Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 302170, 7 pages
http://dx.doi.org/10.1155/2013/302170

Research Article

Improved Rao-Blackwellized Particle Filter by Particle Swarm Optimization

Zeng-Shun Zhao,[1,2] Xiang Feng,[1] Yan-yan Lin,[1] Fang Wei,[1] Shi-Ku Wang,[1] Tong-Lu Xiao,[1] Mao-Yong Cao,[1] Zeng-Guang Hou,[3] and Min Tan[3]

[1] *Shandong Province Key Laboratory of Robotics and Intelligent Technology, College of Information and Electrical Engineering, Shandong University of Science and Technology, Qingdao 266590, China*
[2] *School of Control Science and Engineering, Shandong University, Jinan 250061, China*
[3] *State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100090, China*

Correspondence should be addressed to Zeng-Shun Zhao; zhaozengshun@gmail.com

Received 12 April 2013; Revised 22 July 2013; Accepted 30 July 2013

Academic Editor: Debasish Roy

The Rao-Blackwellized particle filter (RBPF) algorithm usually has better performance than the traditional particle filter (PF) by utilizing conditional dependency relationships between parts of the state variables to estimate. By doing so, RBPF could not only improve the estimation precision but also reduce the overall computational complexity. However, the computational burden is still too high for many real-time applications. To improve the efficiency of RBPF, the particle swarm optimization (PSO) is applied to drive all the particles to the regions where their likelihoods are high in the nonlinear area. So only a small number of particles are needed to participate in the required computation. The experimental results demonstrate that this novel algorithm is more efficient than the standard RBPF.

1. Introduction

The system state estimation is widely used in many fields including statistics, economics, statistical signal processing, and engineering, such as satellite navigation, communication, radar tracking, sonar ranging, target tracking, and robot localization. Many of these problems can be written in the form of the so-called dynamic state space model. A general formulation of such a model is provided as follows:

$$\begin{aligned} s_k &= f\left(s_{k-1}, \xi_{k-1}\right), \\ z_k &= h\left(s_k, v_k\right), \end{aligned} \tag{1}$$

where s_k represents the state vector at instant k, z_k is the observation vector of s at instant k, and $f()$ and $h()$ are the transition function and measurement function, respectively. ξ_{k-1} and v_k are the i.i.d. process noise sequence and measurement noise sequence. As we know, the most famous algorithms, such as the Kalman filter (KF) and the hidden Markov models (HMM), are limited to deal with the finite state spaces with linear gaussian models. However, in many applications, we must handle the time sequence estimation problems which abide by complex nonlinear, non-Gaussian distribution. To manage these problems, sequential Monte Carlo methods (SIS) [1], also known variously as bootstrap filtering [2], the condensation algorithm [3], particles filter [4], interacting particle approximations [5, 6], or survival of the fittest [7], have been introduced. In 1969, the SIS was used in the control field by Handschin and Mayne [8], but the early algorithm had a shortcoming, which was denoted by degeneration. The particle filter (PF) has not come to the attention of researchers until Gordon et al. first proposed resampling methods in 1993 [2], which solved the degeneration inherent in the earlier PF algorithms. One of the major drawbacks of PF is that sampling in high-dimensional spaces can be inefficient. If the model contains a substructure with linear equations, subject to Gaussian noise, the dependency structure can be taken advantage of to accelerate

the estimator. This algorithm is known as the marginalized particle filter (MPF), which also had a reputation as the Rao-Blackwellized particle filter (RBPF) [9–16]. It is widely used in the state and parameter estimations of nonlinear dynamic systems [16], especially in the situations where dependencies within the state space can be analytically exploited.

The key idea of RBPF algorithm is to split the state space into two parts one linear state subspace and one nonlinear state subspace. The linear state subspace can be carried out using standard algorithms, such as the Kalman filter, the HMM filter, the junction tree algorithm, or other finite-dimensional optimal filters, and the nonlinear state subspace is implemented by particle filter. Therefore, the working condition where we could utilize the RBPF is the state space, which could be able to divide into two parts linear state and nonlinear state. Thus the dimensionality problem associated with the standard particle filter is alleviated, since the dimension of the state variables estimated by the particle filter is reduced. In this paper, we propose an improved RBPF that is optimized by the particle swarm, moving the particles to the region where has higher likelihood density. Compared with standard RBPF, the number of required particles can be greatly reduced.

