A Monte Carlo Particle Model Associated with Neural Networks for Tracking Problem

Zhonyu Pang, Derong Liu, *Fellow, IEEE*, Ning Jin, and Zhuo Wang

***Abstract*—Sequential Monte Carlo (SMC) methods, namely, particle filters, are powerful simulation techniques for sampling sequentially from a complex probability distribution. SMC can be used to solve some problems associated with nonlinear non-Gaussian probability distribution. Sampling is a key step for these methods and has vital effects on simulation results. Various sampling strategies have been proposed to improve the simulation results of SMC methods, but degeneracy of particles sometimes is very severe so that there are only a few particles having significant weights. Diversity of particle samples is reduced significantly so that only a few particles are used to represent the corresponding probability distribution. This kind of sampling is not reasonable to approximate probability distribution. This paper addresses a new method which can avoid the phenomenon of particle degeneracy. We split particles with very big weights into two small ones and use the strategy of neural network to adjust positions of tail particles in order to increase their weights. Another advantage is that this method can efficiently make simulation results approach the actual object. Our simulation results of the typical tracking problem show that not only the phenomenon of particle degeneracy is effectively avoided but also tracking results are much better than those of the traditional particle filters. Compared with the move–resample method, our method shows better results under the same conditions.**

***Index Terms*—Backpropagation, neural networks, particle degeneracy, particle filters, resample–move, tracking.**

I. Introduction

PARTICLE filters were originally introduced in the early 1950s by physicists. Its actual development and implementation began in the 1990s, since computers could provide more powerful ability of computation and make this method a reality. These methods have been very popular over the past few years in statistics and related fields, and they are improved greatly in implementations[11], [16], [17],[28], [30], [32]. They are also used widely in various fields, such as econometrics [15], signal processing [20], [27], noise analysis [14], circuit analysis [23], [36], communications [19], [21], [22], performance analysis of wavelet transforms [33], statistical analysis of the affine project algorithm [1], robotics [26], and so on. Particle filters approximate the sequence of probability distributions of interest using a set of random samples called particles. Those particles are propagated over time following the corresponding distributions by sampling and resampling mechanisms. At any time, as the number of particles increases, particles asymptotically converge toward the sequence of probability distribution. In reality, computation time is a very important factor to consider, so the number of particles cannot go too big. Thus, effective sampling algorithms are key steps to capture a probability distribution by a limited number of particles.

Many researchers are working in this field and have obtained some significant results. Liu and Chen [30] sample particles based on a tail importance density. Some other authors use a very similar strategy to realize sampling, e.g., Doucet *et al.* [12], Guo *et al.*[18]. Gilks and Berzuini [16] rejuvenate particles based on Markov chain Monte Carlo (MCMC), and there are some other similar papers like[7] and [10]. Look-ahead technique is another method developed by Liu [28] for sampling. All these methods cannot effectively reduce the mean square error over time. For a sequence over a long time, performance of these methods are poor. MCMC method in [16] requires fast mixture of kernels in order to obtain a good performance. Look-ahead algorithm needs to compute a local integration, and it is expensive in computation time[10].

Arulampalam *et al.*[2] give a tutorial of particle filters for online nonlinear non-Gaussian Bayesian tracking. They summarize several popular types of particle filters with their properties and adaptation. The purpose of resampling in particle filters is to reduce particle degeneracy. Although this method can improve the performance of particle filters, some other practical problems appear, such as diversity of particle sample and statistical repetition of particles with high weights. They mention that this particle impoverishment is severe in the the case of small process noise.

In this paper, we combine a neural network with the typical sampling algorithm for particles in the tail area of probability distribution with low weights while particles with very high weights are split into small ones. This consideration is motivated by raising diversity of particle samples over time, since a typical particle filter will have only a few particles with high weights after several recursive updates of the probability distributions of interest. Our method can adjust the distribution of particles into the area with high weight of probability distribution and make the results approach the real objective.

We also address the application of particle filters on estimation and tracking of a time-varying target. Dynamic modeling is used to simulate the activity of target using nonlinear Gaussian state-space models. We use local linearization technique to approximate nonlinear function of this dynamic model. We com-

Manuscript received May 28, 2007; revised December 13, 2007. First published May 7, 2008; current version published December 12, 2008. This paper was recommended by Associate Editor A. Kuh.

Z. Pang, N. Jin, and Z. Wang are with the Department of Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, IL 60607-7053 USA (e-mail: zpang2@uic.edu; njin@uic.edu; zwang38@uic.edu).

D. Liu is with the Department of Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, IL 60607-7053 USA, and also with the Key Laboratory of Complex Systems and Intelligence Science, Chinese Academy of Sciences, Beijing 100190, China (e-mail: dliu@ece.uic.edu).

Digital Object Identifier 10.1109/TCSI.2008.924894

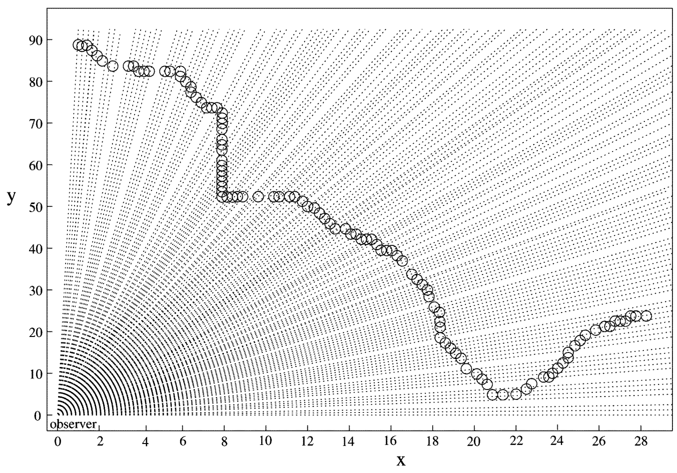


Fig. 1. Ship is moving in $x-y$ plane, where $\cdots$ denotes observed angular positions from the initial fixed point and $\circ$ denotes the true position of ship.

pare our method with the typical particle filter and the resample–move algorithm. Our simulation results, e.g., the number of resampling and error from the actual tracking path under the same conditions, are better than those of the two methods.

This paper is organized as follows. State-space model is represented in Section II. Particle filters are addressed in Section III, including sequential-importance-sampling (SIS)–resampling algorithm, resample–move algorithm, and our method. The simulation results are given in Section IV. Finally, discussion and conclusions are described in Section V.

II. State-Space Model

The problem of interest is the sequential detection of the position of a moving target. As a motivating example, a moving ship is considered in the sea, and one observer is located at a fixed point. This is a classical problem in nonlinear tracking (see Gordon *et al.*[17], Carpenter *et al.*[4], [5], and Gilks *et al.*[16]).

