. Lett.* **2004**, *11*, 482-485.
11. Cheng, C.; Ansari, R. Kernel particle filter for visual tracking. *IEEE Sig. Process. Lett.* **2005**, *12*, 42-45.
12. Haykin, S. *Neural Networks: a Comprehensive Foundation*; Prentice Hall: NJ, U.S.A., **1999**.
13. Yang, N.; Tian, W.; Jin, Z. An interacting multiple model particle filter for maneuvering target location. *Meas. Sci. Tech.* **2006**, *17*, 1307-1311.
14. Bar-Shalom, Y.; Chen, H. IMM estimator with out-of-sequence measurements. *IEEE Trans. Aerosp. Electron. Syst.* **2005**, *41*, 90-98.
15. Angelova, D.; Mihaylova, L. Joint target tracking and classification with particle filtering and mixture Kalman filtering using kinematic radar information. *Dig. Sig. Process.* **2006**, *16*, 180-204.
16. Yu, Y.; Cheng, Q. Particle filters for maneuvering target tracking problem. *Sig. Process.* **2006**, *86*, 195-203.
17. Morelande, M.R.; Challa, S. Manoeuvring target tracking in clutter using particle filters. *IEEE Trans. Aerosp. Electron. Syst.* **2005**, *41*, 252-270.
18. Arulampalam, M.S.; Ristic, B.; Gordon, N.; Mansell, T. Bearings-only tracking of maneuvering targets using particle filters. *EURASIP J. Appl. Sig. Process.* **2004**, *15*, 2351-2365.
19. Mark J. L. Orr. *Introduction to radial basis function networks. Technical Report*; Centre for Cognitive Science, University of Edinburgh: Edinburgh, **1996**.

20. Huang G. B.; Saratchandran P.; Sundararajan S. A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. *IEEE Trans. Neural Net.* **2005**, *16*, 57-67.

© 2007 by MDPI (<http://www.mdpi.org>). Reproduction is permitted for noncommercial purposes.