2. The Standard Particle Filter

Particle filter, as a suboptimal Bayesian filter, the fundamental idea is to recursively approximate the posterior density $p(s_k \mid z_{1:k})$ by a set of particles $\{s_k^i, i = 1, \ldots, N\}$ with associated weights $\{w_k^i, i = 1, \ldots, N\}$ where N is the number of particles and k denotes the index of current time. In practice, it is inconvenient to sample directly from the posterior density distribution $p(s_k \mid z_{1:k})$ the solution is to sample particles $s_k^i \sim q(\cdot)$, where $q(\cdot)$ is referred to as importance density. Then, a weighted approximation to the posterior $p(s_k \mid z_{1:k})$ is given by

$$p(s_k \mid z_{1:k}) \approx \sum_{i=1}^{N} w_k^i \delta\left(s_k - s_k^i\right), \tag{2}$$

where the normalized weights are defined as

$$w_k^i \propto \frac{p\left(s_k^i \mid z_{1:k}\right)}{q\left(s_k^i \mid z_{1:k}\right)}. \tag{3}$$

The more detailed description of particle filter could be referred to [17].

A pseudocode description of the standard particle filter is given by Pseudocode 1.

There are several serious problems in the particle filter algorithm. The first problem is particle degeneracy. After several updating iterations, only a few particles would have significant important weights. The utmost situation is only one weight of particles almost approximate one, even others all almost to zero. The degeneracy problem means that a large computational effort devoted to updating the particles does not make any sense. The second problem is the particle impoverishment which comes from resampling. The introduction of resampling could resolve the particle degeneracy problem, but at the same time it brings the particle impoverishment. In resampling step, the particles with little weights are cast out, so the diversity of particles could be reduced. In some situations, the posterior distribution may be represented by only N of the same particles, resulting in the accuracy of estimation to be very low. The only way to solve this problem depends on increasing the sample set size. But, if we provide large enough samples covering whole state space to ensure a successful estimation, the computation efficiency of particle filter would decrease tremendously and even could not meet real-time requirement. The third problem is sampling inefficient in high-dimensional spaces. In some situations, the dimension of the state vector is large; the number of particles required increased exponentially with dimension number of the state vector in order to obtain reliable estimates for the states. RBPF, as an important modification of the standard particle, is applicable to a certain class of high-dimensional state space models.

3. Rao-Blackwellized Particle Filter

Rao-Blackwellisation is a technique marginalizing out some of the variables from state vector models, which are related to the Rao-Blackwell formula [16, 18]. If some conditional dependencies relationships between elements of the state vector can be analytically explicit, then it is not necessary to draw samples from the entire state space. Consider a state vector s_k which can be partitioned according to

$$s_k = \begin{bmatrix} s_k^l \\ s_k^n \end{bmatrix}, \tag{4}$$

where s_k^l denotes the linear state variables and s_k^n denotes the nonlinear state variables. A relative general model with the properties discussed above is given by

$$s_{k+1}^n = f_k^n\left(s_k^n\right) + A_k^n\left(s_k^n\right) s_k^l + G_k^n\left(s_k^n\right) \xi_k^n, \tag{5}$$

$$s_{k+1}^l = f_k^l\left(s_k^n\right) + A_k^l\left(s_k^n\right) s_k^l + G_k^l\left(s_k^n\right) \xi_k^l, \tag{6}$$

$$z_k = h_k\left(s_k^n\right) + C_k\left(s_k^n\right) s_k^l + \nu_k, \tag{7}$$

where $f()$ and $h()$ are possible nonlinear functions. Furthermore, the noise of the system state is assumed to be white and Gaussian distributed with

$$\xi_k = \begin{bmatrix} \xi_k^l \\ \xi_k^n \end{bmatrix} \sim N\left(0, Q_k\right), \qquad Q_k = \begin{bmatrix} Q_k^l & Q_k^{ln} \\ \left(Q_k^{ln}\right)^T & Q_k^n \end{bmatrix}. \tag{8}$$

