A Smarter Particle Filter

Xiaoqin Zhang[1],Weiming Hu[1],Steve Maybank[2]

[1]National Laboratory of Pattern Recognition, Institute of Automation, Beijing, China
{xqzhang,wmhu}@nlpr.ia.ac.cn
[2]School of Computer Science and Information Systems,Birkbeck College,London,UK
sjmaybank@dcs.bbk.ac.uk

Abstract. Particle filtering is an effective sequential Monte Carlo approach to solve the recursive Bayesian filtering problem in non-linear and non-Gaussian systems. The algorithm is based on importance sampling. However, in the literature, the proper choice of the proposal distribution for importance sampling remains a tough task and has not been resolved yet. Inspired by the animal swarm intelligence in the evolutionary computing, we propose a swarm intelligence based particle filter algorithm. Unlike the independent particles in the conventional particle filter, the particles in our algorithm cooperate with each other and evolve according to the *cognitive effect* and *social effect* in analogy with the cooperative and social aspects of animal populations. Furthermore, the theoretical analysis shows that our algorithm is essentially a conventional particle filter with a hierarchial importance sampling process which is guided by the swarm intelligence extracted from the particle configuration, and thus greatly overcome the sample impoverishment problem suffered by particle filters. We compare the proposed approach with several nonlinear filters in the following tasks: state estimation, and visual tracking. The experiments demonstrate the effectiveness and promise of our approach.

1 Introduction

Particle filters have been extensively studied in the computer vision and pattern recognition community due to its crucial value in numerous applications including visual tracking, robot localization, machine learning, and signal processing.

Essentially, particle filter is a sequential Monte Carlo approach to solve the recursive Bayesian filtering problem, which combines the powerful Monte Carlo sampling technique with Bayesian inference. It relaxes the linearity and Gaussianity constraints of the Kalman filter and provides a tractable solution to non-linear and non-Gaussian problems. The basic idea of particle filtering is to use a number of independent random variables called particles, sampled from a proposal distribution, to represent the posterior probability, and to update the posterior by involving the new observations. The particles is properly propagated and weighted recursively according to the Bayesian rule. Although particle filtering has achieved a considerable success in the analysis of sequential time series, it is faced with a fatal problem-sample impoverishment due to its suboptimal sampling mechanism, based on a proposal distribution. When the proposal distribution is concentrated in the tail of the observation distribution the performance of

the particle filter is very poor since most particles have low weights, thereby leading to the well-known sample impoverishment problem.

Recently PSO (particle swarm optimization) [1–5], a new population based stochastic optimization technique, has received more and more attention because of its considerable success. Unlike the independent particles in the particle filter, the particles in PSO interact locally with one another and with their environment in analogy with the cooperative and social aspects of animal populations, for example as found in birds flocking. Starting from a diffuse population, now called a swarm, individuals, now termed particles, tend to move in the state space and eventually cluster in regions where optimal state is located. The advantages of this mechanism are, on one hand, the robustness and sophistication of the obtained group behavior and, on the other hand, the simplicity and low cost of the computation associated with each particle.

Inspired by the forgoing discussions, we propose a swarm intelligence based particle filter algorithm, in which the particles are viewed as intelligent individuals, e.g. birds, and evolve through communicating and cooperating with each other. Meanwhile, we also conduct a theoretical analysis from a 'Bayesian filtering' perspective, and find that the proposed algorithm is essentially a conventional particle filter with a hierarchial importance sampling process. The hierarchial importance sampling process which consists of two stages: 1) a coarse sampling from the state transition distribution $p(x_t|x_{t-1})$, 2) a fine sampling carried out by the PSO iterations which are based on the 'cognitive' and 'social' aspects of particle populations. In this way, the newest observations are gradually taken into consideration to approximate the sampling results from the optimal proposal distribution $p(x_t|x_{t-1}, y_t)$ [6], and thereby overcome the sample impoverishment problem suffered by convectional particle filters.

This paper is arranged as follows. The standard particle filter and its limitation are presented in Section 2. The proposed annealed Gaussian based particle swarm optimization is introduced in Section 3. Section 4 gives a detailed description of the smarter particle filter and its theoretical analysis. Experimental results are shown in Section 5, and Section 6 is devoted to conclusion.