A ship is moving through random smooth accelerations and decelerations as shown in Fig. 1. A stationary observer at the fixed point of the plane obtains a noisy measurement α of the angular position of the ship over time. Assume that x_n, y_n denote coordinates of the ship, where n is discrete time, $n=0,1,2,\ldots$. The state-space model [16] is described as follows:

$$\begin{aligned}\dot{x}_n &\sim N(\dot{x}_{n-1},\tau^2)\\ \dot{y}_n &\sim N(\dot{y}_{n-1},\tau^2)\\ x_n &= x_{n-1}+\dot{x}_{n-1}\\ y_n &= y_{n-1}+\dot{y}_{n-1}\\ \alpha_n &= \tan^{-1}(y_n/x_n)+w_n\end{aligned}\tag{1}$$

where $N(m,v)$ denotes a normal distribution with mean m and variance v, n is discrete time, $\dot{x}_n$ and $\dot{y}_n$ are velocities of x- and y-direction of the ship, respectively, τ is an unknown constant number, w_n is the white noise with Gaussian distribution $N(0,\eta^2)$, and η is an unknown variance.

As the ship moves over time, the observer can measure a series of angular positions of the ship with noise. For a given τ, we actually consider the velocities $\dot{x}_n, \dot{y}_n$ of the ship as Gaussian distribution with varying mean value. Thus, the vector of unknown parameters at time n is

$$\theta_{1:n}=(x_1,y_1,x_2,y_2,\ldots).\tag{2}$$

Therefore, the target distribution of $\theta_{1:n}$ is a posterior distribution of interest

$$\pi_n(\theta_{1:n})=p(\theta_{1:n}|\alpha_{1:n}).$$

We consider coordinates x,y as independent components, and we have

$$\begin{aligned}\pi(x_{1:n})&=p(x_{1:n}|\alpha_{1:n})\\ \pi(y_{1:n})&=p(y_{1:n}|\alpha_{1:n})\end{aligned}\tag{3}$$

where the notation $(\cdot)_{1:n}$ indicates all the elements from discrete time 1 to n. The space of x, y extends with the increase of time. However, at any time, the number of particles keeps constant and is used to approximate the conditional posterior distribution $p(\theta_{1:n}|\alpha_{1:n})$ for the current position of the ship. In addition, each particle should make up a path from the beginning to current time. It is not true in practice since particle degeneracy makes some particles lose their paths. We will address this issue later.

By Bayesian theory, a recursive formula for $p(\theta_{1:n}|\alpha_{1:n})$ can easily be obtained with

$$p(\theta_{1:n}|\alpha_{1:n})=p(\theta_{1:n-1}|\alpha_{1:n-1})\frac{p(\alpha_n|\theta_n)p(\theta_n|\theta_{n-1})}{p(\alpha_n|\alpha_{1:n-1})}\tag{4}$$

where $p(\alpha_n|\theta_n)$ is the likelihood function and $p(\theta_n|\theta_{n-1})$ can recursively be calculated by the prior distribution of the parameters $p(\theta_1)$. From (4), a posterior probability distribution can be obtained either by analysis or by approximation. For model (1), we cannot analytically calculate the terms $p(\alpha_n|\theta_n)$ and $p(\alpha_n|\alpha_{1:n-1})$, since they are nonlinear and non-Gaussian distributions.

Then, we use the local linearization technique [12] to get a linear equation. The aim of linearization here is to obtain an importance function, which can be calculated analytically, and the algorithm should converge asymptotically toward the desired distribution under usual assumptions. Now, consider the observer equation

$$\alpha_n=\tan^{-1}(y_n/x_n)+w_n=f(x_n,y_n)+w_n.$$

By the first-order Taylor expansion formula for the component x, this equation becomes

$$\begin{aligned}\alpha_n&=\tan^{-1}(y_n/x_n)+w_n\\ &\approx\tan^{-1}(y_{n-1}/x_{n-1})+\left.\frac{\partial f(x_n,y_n)}{\partial x_n}\right|_{x_n=g(x_{n-1})}\\ &\quad\times(x_n-g(x_{n-1}))+w_n\end{aligned}\tag{5}$$

where $g(x_{n-1})=x_{n-1}+\dot{x}_{n-1}$ from (1). Thus, a new linear Gaussian observer equation is obtained by linearization of $\tan^{-1}(\cdot)$. Based on (1) and (5), we can do calculation to obtain a Gaussian importance function $p(\theta_n|\alpha_n,\theta_{n-1})\sim N(m_{xn},\sigma_{xn})$ with mean m_{xn} and covariance σ_{xn} for one of components, e.g., x coordinate. m_{xn} and σ_{xn} are obtained by the following formula:

$$\sigma_{xn}^{-1}=\tau^{-1}+\left[\left.\frac{\partial f(x_n,y_n)}{\partial x_n}\right|_{x_n=g(x_{n-1})}\right]^{\mathrm{T}}\eta^{-1} \times \left.\frac{\partial f(x_n,y_n)}{\partial x_n}\right|_{x_n=g(x_{n-1})} \tag{6}$$

$$m_{xn}=\sigma_{xn}\left(\tau^{-1}g(x_{n-1})+\left[\left.\frac{\partial f(x_n,y_n)}{\partial x_n}\right|_{x_n=g(x_{n-1})}\right]^{\mathrm{T}}\eta^{-1} \times\left(\alpha_n-f\left(g(x_{n-1})\right)+\left.\frac{\partial f(x_n,y_n)}{\partial x_n}\right|_{x_n=g(x_{n-1})} \times g(x_{n-1})\right)\right) \tag{7}$$

where T denotes transpose. Since

$$\frac{\partial f(x_n,y_n)}{\partial x_n}=-\frac{y_n}{x_n^2+y_n^2}$$

finally, (6) and (7) become

$$\sigma_{xn}^{-1}=\tau^{-1}+\left[\frac{y_n}{(x_{n-1}+\dot{x}_{n-1})^2+y_n^2}\right]^2\times\eta^{-1} \tag{8}$$

$$m_{xn}=\sigma_{xn}\left(\tau^{-1}(x_{n-1}+\dot{x}_{n-1})+\left[\frac{-y_n}{(x_{n-1}+\dot{x}_{n-1})^2+y_n^2}\right] \times\eta^{-1}\left(\alpha_n-\tan^{-1}\frac{-y_n}{x_{n-1}+\dot{x}_{n-1}} +\left[\frac{-y_n}{(x_{n-1}+\dot{x}_{n-1})^2+y_n^2}\right] \times(x_{n-1}+\dot{x}_{n-1})\right)\right). \tag{9}$$