The measurement noise is also assumed to be white and Gaussian distributed according to the following equation:

$$\nu_k \sim N\left(0, R_k\right). \tag{9}$$

Furthermore, s_0 is divided as follows:

$$s_0^l \sim N\left(\overline{s}_0, \overline{P}_0\right), \tag{10}$$

$$s_0^n \sim p\left(s_0^n\right), \tag{11}$$

$\left[\left\{s_k^i, w_k^i\right\}_{i=1}^N\right] = \mathrm{PF}\left[\left\{s_{k-1}^i, w_{k-1}^i\right\}_{i=1}^N, z_k\right]$
FOR = $1:N$
  Draw $s_k^i \sim p\left(s_k \mid s_{k-1}^i\right)$
  Calculate $w_k^i = p\left(z_k \mid s_k^i\right)$
END FOR
Calculate total weight: $t = \mathrm{sum}\left[\left\{w_k^i\right\}_{i=1}^N\right]$
For = $1:N$
  Normalize: $w_k^i = t^{-1} w_k^i$
END FOR
Resample [17]
$\left[\left\{s_k^i, w_k^i, \ldots\right\}_{i=1}^N\right] = \mathrm{RESAMPLE}\left[\left\{s_k^i, w_k^i\right\}_{i=1}^N\right]$

Pseudocode 1: Description of the standard particle filter.

where the formulas (5), (6), and (7) describe a mixed linear/nonlinear system. If the nonlinear state s_k^n is known, the formula (6) describes a linear, Gaussian subsystem. So the linear state s_k^l can be marginalized out from the posterior density $p(s_k \mid z_{1:k})$

$$p\left(s_k^l, s_k^n \mid z_{1:k}\right) = p\left(s_k^l \mid s_k^n, z_{1:k}\right) p\left(s_k^n \mid z_{1:k}\right), \tag{12}$$

where the linear state $p(s_k^l \mid s_k^n, z_{1:k})$ is analytically tractable whose solution can be given by the Kalman filter. Meanwhile, the nonlinear state $p(s_k^n \mid z_{1:k})$ can be estimated using the particle filter. Therefore the posterior density of the whole space can be approximated to

$$\begin{aligned} p\left(s_k^l, s_k^n \mid z_{1:k}\right) &= p\left(s_k^l \mid s_k^n, z_{1:k}\right) p\left(s_k^n \mid z_{1:k}\right) \\ &= \sum_{i=1}^{N} w_k^i \delta\left(s_k^n - s_k^{nj}\right) N\left(s_k^l; s_k^{lj}, P_k^j\right). \end{aligned} \tag{13}$$

The conditional probability density functions for s_k^l and s_{k+1}^l are given by

$$\begin{aligned} p\left(s_k^l \mid s_k^n, z_k\right) &= N\left(s_{k|k}^l, P_{k|k}\right), \\ p\left(s_{k+1}^l \mid s_{k+1}^n, z_k\right) &= N\left(s_{k+1|k}^l, P_{k+1|k}\right), \end{aligned} \tag{14}$$

where

$$\begin{aligned} s_{k|k}^l &= s_{k|k-1}^k + K_k\left(z_k - h_k - C_k s_{k|k-1}\right), \\ P_{k|k} &= P_{k|k-1} - K_k M_k K_k^T, \\ M_k &= C_k P_{k|k-1} C_k^T + R_k, \\ K_k &= P_{k|k-1} C_k^T M_k^{-1}. \end{aligned} \tag{15}$$