2 Particle Filter and Its Limitation

To make this paper self-contained, we first briefly review the conventional particle filter , which is described in more detail in [7], and then summarize its major limitation.

2.1 Particle Filter

The particle filter is an on-line Bayesian inference process for estimating the unknown state x_t at time t from sequential observations $y_{1:t}$ perturbed by noise. A dynamic state-space form employed in the Bayesian inference framework is shown as follows [7],

$$\text{state transition model } x_t = f_t(x_{t-1}, \epsilon_t) \leftrightarrow p(x_t|x_{t-1}) \tag{1}$$

$$\text{observation model } y_t = h_t(x_t, \nu_t) \leftrightarrow p(y_t|x_t) \tag{2}$$

where x_t, y_t represent system state and observation, ϵ_t, ν_t are the system noise and observation noise. $f_t(.,.)$ and $h_t(.,.)$ are the state transition and observation models,

which are determined by probability distributions $p(x_t|x_{t-1})$ and $p(y_t|x_t)$ respectively. The Bayesian inference process is achieved by

$$p(x_t|y_{1:t}) \propto p(y_t|x_t)p(x_t|y_{1:t-1}) \tag{3}$$

where the prior $p(x_t|y_{1:t-1})$ is the propagation of the previous posterior along the temporal axis,

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1} \tag{4}$$

When the state transition and observation models are nonlinear and non-Gaussian, the above integration is intractable and one has to resort to numerical approximations such as particle filters. The basic idea of particle filter is to use a number particles $\{x_t^i\}_{i=1}^N$, sampled directly from the state space, to approximate the posterior distribution. Thus the posterior can be formulated as $p(x_t|y_{1:t}) = \frac{1}{N}\sum_{i=1}^N \delta(x_t - x_t^i)$, where $\delta(\cdot)$ is the Dirac function. Since it is usually impossible to sample from the true posterior, an easy-to-implement distribution, the so-called *proposal distribution* denoted by $q(\cdot)$ is employed, hence $x_t^i \sim q(x_t|x_{t-1}^i, y_{1:t}), (i = 1, \cdots, N)$, then each particle's weight is set to

$$w_t^i \propto \frac{p(y_t|x_t^i)p(x_t^i|x_{t-1}^i)}{q(x_t|x_{t-1}^i, y_{1:t})}. \tag{5}$$

Finally, the posterior probability distribution is approximated as $p(x_t|y_{1:t}) = \sum_{i=1}^N w_t^i$ $\delta(x_t - x_t^i)$. After the importance sampling step, a re-sampling step is adopted to ensure the efficiency of the particles' evolution. To summarize, the detail process of particle filter is presented in Algorithm 1.

Algorithm 1 Particle Filter

1. Initialization: for $n = 1, \cdots, N$, sample $x_0^{(n)} \sim p(x_0)$, $w_0^{(n)} = 1/N$.
2. For time steps $t = 1, 2, \cdots$
3. Importance Sampling: for $n = 1, \cdots, N$, draw samples from the importance proposal distribution as follows:
$$\tilde{x}_t^{(n)} \sim q(x_t|x_{t-1}^{(n)}, y_{1:t})$$
4. Weight update: evaluate the importance weights with Equation (5).
5. Normalize the importance weights:
$$\tilde{w}_t^{(n)} = \frac{w_t^{(n)}}{\sum_{i=1}^N w_t^{(i)}}$$
6. Output the statistics of the particles: MMSE or MAP estimate.
7. Resampling: generate N new particles $x_t^{(n)}$ from the set $\{\tilde{x}_t^{(n)}\}_{n=1}^N$ according to the importance weights $\{\tilde{w}_t^{(n)}\}$.
8. Repeat Steps 3 to 7.

2.2 Limitation

The proposal distribution $q(\cdot)$ is critically important for a successful particle filter because it concerns putting the sampling particles in the useful areas where the posterior

is significant. In practice, the state transition distribution $p(x_t|x_{t-1})$ is usually taken as the proposal distribution for its simplicity. However, this proposal distribution contains little information about the current observations, consequently resulting to a inefficient sampling.

As shown in Fig.1(a), when the transition model is situated in the tail of the observation distribution, then the weight of most particles are low, thereby leading to the sample impoverishment problem.