Following the earlier procedure, the Gaussian importance function for coordinate y can easily be obtained:

$$\sigma_{yn}^{-1}=\tau^{-1}+\left[\frac{x_n}{(y_{n-1}+\dot{y}_{n-1})^2+x_n^2}\right]^2\times\eta^{-1} \tag{10}$$

$$m_{yn}=\sigma_{yn}\left(\tau^{-1}(y_{n-1}+\dot{y}_{n-1})+\left[\frac{x_n}{(y_{n-1}+\dot{y}_{n-1})^2+x_n^2}\right] \times\eta^{-1}\left(\alpha_n-\tan^{-1}\frac{x_n}{y_{n-1}+\dot{y}_{n-1}} +\left[\frac{x_n}{(y_{n-1}+\dot{y}_{n-1})^2+x_n^2}\right] \times(y_{n-1}+\dot{y}_{n-1})\right)\right). \tag{11}$$

At any time point t, there are a number of particles which are used to approximate the importance function distribution, and each particle follows the earlier dynamic model.

III. Particle Filters

This section includes three parts. The first part briefly describes the typical particle filter, namely, the SIS and resampling algorithm. The second one introduces the particle filter with resample–move algorithm [16]. The last one addresses our particle filter with neural networks.

A. SIS–Resampling Algorithm

The basis of particle filters is an SIS algorithm. Most sequential Monte Carlo methods developed over the last decade are based on this SIS algorithm. This technique is capable of implementing a recursive Bayesian filter by Monte Carlo simulations. The key idea is to use a sample of random particles to represent a posterior probability distribution approximately. The sequential sampling is very important in realizing this algorithm. Assume an arbitrary distribution $p(x)$. Samples are supposed to be drawn from $p(x)$, but in many practical cases, $p(x)$ is not a standard probability distribution, e.g., Gaussian distribution, so it is difficult to draw samples from $p(x)$. Therefore, based on the Bayesian importance-sampling scheme [3], a sample $x^i, i=1,\ldots,N$ can be drawn from another probability distribution $q(x)$ called the importance function, which is easy to sample. Thus, these particles can approximate the distribution $q(x)$. In order to use these particles to represent the desired distribution $p(x)$, a weighted approximation to the density $p(x)$ is given by

$$\hat{p}(x)=\frac{\sum_{i=1}^N\tilde{w}^i\delta(x-x^i)}{\sum_{i=1}^N\tilde{w}^i} \tag{12}$$

where

$$\tilde{w}^i=\frac{p(x^i)}{q(x^i)} \tag{13}$$

and $\delta(\cdot)$ is a Dirac delta function defined by

$$\delta(x-x^i)=\begin{cases}1, & \text{if } x=x^i\\ 0, & \text{otherwise.}\end{cases} \tag{14}$$

If the samples are drawn from an importance function $q(x_{1:n}|\alpha_{1:n})$, then the weights in (13) are decided as

$$\tilde{w}^i=\frac{p(x_{1:n}^i|\alpha_{1:n})}{q(x_{1:n}^i|\alpha_{1:n})}. \tag{15}$$

Now, we can proceed to obtain a recursive updating equation which can keep the previous trajectories of particles when a set of new data is available. At each iteration, samples can approximate the corresponding distribution, e.g., $p(x_{1:n-1}|\alpha_{1:n-1})$, and go ahead to approximate $p(x_{1:n}|\alpha_{1:n})$ with a new set of samples. From the Bayesian theory, we can easily obtain

$$q(x_{1:n}|\alpha_{1:n})=q(x_n|x_{1:n-1},\alpha_{1:n})q(x_{1:n-1}|\alpha_{1:n-1}). \tag{16}$$

From (16), we already have samples $x_{1:n-1}^i\sim q(x_{1:n-1}|\alpha_{1:n-1})$ and can draw a particle from $x_n^i\sim q(x_n|x_{1:n-1},\alpha_{1:n})$ to augment samples to $x_{1:n}^i$. The aim is to approximate a density function $p(\cdot)$, so $p(x_{1:n}|\alpha_{1:n})$ is expressed as follows, based on the Bayesian theory and the Markov properties [2]

$$p(x_{1:n}|\alpha_{1:n})=\frac{p(\alpha_n|x_n)p(x_n|x_{n-1})}{p(\alpha_n|\alpha_{1:n-1})}p(x_{1:n-1}|\alpha_{1:n-1}). \tag{17}$$

Now, particle weights are considered. Substituting (16) and (17) into (15), the updated weight equation is

$$\tilde{w}_n^i = \tilde{w}_{n-1}^i \frac{p\left(\alpha_n | x_n^i\right) p\left(x_n^i | x_{n-1}^i\right)}{q\left(x_n^i | x_{1:n-1}^i, \alpha_{1:n}\right)}. \tag{18}$$

In (18), the term $p(\alpha_n|\alpha_{1:n-1})$ is omitted, since it is a value by calculation. Doucet [9] showed that the effect of omission is compensated by normalizing the weights with

$$w_n^i = \frac{\tilde{w}_n^i}{\sum_{i=1}^{N} \tilde{w}_n^i}. \tag{19}$$

The SIS algorithm has been obtained, but two problems exist in practice. One is the phenomenon of degeneracy and the other is the choice of importance function $q(x)$. In general, all but a few particles will have negligible weights after several iterations, and a large computational effort is devoted to updating trajectories whose contribution to the final estimation is almost zero [12]. The variance of particles will increase over time [12], [25], and thus, it is difficult to avoid the degeneracy phenomenon. Liu *et al.*[30] introduced an approximate method to measure particle degeneracy by

$$\hat{N}_{\text{eff}} = \frac{1}{\sum_{i=1}^{N_s} \left(w_n^i\right)^2} \tag{20}$$

where w_n^i is the normalized weight obtained by (19). The smaller the $\hat{N}_{\text{eff}}$, the worse the degeneracy. Generally speaking, increasing the number of particles can reduce degeneracy, but it is impractical. There are two ways to make degeneracy better. One is a choice of optimal importance function and the other is resampling. However, degeneracy still exists although two methods are used. We can briefly describe these two methods.