The formulation above updates linear state by Kalman filter when the new observation z_k is obtained. After getting the prediction of nonlinear state s_{k+1}^n, the formulas (5) and (6) can be viewed as measurement equation and prediction equation, and $s_{k+1|k}^l$ and $P_{k+1|k}$ can be computed as follows:

$$\begin{aligned} s_{k+1|k}^l &= \overline{A}_k^l s_{k|k}^L + G_k^l\left(Q_k^{\ln}\right)^T \left(G_k^n Q_k^n\right)^{-1} \eta_k \\ &\quad + f_k^l + L_k\left(\eta_k - A_k^n s_{k|k}^l\right), \\ P_{k+1|k} &= \overline{A}_k^l P_{k|k}\left(A_k^k\right)^T + G_k^l Q_k^l \left(G_k^l\right)^T - L_k N_k L_k^T, \\ N_k &= A_k^n P_{k|k}\left(A_k^n\right)^T + G_k^n Q_k^n \left(G_k^n\right)^T, \\ L_k &= \overline{A}_k^l P_{k|k}\left(A_k^n\right)^T N_k^{-1}, \end{aligned} \tag{16}$$

where

$$\begin{aligned} \eta_k &= s_{k+1}^n - f_k^n, \\ \overline{A}_k^l &= A_k^l - G_k^l\left(Q_k^{ln}\right)^T \left(G_k^n Q_k^n\right)^{-1} A_k^n, \\ \overline{Q}_k^l &= Q_k^l - \left(Q_k^{ln}\right)^T \left(Q_k^n\right)^{-1} Q_k^{ln}, \end{aligned} \tag{17}$$

where Q_k^{ln} is covariance matrix of the linear state s_k^l and nonlinear state s_k^n. More references are listed in [14, 15].

The measurement equation and the prediction equation of nonlinear state are given by

$$p\left(z_k \mid s_k^n, \eta_{k-1}\right) = N\left(h_k + C_k s_{k|k-1}^l, C_k P_{k|k-1} C_k^T + R_k\right) \tag{18}$$

$$\begin{aligned} &p\left(s_{k+1}^n \mid s_k^n, \eta_k\right) \\ &\quad = N\left(f_k^n + A_k^n s_{k|k}^l, A_k^n P_{k|k}\left(A_k^n\right)^T + G_k^n Q_k^n \left(G_k^n\right)^T\right). \end{aligned} \tag{19}$$

A pseudocode description of the Rao-Blackwellized particle filter is supplied in Pseudocode 2.

4. Particle Swarm Optimization

The particle swarm optimization, as well known, is a population-based parallel evolutionary computation technique developed by Kennedy and Eberhart [19–23]. The PSO

(1) Initialization
For $i = 1, \ldots, N$, initialize the particles, $s_0^{n,(i)} \sim p(x_0^n)$ and set $\{s_0^{l,(i)}, P_0^{(i)}\} = \{\overline{s}_0^l, \overline{P}_0\}$
(2) FOR $k > 0$
(2.1) PF measurement update
Compute the particle weight $\widetilde{w}_k^i$ using formula (18),
(2.2) normalize the weight

$$w_k^i = \frac{\widetilde{w}_k^i}{\sum_{i=1}^k \widetilde{w}_k^i}$$

(2.3) Particle filter time update and Kalman filter:
(a) Kalman filter measurement update using formula (15)
(b) Particle filter time update:
For $i = 1, \ldots, N$ predict new particles according to (19)
(c) Kalman filter time update using (16)–(17)
(2.4) Set $K := K + 1$ and iterate from step (2.1)

Pseudocode 2: Description of the Rao-Blackwellized particle filter.

technique has its inspiration rooted in observations and simulations of social behavior of animal swarms such as bird flocks searching for corn. PSO algorithm is initialized with a population of random particles which correspond to candidate solutions. Each simple entity is assigned a random velocity according to both its own and the whole flock's flying experiences. After evaluating the fitness function at its current location, the entities called particles are then collectively flown through hyperspace, searching for food. Finding the optimum location for the best objective function can be viewed as analogues of the evolving trajectory of such a swarm behavior.

The PSO has been found to be robust and fast in solving nonlinear, nondifferentiable, and multimodal optimization problems. The PSO algorithm can be described in mathematical formulations as below.