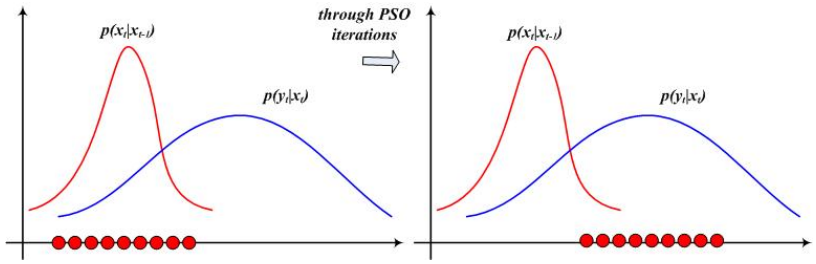


Fig. 1. An illustration of importance sampling (left: sample from $p(x_t|x_{t-1})$, right: after PSO iterations)

3 Annealed Gaussian Based PSO

3.1 Traditional PSO

Particle swarm optimization [1], is a population based stochastic optimization technique, which is inspired by the social behavior of bird flocking. In detail, a PSO algorithm is initialized with a group of random particles $\{x^{i,0}\}_{i=1}^{N}$ (N is the number of particles). Each particle $x^{i,0}$ has a corresponding fitness value which is evaluated by a fitness model $f(x^{i,0})$, and has a relevant velocity $v^{i,0}$ which is a function of the best state found by that particle (p^i, for individual best), and of the best state found so far among all particles (g, for global best). Given these two best values, each particle updates its velocity and state with following equations in the nth iteration,

$$v^{i,n+1} = w^n v^{i,n} + \varphi_1 u_1 (p^i - x^{i,n}) + \varphi_2 u_2 (g - x^{i,n}) \tag{6}$$

$$x^{i,n+1} = x^{i,n} + v^{i,n+1} \tag{7}$$

where w^n is the inertial weight, the φ_1, φ_2 are acceleration constants, and $u_1, u_2 \in (0, 1)$ are uniformly distributed random numbers. The inertial weight w is usually a monotonically decreasing function of the iteration n. For example, given a user-specified maximum weight w_{max} and a minimum weight w_{min}, one way to update w is as follows:

$$w^{n+1} = w^n - dw,\ dw = (w_{max} - w_{min})/T \tag{8}$$

where T is the maximum iteration number. In Eq.(6), the three different parts represent *inertial velocity*, *cognitive effect* and *social effect* respectively. After the nth iteration,

the fitness value of each particle is evaluated by a predefined fitness model as follows.

$$f(x^{i,n+1}) = p(y^{i,n+1}|x^{i,n+1}) \tag{9}$$

where $y^{i,n+1}$ is the observation corresponding to the state $x^{i,n+1}$. Then the individual best and global best of the particles are updated in the following equations:

$$p^i = \begin{cases} x^{i,n+1}, & \textbf{if } f(x^{i,n+1}) > f(p^i) \\ p^i, & \textbf{else} \end{cases} \tag{10}$$

$$g = \arg\max_{p^i} \ f(p^i) \tag{11}$$

In this way, the particles search for the optima through the above iterations until the fitness value of g reaches a certain threshold or the maximum iteration number is encountered.

3.2 Annealed Gaussian Based PSO

In the above version of PSO algorithm, there are several parameters to be tuned: inertial weights w^n, acceleration constants φ_1, φ_2. There is a lack of a mechanism for controlling of these parameters, which fosters the danger of swarm explosion and divergence especially in high dimensions. Therefore, we propose an annealed Gaussian based particle swarm optimization (AGPSO) algorithm, where the particles and their velocities are updated in the following way,

$$v^{i,n+1} = |r_1|(p^i - x^{i,n}) + |r_2|(g - x^{i,n}) + \eta \tag{12}$$

$$x^{i,n+1} = x^{i,n} + v^{i,n+1} \tag{13}$$

where r_1, r_2 are random numbers sampled from the Gaussian probability distribution $\mathcal{N}(0,1)$, and η is zero-mean Gaussian perturbation noise to avoid trapping in local optima whose covariance matrix is changed in an adaptive simulated annealing way [8]:

$$\Sigma_\eta^n = \Sigma e^{-cn} \tag{14}$$

where Σ is the covariance matrix of the predefined transition distribution, c is an annealing constant, and n is the iteration number. Compared with the traditional PSO, it has two major merits: a) a big reduction in the number of parameters–there is a single annealing parameter, b) it converges much faster than traditional PSO (see Section 5.1).