The optimal importance function [6], [12], [25], [29] can be chosen as

$$q\left(x_n | x_{n-1}^i, \alpha_n\right)_{\text{optimal}} = p\left(x_n | x_{n-1}^i, \alpha_n\right). \tag{21}$$

Substituting (21) into (18), we can get

$$\begin{aligned} w_n^i &\propto w_{n-1}^i p\left(\alpha_n | x_{n-1}^i\right) \\ &= w_{n-1}^i \int p\left(\alpha | x_n^*\right) p\left(x_n^* | x_{n-1}^i\right) \mathrm{d}x_n^*. \end{aligned} \tag{22}$$

In order to use the optimal result in (22), particles must be drawn from density distribution $p(x_n|x_{n-1}^i, \alpha_n)$ and integral in (22) can be analytically calculated. Otherwise, we cannot apply this optimal equation in practice. However, a very close distribution function which is satisfied with the earlier two conditions may be chosen to improve performance of a particle filter. A possible case is that the dynamic model is the one having Gaussian state space with nonlinear transition equation[12]. After linearization, our tracking problem is the case and the optimal function can be used to realize a particle filter.

The second method that can be used to reduce the effect of degeneracy is resampling whenever the particle degeneracy is severe, e.g., the result from (20) is lower than an assigned threshold. The main idea of resampling is to eliminate particles with small weights and to split particles with big weights into equivalent small weights. After this procedure, all the particles have the same weights. It is possible to implement the resampling procedure in $O(N)$ by using a classical algorithm [9], [32], [34].

Resampling procedures can decrease the degeneracy phenomenon but it introduces practical and theoretical problems [12]. From a theoretical point of view, the simulated trajectories are no longer statistically independent after resampling so the previous convergence result will be lost. From a practical point of view, it limits the opportunity to parallel computation since all the particles must be combined, although the importance-sampling steps can still be realized in parallel.

A generic particle filter is then described by Algorithm 1 [12] as follows.

Algorithm 1: SIS/Resampling Particle Filter

1) Importance sampling
   a) For $i = 1, \ldots, N$, sample $\tilde{x}_n^i \sim q(x_n | x_{1:n-1}^i, \alpha_{1:n})$ and set $\tilde{x}_{1:n} \triangleq (x_{1:n-1}^i, \tilde{x}_n^i)$.
   b) For $i = 1, \ldots, N$, evaluate the importance weights up to a normalizing constant:

$$\tilde{w}_n^i = \tilde{w}_{n-1}^i \frac{p\left(\alpha_n | \tilde{x}_n^i\right) p\left(\tilde{x}_n^i | \tilde{x}_{n-1}^i\right)}{q\left(\tilde{x}_n^i | \tilde{x}_{1:n-1}^i, \alpha_{1:n}\right)}. \tag{23}$$

   c) For $i = 1, \ldots, N$, normalize the importance weights:

$$w_n^i = \frac{\tilde{w}_n^i}{\sum_{j=1}^{N} \tilde{w}_n^j}. \tag{24}$$

   d) Evaluate $\hat{N}_{\text{eff}}$ using (20).
2) Resampling
   a) If $\hat{N}_{\text{eff}} \geq N_{\text{threshold}}, x_{1:n}^i = \tilde{x}_{1:n}^i$ for $i = 1, \ldots, N$.
   b) Otherwise, for $i = 1, \ldots, N$, sample an index $j(i)$ distributed according to the discrete distribution with N elements satisfying $\Pr\{j(i) = l\} = w_n^l$ for $l = 1, \ldots, N$. For $i = 1, \ldots, N, x_{1:n}^i = \tilde{x}_{1:n}^{j(i)}$ and $w_n^{*i} = 1/N$.

B. *Resample–Move Algorithm*

Gilks and Berzuini [16] introduced this algorithm in 2001. The purpose of this algorithm is to avoid particle degeneracy over time when a particle filter is used. As we know, degeneracy will make a particle filter suffer from progressive impoverishment of the representativeness of particles as the dynamic process evolves. This method is based on SIS/resampling algorithm and adds one step of resampling–move in order to reduce particle degeneracy.

The algorithm 2 [16] is shown as follows.

Algorithm 2: Resample–Move Algorithm

1) **Initialization**: At $k = 1$, generate the initial set of particles S_1 by sampling, independently for $j = 1, \ldots, n_1$,

$$\theta_1^j \sim \pi_1 \tag{25}$$

where "$\sim$" denotes "is sampled from." π_1 is the target distribution. Then, for $k = 1, 2, \ldots$, we have the rejuvenation step.

2) **Rejuvenation**: At each time $n = k + 1$, calculate weights w_k^i, for $i = 1, \ldots, n_k$. Generate S_{k+1} by performing the following two steps, independently for $j = 1, \ldots, n_{k+1}$:
   a) **resample step**—randomly select a particle from S_k, such that θ_k^i is selected with probability proportional to w_k^i, for $i = 1, \ldots, n_k$, and denote the selected particle by θ_k^{i}.
   b) **move step**—move $\theta_k^{i_j}$ to a new position θ_{k+1}^j by sampling

$$\theta_{k+1}^j \sim q_k^{i_j} \tag{26}$$

   where q_k denotes a Markov chain transition kernel with stationary distribution π_{k+1}.

This algorithm assumes that it is possible to sample independently and directly from the target distribution π_1. The resample step at time $k+1$ can select particles at random from the current particle set S_k, for each $j = 1, \ldots, n_{k+1}$. This is a weight-dependent selection, namely, particles with high weights have a high chance to be selected while particles with small weights have a little chance to be selected. The incremental weights w_k^i used in the resample step involve π_k and π_{k+1}, but they use constants c_k and c_{k+1} to normalize target function π_k and π_{k+1} individually, namely, $\tilde{\pi}_k = c_k \pi_k$ and $\tilde{\pi}_{k+1} = c_{k+1}\pi_{k+1}$ which are easy to obtain.

The move step in this algorithm at time $k + 1$ in effect performs one or more iterations of a MCMC algorithm on each of the particles selected at the resample step. Assume that the distribution of the MCMC is invariant π_{k+1}. The common choice of transition kernel involved in this step is a Gibbs sampler or a Gibbs or Metroplis–Hastings move [35]. The kernel q_k here needs to be neither irreducible nor reversible.

For the tracking problem we consider, Gilks *et al.*[16] gave the detailed implementation of the resample–move algorithm. At any discrete time k, the target posterior distribution of θ_k in (2) can be obtained.

Specifically, in the augmentation stage at any time k, a pair of samples can be obtained by sampling from the conditional prior density function $N(\dot{x}_{k-1}, \tau^2) \times N(\dot{y}_{k-1}, \tau^2)$. Thus, each particle θ_{k-1}^i will be augmented to $\tilde{\theta}_{k-1}^i$.