For a random particle swarm, there are m particles in an n dimensional space, denoted $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$, $i = 1, \ldots, m$, and $V_i = (v_{i1}, v_{i2}, \ldots, v_{in})$, $i = 1, \ldots, m$, where X_i and V_i are the position and velocity of the ith particle, respectively. At each step of iterations, two main factors, P_i and G, have been defined which could reflect the objective-fitness value of one given particle or the whole swarm, representing approximation to the expected state; this to say, $P_i = (p_{i1}, p_{i2}, \ldots, p_{in})$ represents the previous best fitness value of ith particle up to current step; $G = (g_1, g_2, \ldots, g_n)$ denotes the best fitness value of all particles in the whole population. After obtaining the two best values, each particle updates its position and velocity according to the following equations:

$$V_i^{t+1} = w \cdot V_i^t + c_1 \cdot r_1 \cdot \left(p_i^t - X_i^t\right) + c_2 \cdot r_2 \cdot \left(G - X_i^t\right), \quad (20)$$

$$X_i^{t+1} = X_i^t + V_i^t. \quad (21)$$

Here, we define r_1 and r_2 as the random numbers derived from the range $[0, 1]$, which are generated according to a uniform probability distribution $U[0, 1]$. w denotes the inertial weight (constriction factor), and it has been proven that using this factor while limiting the maximum velocity to the dynamic range of the position variable on each dimension is a better approach [24]. c_1 and c_2 are positive constants called acceleration coefficients [21–23], which are utilized to control the velocity variable in the velocity updating step of PSO.

5. PSO Optimized Rao-Blackwellized Particle Filter

First, the newest observation is taken into account by defining a PSO fitness function

$$\text{fitness} = \exp\left[-\frac{1}{2R_k}\left(z_{\text{new}} - z_{\text{pred}}\right)^2\right]. \quad (22)$$

Here, R_k is the observation noise covariance, z_{new} is the newest observation, and z_{pred} is the predicted observation. The reason why we select this formula (22) as the fitness function is that its absolute value can describe how well the newest observation could "approximate" or "fit" the predicted observation. The "best value" in Section 4 refers to the maximun value of the objective function which could correspond to the most satisfied state for our given configuration. In this paper, we prefer the value that could well approximate predicted state with lowest error between the predicted state and the real state.

The PSO moves all the particles towards the region where they have the best fitness values. But sometimes the maximum velocity $V_{\max}$ of classic PSO is difficult to determine, which impacts the searching step size. Here we adopt an improved PSO-Gaussian swarm, which updates velocity based on a Gaussian distribution [20].

If all particles are distributed around high likelihood region, then the fitness value of each particle would be high. On the other side, if the best previous value of P_i and the global best value of G are both very low, it means that particles are not distributed around high likelihood area. Hence, via PSO mechanism, each particle updates the velocity and positions of according to (20) and (21). But it must be noticed that, inside the formula (20), the terms r_1 and r_2 are generated according to the Gaussian probability distribution, that is, $N(0, 1)$.

(1) Initialization

For $i = 1, \ldots, N$, initialize the particles, $x_0^{n,(i)} \sim p(x_0^n)$ and set $\{x_0^{l,(i)}, P_0^{(i)}\} = \{\overline{x}_0^l, \overline{P}_0\}$

(2) FOR $k > 0$

(2.1) PF measurement update

$\left\lfloor \{x_k^{i*}\}_{i=1}^N \right\rfloor = \text{PSO}\left\lfloor \{x_k^i\}_{i=1}^N, z_k \right\rfloor$

Compute the particle weight $\widetilde{w}_k^i$ using formula (18),

(2.2) normalize the weight

$$w_k^i = \frac{\widetilde{w}_k^i}{\sum_{i=1}^k \widetilde{w}_k^i}$$

(2.3) Particle filter time update and Kalman filter:

(a) Kalman filter measurement update using formula (15)

(b) Particle filter time update:

For $i = 1, \ldots, N$ predict new particles according to (19)

(c) Kalman filter time update using (16)–(17)

(2.4) Set $K := K + 1$ and iterate from step (2.1)

Pseudocode 3: The PSO optimized Rao-Blackwellized particle filter.

Next, each particle's weight would be updated and normalized by (9) and (10)

$$w_t^i = w_{t-1}^i p\left(y_t \mid x_t^i\right),$$
$$w_t^i = \frac{w_t^i}{\sum_{i=1}^N w_t^i}. \tag{23}$$

Then using resampling step to select and replicate the particle with large weights,

$$\left\{x_t^i, \frac{1}{N}\right\}_{i=1}^N = \left\{x_t^i, w_t^i\right\}_{i=1}^N. \tag{24}$$

A pseudocode description of the PSO optimizes Rao-Blackwellized particle filter which is given by Pseudocode 3.