In the rejuvenation stage at time k, they use three types of moves

1) *Rescaling*: All coordinates x, y and velocities $\dot{x}, \dot{y}$ representing the ship's trajectory are reduced or amplified by the same factor. For the specification, they implemented this move according to a Metropolis–Hastings scheme by using a symmetric proposal distribution.
2) *Local perturbation*: This is the key step to implement resampling–move algorithm. The particle trajectory is perturbed by moving a block of consecutive points of the trajectory to a new position while the remaining part of the trajectory is not changed. This move also follows a Metropolis–Hastings scheme. The move is applied to the vector $\dot{x}$ first and then the vector $\dot{y}$. For any given block $\dot{x}_{a:b}$ currently at value $\dot{x}'_{a:b}$, a new candidate $\dot{x}''_{a:b}$ can be drawn from the proposal distribution.
3) *τ–move*: Update the value of τ. The parameter τ is updated by a standard Gibbs move.

For the details of the earlier moves, see [16] and [11Ch. 6].

C. Particle Filters Associated with Neural Networks

In order to avoid particle degeneracy, many methods have been proposed, e.g., Doucet *et al.*[13] used an optimal importance density function; the resampling technique [24] is applied to particle filters; or the resample–move algorithm[16] is developed. All of these methods can reduce particle degeneracy at different levels for some models, but they introduce some practical problems described before. All of them are not effective in reducing sample variance which increases over time. If time lasts long enough, tracking trajectory of the ship will deviate from the true position significantly, which is not acceptable in practice. Thus, we propose a new method which considers both particle degeneracy and sample variance.

1) Neural Networks: The backpropagation neural networks are one of the most common neural-network structures as they are simple and effective and used in various fields from character recognition to system controls.

The backpropagation neural networks start as a network of nodes arranged in several layers—input, hidden, and output layers. The input and output layers serve as input and output of the model, and the hidden layers serve to connect the input and output layers and to provide values to the output layer. Before any data run through the network, the weights for all the nodes are typically random, but some specific applications can use initial weights given. There are two steps to use the backpropagation neural networks: training and testing. After training, neural networks actually build a nonlinear model to represent relationship between input data and target data. For our problem, we just need the training step since we want to find model parameters with respect to our objective.

The basic idea of the backpropagation neural networks is to use the steepest descent (gradient) procedure to minimize the error energy at the output layer. The error energy can be denoted as follows:

$$\epsilon \triangleq \frac{1}{2}\sum_k (d_k - y_k)^2 = \frac{1}{2}\sum_k e_k^2 \tag{27}$$

where $k = 1, \ldots, N$; N is the number of neurons in the output layer, d_k is the target value, and y_k is the output of neural network. By using gradient procedure and updating weights of all neurons to train a neural network, proper weights can be found to make output of the network close to the desired objective within the assigned error. The activation function in the neural network can be chosen according to the actual problems (for details, see [37] or other related books).

2) Combination of Neural Networks and Particle Filters: For the ship tracking, the dynamic model is given in (1). All the density distributions are Gaussian so we can analytically obtain some function values. Equation (5) linearizes the last equation of (1). Thus, the dynamic model becomes the one with linear Gaussian state space. This model actually is not Markovian as

(5) depends on x_{n-1} and y_{n-1}. We can choose the optimal function $q(x_n|x_{n-1}^i,\alpha_n)=p(x_n|x_{n-1}^i)$, since particles can be sampled from $p(\cdot)$ directly. Thus, the importance weight function (23) becomes

$$\tilde{w}_n^i=\tilde{w}_{n-1}^i p\left(\alpha_n|\theta_{n-1}^i\right). \tag{28}$$

We already derived the posterior density distribution function $p(\theta_n|\alpha_n,\theta_{n-1})$ for components x and y in (8), (9), (10), and (11).

Even though the optimal density functions are used, impoverishment phenomenon still exists, and this does not guarantee that Algorithm 1 will be efficient. If the discrepancy between two consecutive distributions $q_n(\theta_{1:n-1})$ and $q_{n-1}(\theta_{1:n-1})$ is high, then the variance of the incremental weights is pretty high [10]. Consequently, it will be necessary to resample very often, and the particle joint distribution will be unreliable. Thus, the marginal distribution will be approximated by a few particles after a period of time because of resampling. The effective method to improve performance is to modify both particles and weights. This cannot change the fact that sample variance will be increased over time. For a long time, variance will arise to an unbelievable level. Our method considers all earlier elements. In general, there are two extra steps beyond the SIS/resampling algorithm.

The first step is to split. At any time, a set of particles are sampled from the importance distribution $N(m_{xn},\sigma_{xn})$ or $N(m_{yn},\sigma_{yn})$. After normalizing, the particle weights, particles with weights larger than $2/N$ and smaller than $0.2/N$ are identified, where N is the total number of particle. Particles with very big weights are split into two small ones by half the weights. Normally, the amount of this kind of particles is just a few. After this step, the number of particles is augmented. In order to keep a fixed number of particles and to reduce inputs of neural network, the difference between the total particle number before and after augment is calculated. The number of particle equivalent to that difference with smaller weights are ignored. This will not have too much effect on approximating the posterior distribution, since these particles have negligible weights.

The second step is to update particles with small weights by a neural network. The number of input neurons is determined by the number of particles whose weights need to be adjusted. The weights of these particles are set as inputs of the neural network. The hidden layer has two neurons: one for component x and one for component y. The mean values of the weights of particles whose weights do not need to change are computed and are set as biases of corresponding neurons for components x and y, respectively. This allows us to adjust particle weights so that the mean value of particles will approach the true trajectory of the object. The target is the measured angular position at any time. Normally, the measurement noise is small owing to modern measurement techniques.

This procedure can actually reduce variance of samples. Suppose that there are two initial particles sampled from the same probability density function. One set of particles are distributed into areas with high probability density and the other set of particles are distributed partly into areas with high probability density. Even though they have the same mean value, their variances are totally different. After the earlier procedure of neural network, the particles in the area with low probability density can be adjusted into areas with high probability densities. This will reduce sample variance and increase weights of some particles. This procedure can effectively avoid particle degeneracy in particle filters while it can make simulated trajectory close to the real one. The algorithm is summarized as follows.