6. Experimental Results

We choose the classical bearing only tracking mode to test the novel algorithm. The basic frequency of the computer is 2.6 GHZ; EMS memory is 1 G. Hard disk capability is 80 G. The simulation software is MATLAB 7.0.

The dynamic system state equation of bearing only tracking mode is given by

$$s_k = Fs_{k-1} + \Gamma\xi_{k-1}, \quad k = 1, 2, \ldots, K, \tag{25}$$

where $F = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, $\Gamma = \begin{bmatrix} 0.5 & 0 \\ 1 & 0 \\ 0 & 0.5 \\ 0 & 1 \end{bmatrix}$, the target state vector is $s_k = [x_k, \dot{x}_k, y_k, \dot{y}_k]^T$, and x_k and y_k are position of the target in two dimensions plane. $\dot{x}_k$ and $\dot{y}_k$ are velocities of the target. $\xi_k = [\xi_k^x, \xi_k^y]^T$ is assumed to be Gaussian distributed noise, and satisfied $\xi_k^* \sim N(0, \delta_\xi^2)$.

The measurement equation is given by

$$z_k = \tan^{-1}\left(\frac{y_k}{x_k}\right) + v_k. \tag{26}$$

Inside, the measurement noise satisfies $v_k \sim N(0, \delta_v^2)$.

Table 1: Performance comparison between PF, RBPF, and RBPF-PSO.

Algorithm	RMSE	Time (S)	Number of particles
PF	0.14298	19.625	1000
RBPF	0.019902	16.078	500
RBPF-PSO	0.015079	2.297	20

The target initialization state is $s_0 = [-0.005, 0.001, 0.7, -0.005]^T$, process noise covariance $\delta_\xi = 0.001$, measurement noise covariance $\delta_v = 0.005$, Time $K = 60$. We divide target state into nonlinear state s_k^n and linear state s_k^l. Where $s_k^n = [x_k, y_k]^T$, $s_k^l = [\dot{x}_k, \dot{y}_k]^T$. Root-mean-square error (RMSE) is used to describe the accuracy estimation, which is given by

$$\text{RMSE}_k = \frac{1}{N}\sum_{i=1}^N \sqrt{\left(\hat{x}_k^i - x_k\right)^2 + \left(\hat{y}_k^i - y_k\right)^2}. \tag{27}$$

The evaluations of different algorithms are displayed in Figure 1 and Table 1.

From the experimental results, we can see that RBPF-PSO uses a few particles to get the required estimation accuracy and consume the least computing time; RBPF-PSO with 20 particles could achieve higher accuracy than PF with 1000 particles and RBPF with 500 particles. The RMSE of RBPF-PSO shows the best prevision. By this simulation, we could see that it is nearly one order of magnitude promotion between RMSE of the proposed algorithm and the RMSE of standard PF. Meanwhile, the compution time spent by the proposed algorithm is much less than that the others. What is more, in Figure 1, when time $k > 25$, PF and RBPF both start to diverge from the real tracking trajecory.

In addition, we could draw similar conclusions by changing the particle num.

From Figure 2 and Table 2, the RBPF-PSO that uses 10 particles can achieve higher accuracy than the PF with 1000

Table 2: Performance comparison between PF, RBPF, and RBPF-PSO.

Algorithm	RMSE	Time (S)	Number of particles
PF	0.11154	20.156	1000
RBPF	0.23974	16.766	200
RBPF-PSO	0.087333	9.343	10

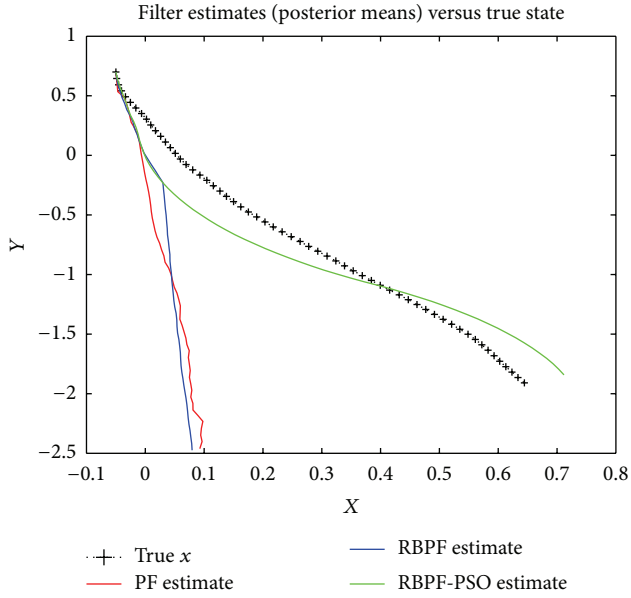

Figure 1: The estimations of different particle filters when the regular PF diverges.

particles and RBPF with 200 particles. RBPF-PSO uses a few particles to get the required estimation accuracy. However, the difference between Figures 1 and 2 is that, in Figure 2, the RBPF algorithm with 200 particles failed to track the tendency of the real data, beyond all expectations. Even with 10 particles, the proposed RBPF-PSO algorithm can still produce smooth and satisfied estimation accuracy. Among the three algorithms, the worst algorithm with the poorest RMSE error; and the best algorithm is still the proposed RBPF-PSO, with the smallest particle number and the best accuracy.

7. Conclusion and Discussions

In this paper, particle swarm optimization Rao-Blackwellized particle filter was proposed. We used particle swarm optimization to make the particle move to regions where the likelihood of particle is high, so only a few particles were able to represent the approximate posterior density. And the efficiency of the algorithm were improved. Even when RBPF algorithm fails to estimate, the proposed RBPF-PSO algorithm can still achieve satisfied estimation accuracy with much less particle number.

Actually, we just take the top maximum value as the unique solution here. However, we think that the interested readers can take multiple local maximum as possible candidate solutions by several subswarms corporations in the implementation detail for further development.

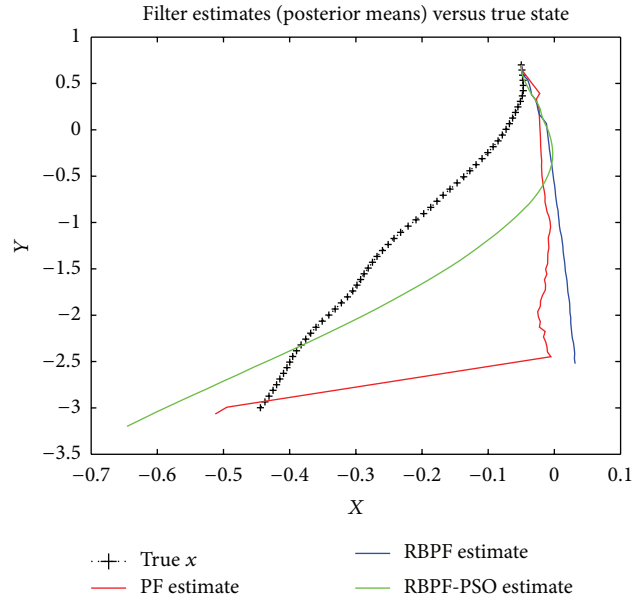

Figure 2: The estimates of different particle filters when the RBPF fails.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grants nos. 60805028, 61175076, and 61225017), Natural Science Foundation of Shandong Province (ZR2010FM027), China Postdoctoral Science Foundation (2012M521336), Open Research Project under Grant 20120105 from SKLMCCS, SDUST Research Fund (2010KYTD101), and SDUST Graduate Innovation Foundation (YC130218).

References

[1] M. N. Rosenbluth and A. W. Rosenbluth, "Monte carlo calculation of the average extension of molecular chains," *The Journal of Chemical Physics*, vol. 23, no. 2, pp. 356–359, 1955.

[2] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-gaussian Bayesian state estimation," *IEE Proceedings F*, vol. 140, no. 2, pp. 107–113, 1993.

[3] J. MacCormick and A. Blake, "A probabilistic exclusion principle for tracking multiple objects," in *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV '99)*, pp. 572–578, September 1999.