Algorithm 3: SIS/Resampling Particle Filter With Neural Networks

1) **Importance sampling**
   a) For $i=1,\ldots,N$, sample $\tilde{\theta}_n^i\sim q(\theta_n|\theta_{1:n-1}^i,\alpha_{1:n})$ and set $\tilde{\theta}_{1:n}\triangleq(\theta_{1:n-1}^i,\tilde{\theta}_n^i)$, where $\tilde{\theta}_n^i=(\tilde{x}_n^i,\tilde{y}_n^i)$.
   b) For $i=1,\ldots,N$, evaluate the importance weights for component x up to a normalizing constant

$$\tilde{w}_n^i=\tilde{w}_{n-1}^i\frac{p\left(\alpha_n|\tilde{x}_n^i\right)p\left(\tilde{x}_n^i|\tilde{x}_{n-1}^i\right)}{q\left(\tilde{x}_n^i|\tilde{x}_{1:n-1}^i,\alpha_{1:n}\right)}. \tag{29}$$

   The same procedure is for component y.
   c) For $i=1,\ldots,N$, normalize the importance weights for components x and y, respectively

$$w_n^i=\frac{\tilde{w}_n^i}{\sum_{j=1}^N\tilde{w}_n^j}. \tag{30}$$

   d) At time n, identify particles with high weights larger than $2/N$ and low weights less than $0.2/N$. Replace some low weight particles with high ones if needed.
   e) At time n, adjust particles for components x and y with low weights by neural network. Assign and normalize weights by (29) and (30).
   f) Evaluate $\hat{N}_{\text{eff}}$ using (21).
2) **Resampling if necessary**
   a) If $\hat{N}_{\text{eff}}\geq N_{\text{threshold}}, \theta_{1:n}^i=\tilde{\theta}_{1:n}^i$ for $i=1,\ldots,N$.
   b) Otherwise, for $i=1,\ldots,N$, sample an index $j(i)$ distributed according to the discrete distributions of components x and y with N elements, respectively, satisfying $\Pr\{j(i)=l\}=w_n^l$ for $l=1,\ldots,N$.
   For $i=1,\ldots,N, \theta_{1:n}^i=\tilde{\theta}_{1:n}^{j(i)}$ and $w_n^{*i}=(1)/(N)$.

We know that the sampling and resampling procedure can be implemented with $O(N)$ by using a classical algorithm [9], [32], [34]. Nikova and Stoeva [31] mentioned that the time complexity of backpropagation learning is about $O(n)$, where n is the number of inputs. Therefore, the neural network used in our algorithm can be realized by $O(N)$, since the number of particles N is bigger than the number of inputs of the neural network. Thus, our algorithm has approximately the time complexity of $O(N)$.

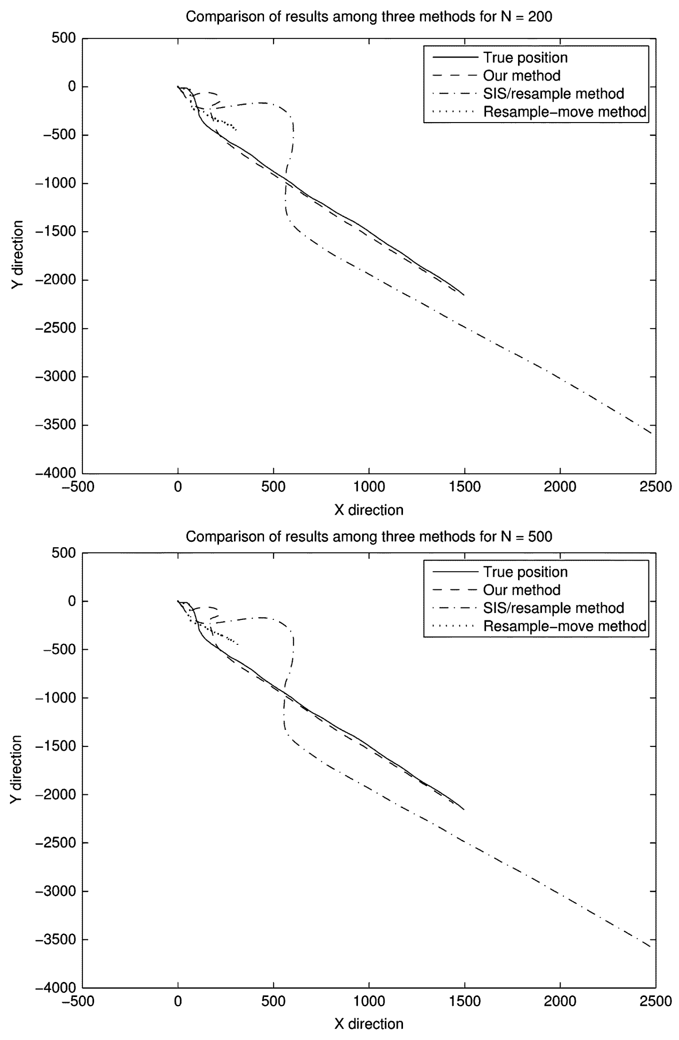


Fig. 2. Comparison of performance among three algorithms for particle number $N = 200, 500$.

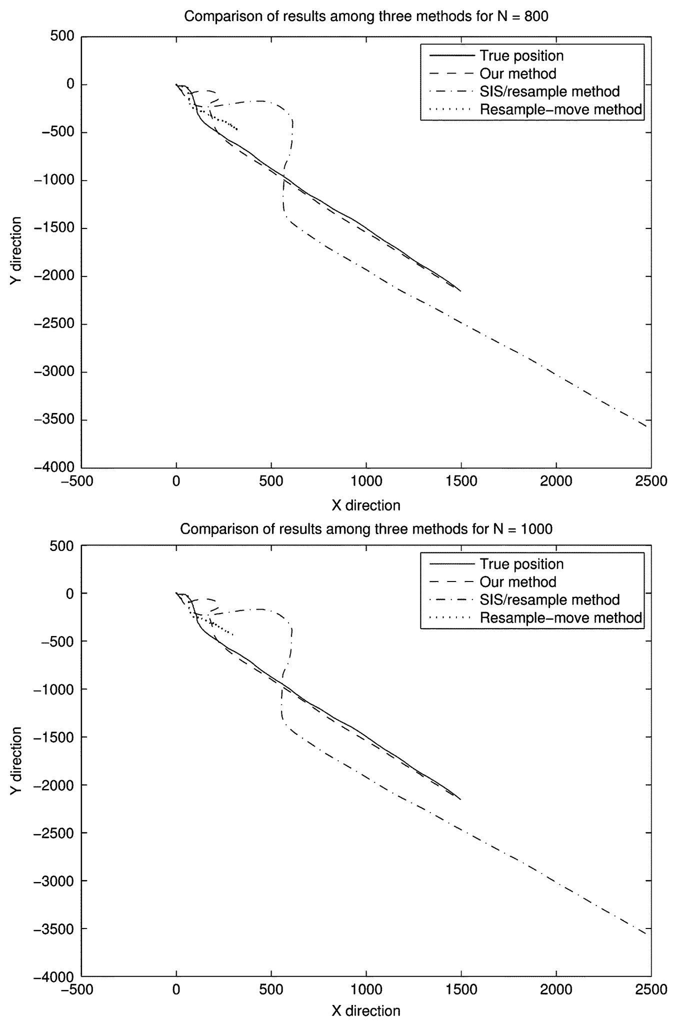


Fig. 3. Comparison of performance of three algorithms for particle number $N = 800, 1000$.

IV. Simulation Results

A series of 200 observations are given by equations in (1) and the initial conditions are given as follows:

$$\begin{aligned} x_0 &= 0.01 \quad y_0 = 0.95 \\ \dot{x}_0 &= 0.002 \quad \dot{y}_0 = -0.013 \\ \tau &= 0.6 \quad \eta = 0.001 \\ p(x_0) &\sim N(0.01, 0.04) \\ p(y_0) &\sim N(0.95, 0.4). \end{aligned} \tag{31}$$

Under the same conditions, simulations are done by using three algorithms, namely, SIS/resampling algorithm, resample–move algorithm, and our algorithm. For the purpose of comparison, we plot the simulation results of three algorithm into one figure. Figs. 2 and 3 show tracking results for particle number $N = 200, 500, 800, 1000$, respectively.

In these figures, the solid line represents the true running trajectory of the ship. The dashed line shows the simulated trajectory by using our method. The dashed dotted line comes from the SIS/resampling algorithm, and the dotted line is obtained by using resample–move algorithm. It is obvious that the results of our method are very close to the true trajectory from the figures. Even when the particle number is 200, our results are still good enough. The standard particle filter of SIS/resampling can capture the direction of ship running, but it cannot track the position of ship precisely. For the resampling–move algorithm, the tendency of ship running is close to the true one, but coordinates are much less than the true ones.

TABLE I
Average Resampling Number of the Three Algorithms for 100 Simulations, 200 Time Instances

Particle number	200	500	800	1000
Our method	2.34	2.2	2.0	0.67
SIS/resamping	917	910.5	909	909
Resampling-move	900.5	890	890	854

We count the resampling number of the three methods. The threshold $N_{\text{threshold}}$ is set to $N/3$, which is the same as [12]. Table I shows results of all three methods.

The standard algorithm of SIS/resampling and resampling–move algorithm need to be resampled very often, while our method has a better performance without resampling so often.

V. Discussion and Conclusion

A particle filter associated with neural networks is addressed in this paper. The other two algorithms, namely, SIS/resampling and resample–move algorithms, are briefly introduced.

This algorithm can efficiently reduce resampling amount and has superior performance. The reason is that particles with low weights are adjusted to the high weight area by using neural networks. The other two algorithms suffer from the resampling problem under the same initial conditions. Very similar performances can be found in [10].

The SIS/resample algorithm and the resample–move algorithm do not consider performance in terms of variance so their simulated paths of the ship deviate the actual one significantly. This is due to the fact of increasing variance over time. The SIS/resample instability is obvious for small number of particles used. If particle number increases, e.g., 5000 or 10 000, the performance would be better than small particle number. However, it will increase computation time very much, so it is impractical to increase particle number unlimitedly. The simulated path of resample–move algorithm may not catch up with the real one. It is obvious when time lasts long. If the variance τ is much smaller than the present one, this method may have a good performance since it can capture the direction of ship running.

Our method can capture both the direction of the ship running and its positions. It is much helpful in practice to use this method. At any time t, this method needs to update particles, and further, it will increase computation time a little. However, this can be compensated by using a small number of particles. In the simulation, observation results have small noise, which is true in reality.

References

[1] S. J. M. de Almeida, J. C. M. Bermudez, N. J. Bershad, and M. H. Costa, "A statistical analysis of the affine projection algorithm for unity step size and autoregressive inputs," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 7, pp. 1394–1405, Jul. 2005.

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

[3] J. Bernardo and A. Smith, *Bayesian Theory*. New York: Wiley, 1994.

[4] J. R. Carpenter, P. Clifford, and P. Fernhead, Building robust simulation-based filters for evolving data sets Dept. Stat., Univ. Oxford, Oxford, U.K., 1998, Tech. Rep.

[5] J. R. Carpenter, P. Clifford, and P. Fernhead, "Improved particle filter for nonlinear problems," *Proc. Inst. Elect. Eng.—Radar, Sonar Navig.*, vol. 146, no. 1, pp. 2–7, Feb. 1999.

[6] R. Chen and J. S. Liu, "Predictive updating methods with application to Bayesian classification," *J. Roy. Stat. Soc. B*, vol. 58, no. 2, pp. 397–415, 1996.

[7] N. Chopin, "A sequential particle filter method for static models," *Biometrika*, vol. 89, no. 3, pp. 539–552, Aug. 2002.

[8] N. Chopin, "Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference," *Ann. Stat.*, vol. 32, no. 6, pp. 2385–2411, 2004.

[9] A. Doucet, On sequential simulation-based methods for Bayesian filtering Signal Process. Group, Univ. Cambridge, Cambridge, U.K., 1998, Tech. rep.

[10] A. Doucet, M. Briers, and S. Sénnécal, "Efficient block sampling strategies for sequential Monte Carlo methods," *J. Comput. Graph. Stat.*, vol. 15, no. 3, pp. 693–711, Sep. 2006.

[11] A. Doucet, N. D. Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. New York: Springer-Verlag, 2001.

[12] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Stat. Comput.*, vol. 10, no. 3, pp. 197–208, Jul. 2000.

[13] A. Doucet, N. Gordon, and V. Krishnamurthy, "Particle filters for state estimation of jump Markov linear systems," *IEEE Trans. Signal Process.*, vol. 49, no. 3, pp. 613–624, Mar. 2001.

[14] B. Fowler, M. D. Godfrey, and S. Mims, "Reset noise reduction in capacitive sensors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 8, pp. 1658–1669, Aug. 2006.

[15] J. Geweke, "Bayesian inference in econometric models using Monte Carlo integration," *Econometrica*, vol. 57, no. 6, pp. 1317–1339, Nov. 1989.

[16] W. Gilks and C. Berzuini, "Following a moving target: Monte Carlo inference for dynamic Bayesian models," *J. Roy. Stat. Soc. B*, vol. 1, no. 1, pp. 127–146, 2001.

[17] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *Proc. Inst. Elect. Eng.—Radar, Signal Process.*, vol. 140, no. 2, pp. 107–113, Apr. 1993.

[18] D. Guo, X. Wang, and R. Chen, "New sequential Monte Carlo methods for nonlinear dynamic systems," *Stat. Comput.*, vol. 15, no. 2, pp. 135–147, Apr. 2005.