[4] J. Carpenter and P. Clifford, "Improved particle filter for nonlinear problems," *IEE Proceedings: Radar, Sonar and Navigation*, vol. 146, no. 1, pp. 2–7, 1999.

[5] D. Crisan, P. del Moral, and T. Lyons, "Discrete filtering using branching and interacting particle systems," *Markov Processes and Related Fields*, vol. 5, no. 3, pp. 293–318, 1999.

[6] P. del Moral, "Nonlinear filtering: interacting particle solution," *Markov Processes and Related Fields*, vol. 2, no. 4, pp. 555–579, 1996.

[7] K. Kanazawa, D. Koller, and S. J. Russell, "Stochastic simulation algorithms for dynamic probabilistic networks," in *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI '95)*, pp. 346–351, 1995.

[8] J. E. Handschin and D. Q. Mayne, "Monte Carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering," *International Journal of Control*, vol. 9, no. 5, pp. 547–559, 1969.

[9] C. Andrieu and A. Doucet, "Particle filtering for partially observed Gaussian state space models," *Journal of the Royal Statistical Society. Series B*, vol. 64, no. 4, pp. 827–836, 2002.

[10] A. Doucet, N. de Freitas, and N. Gordon, Eds., *Sequential Monte Carlo Methods in Practice*, Statistics for Engineering and Information Science, Springer, New York, NY, USA, 2001.

[11] M. J. Daly, J. P. Reilly, and M. R. Morelande, "Rao-Blackwellised particle filtering for blind system identification," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '05)*, pp. IV321–IV324, Philadelphia, Pa, USA, March 2005.

[12] Z. Khan, T. Batch, and F. Dellaert, "A Rao-Blackwellized particle filter for eigentracking," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '04)*, vol. 2, pp. II980–II986, July 2004.

[13] 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.

[14] A. Giremus, E. Grivel, J. Grolleau, and M. Najim, "A Rao-Blackwellized particle filter for joint channel/symbol estimation in MC-DS-CDMA systems," *IEEE Transactions on Communications*, vol. 58, no. 8, pp. 2292–2304, 2010.

[15] S. Särkkä, A. Vehtari, and J. Lampinen, "Rao-Blackwellized particle filter for multiple target tracking," *Information Fusion*, vol. 8, no. 1, pp. 2–15, 2007.

[16] R. Sajeeb, C. S. Manohar, and D. Roy, "Rao-Blackwellization with substructuring for state and parameter estimations of a class of nonlinear dynamical systems," *International Journal of Engineering Under Uncertainty: Hazards, Assessment and Mitigation*, vol. 1, no. 1-2, 2009.

[17] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[18] G. Casella and C. P. Robert, "Rao-blackwellisation of sampling schemes," *Biometrika*, vol. 83, no. 1, pp. 81–94, 1996.

[19] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, IEEE Service Center, Perth, Australia, December 1995.

[20] R. A. Krohling, "Gaussian swarm: a novel particle swarm optimization algorithm," in *Proceedings of the IEEE Conference on Cybernetics and Intelligent Systems (CIS '04)*, pp. 372–376, Singapore, December 2004.

[21] R. Poli, J. Kennedy, and T. Balckwell, "Particle swarm optimization," *Swarm Intelligence*, vol. 1, pp. 33–57, 2007.

[22] Z.-S. Zhao, J.-Z. Wang, X.-Z. Cheng, and Y.-J. Qi, "Particle swarm optimized particle filter and its application in visual tracking," in *Proceedings of the 6th International Conference on Natural Computation (ICNC '10)*, pp. 2673–2676, August 2010.

[23] Z. Zhao, J. Wang, Q. Tian, and M. Cao, "Particle swarm-differential evolution cooperative optimized particle filter," in *Proceedings of the International Conference on Intelligent Control and Information Processing (ICICIP '10)*, pp. 485–490, August 2010.

[24] Y. Li, J. Liang, and J. Hu, "A multi-swarm cooperative hybrid particle swarm optimizer," in *Proceedings of the 6th International Conference on Natural Computation (ICNC '10)*, vol. 5, pp. 2535–2539, August 2010.