[19] S. Haykin, K. Huber, and Z. Chen, "Estimation for MIMO wireless communications," *Proc. IEEE*, vol. 92, no. 3, pp. 439–454, Mar. 2004.

[20] D. He and H. Leung, "Semi-blind identification of ARMA systems using a dynamic-based approach," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 1, pp. 179–190, Jan. 2005.

[21] R. Hormis, D. Guo, and X. Wang, "Monte Carlo FEXT cancellers for DSL channels," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 9, pp. 1894–1908, Sep. 2005.

[22] Y. Huang and P. M. Djuric, "A blind particle filtering detector of signals transmitted over flat fading channels," *IEEE Trans. Signal Process.*, vol. 52, no. 7, pp. 1891–1900, Jul. 2004.

[23] H. Kettani and B. R. Barmish, "A new Monte Carlo circuit simulation paradigm with specific results for resistive networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 6, pp. 1289–1299, Jun. 2006.

[24] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models," *J. Comput. Graph. Stat.*, vol. 5, no. 1, pp. 1–25, Mar. 1996.

[25] A. Kong, J. S. Liu, and W. H. Wong, "Sequential imputations and Bayesian missing data problems," *J. Amer. Stat. Assoc.*, vol. 89, no. 425, pp. 278–288, Mar. 1994.

[26] C. Kwok, D. Fox, and M. Meila, "Real-time particle filters," *Proc. IEEE*, vol. 92, no. 3, pp. 469–484, Mar. 2004.

[27] J. R. Larcoque, J. P. Reilly, and W. Ng, "Particle filters for tracking an unknown number of sources," *IEEE Trans. Signal Process.*, vol. 50, no. 12, pp. 2926–2937, Dec. 2002.

[28] J. S. Liu, *Monte Carlo Strategies in Scientific Computing*. New York: Springer-Verlag, 2001.

[29] J. S. Liu and R. Chen, "Blind deconvolution via sequential imputation," *J. Amer. Stat. Assoc.*, vol. 90, no. 430, pp. 567–576, Jun. 1995.

[30] J. S. Liu and R. Chen, "Sequential Monte Carlo methods for dynamic systems," *J. Amer. Stat. Assoc.*, vol. 93, no. 443, pp. 1032–1044, 1998.

[31] A. Nikova and S. Stoeva, "Quick fuzzy backpropagation algorithm," *Neural Netw.*, vol. 14, no. 2, pp. 231–244, Mar. 2001.

[32] M. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *J. Amer. Stat. Assoc.*, vol. 94, no. 446, pp. 590–599, Jun. 1999.

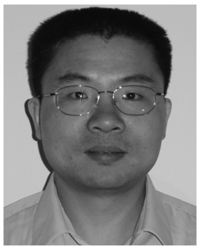
[33] A. M. Reza and L. Zhu, "Analysis of error in the fixed-point implementation of two-dimensional discrete wavelet transforms," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 3, pp. 641–655, Mar. 2005.

[34] B. D. Ripley, *Stochastic Simulation*. New York: Wiley, 1987.

[35] C. Robert and G. Casella, *Monte Carlo Statistical Methods*. New York: Springer-Verlag, 1999.

[36] D. R. Rolston, D. M. Gross, G. W. Roberts, and D. V. Plant, "A distributed synchronized clocking method," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 8, pp. 1597–1607, Aug. 2005.

[37] J. M. Zurada, *Introduction to Artificial Neural Systems*. New York: West Publ. Co., 1992.



Zhonyu Pang received the M.S. degree in electrical engineering from Oklahoma State University, Stillwater, in 2003 and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Chicago, Chicago, in 2007.

He was with a financial firm in Chicago. He is currently with Eagle 7 Trading Company, Chicago. His research interests include biomedical signal processing, epileptic-seizure prediction, artificial intelligence, pattern recognition, and bioinformatics.

Derong Liu (S'91–M'94–SM'96–F'05) received the Ph.D. degree in electrical engineering from the University of Notre Dame, Notre Dame, IN, in 1994.

From 1993 to 1995, he was a Staff Fellow with General Motors Research and Development Center, Warren, MI. From 1995 to 1999, he was an Assistant Professor with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ. In 1999, he joined the University of Illinois at Chicago, Chicago, where he is currently a Full Professor of electrical and computer engineering and of computer science and, since 2005, has been the Director of Graduate Studies in the Department of Electrical and Computer Engineering. In 2008, he was selected into the "100 Talents Program" by the Chinese Academy of Sciences, Beijing. He has published seven books (four research monographs and three edited volumes).

Dr. Liu is a member of Eta Kappa Nu. He is the General Chair for the 2008 IEEE International Conference on Networking, Sensing and Control (Sanya, China). He served as the Program Chair for the 2008 International Joint Conference on Neural Networks, the 2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, the 21st IEEE International Symposium on Intelligent Control (2006), and the 2006 International Conference on Networking, Sensing and Control. He is an elected AdCom member of the IEEE Computational Intelligence Society (2006–2008), Chair of the Chicago Chapter of the IEEE Computational Intelligence Society, Chair of the Technical Committee on Neural Networks of the IEEE Computational Intelligence Society, and Past Chair of the Technical Committee on Neural Systems and Applications of the IEEE Circuits and Systems Society. He was the recipient of the Michael J. Birck Fellowship from the University of Notre Dame (1990), the Harvey N. Davis Distinguished Teaching Award from Stevens Institute of Technology (1997), the Faculty Early Career Development (CAREER) award from the National Science Foundation (1999), and the University Scholar Award from University of Illinois (2006–2009). He is an Associate Editor of *Automatica*. He is currently the Editor of the IEEE Computational Intelligence Society's ELECTRONIC LETTER, an Associate Editor of the IEEE TRANSACTIONS NEURAL NETWORKS, *IEEE Computational Intelligence Magazine*, and the *IEEE Circuits and Systems Magazine*.

Ning Jin is working toward the Ph.D. degree in electrical and computer engineering at the University of Illinois at Chicago, Chicago.

He was an Associate Professor with the Department of Mathematics, Nanjing Normal University. From 2000 to 2005, he was a Visiting Scholar with the Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago. His research interests include optimal control and dynamic programming, artificial intelligence, pattern recognition, neural network, and wavelet analysis.

Zhuo Wang received B.S. degree in electrical engineering from Beihang University, Beijing, China, in 2006. Since the Fall of 2006, he has been working toward the Ph.D. degree in electrical and computer engineering at the University of Illinois at Chicago, Chicago.

His research interests include biosignal processing, intelligent control, and computational neuroscience.