4 Swarm Intelligence Based Particle Filter

4.1 Motivation

In [6], it is shown that the 'optimal' importance proposal distribution is $p(x_t|x_{t-1}^i, y_t)$ in the sense of minimizing the variance of the importance weights. However, in practice, it is impossible to use $p(x_t|x_{t-1}^i, y_t)$ as the proposal distribution in the non-linear and non-Gaussian cases, since it is difficult to sample from $p(x_t|x_{t-1}^i, y_t)$ and to evaluate $p(y_t|x_{t-1}^i) = \int p(y_t|x_t)p(x_t|x_{t-1}^i)dx_t$. So the question is, how to incorporate the current observation y_t into the transition distribution $p(x_t|x_{t-1})$ to form an effective proposal distribution at a reasonable computation cost.

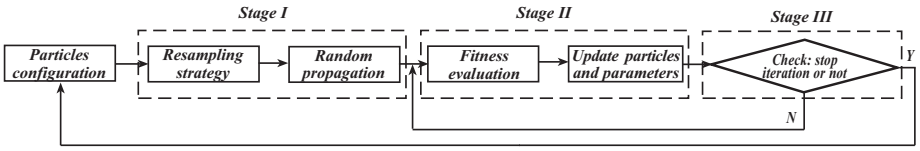


Fig. 2. Overview of the proposed algorithm

4.2 The Smarter Particle Filter

From the description of Section 3, we can see that the PSO iterations can naturally take the observation into consideration, since the particles cooperate and evolve according to their fitness values which are updated by their corresponding observations. Inspired by this property of the PSO, we propose a swarm intelligence based particle filter, in which the particles are firstly propagated by the state transition model, and then corporately evolve according to the PSO iterations.

To give a clear view, the flowchart of the swarm intelligence based particle filter is shown in Fig.2. First, the individual best of particles from the previous time $t-1$ are resampled and randomly propagated by state transition model to enhance their diversities. Then, by moving the particle swarm towards the particle with the best fitness value, PSO drives all particles towards high likelihood regions. Finally, when the fitness value of g_t reaches a certain threshold or the maximum iteration number is encountered, the optimized sampling process is stopped. The global best g_t or the mean of individual best p_t^i is output as the maximum a posterior (MAP) estimate or minimum mean square error (MMSE) estimate. The details of the proposed algorithm are as follows.

1. **Input:** the N individual best particles $\{p_{t-1}^i\}_{i=1}^N$ at time $t-1$;
2. Resample the above particles according to their fitness value, resulting to a new particle set $\{\tilde{p}_{t-1}^i\}_{i=1}^N$;
3. Randomly propagate the particle set to enhance their diversities according to the following transition model
$$x_t^{i,0} \sim p(x_t|\tilde{p}_{t-1}^i)$$
4. **for** $n = 0, 1, 2, \ldots, T$ **do**
5. Carry out the PSO iteration based on Equations (12),(13)
$$v_t^{i,n+1} = |r_1|(p_t^i - x_t^{i,n}) + |r_2|(g_t - x_t^{i,n}) + \eta$$
$$x_t^{i,n+1} = x_t^{i,n} + v_t^{i,n+1}$$
6. Evaluate the fitness values
$$f(x_t^{i,n+1}) = p(y_t^{i,n+1}|x_t^{i,n+1})$$
where $y_t^{i,n+1}$ is the observation corresponding to $x_t^{i,n+1}$;
7. Update the two best particles and the covariance matrix
$$p_t^i = \begin{cases} x_t^{i,n+1}, \ \textbf{if } f(x_t^{i,n+1}) > f(p_t^i) \\ p_t^i, \quad \textbf{else} \end{cases}, \ g_t = \arg\max_{p_t^i} f(p_t^i)$$
$$\Sigma_\eta^{n+1} = \Sigma e^{-c(n+1)}$$

8. Check the convergence criterion;
9. If satisfied, **break**;
10. **end for**
11. **Output:** the global best g_t or the mean of $\{p_t^i\}_{i=1}^N$;

4.3 Theoretical Analysis From Bayesian Filtering View

In this part, we conduct a theoretical analysis of our algorithm from a Bayesian filtering view, and show why our algorithm improves on the particle filter.

Hierarchical Importance Sampling In our algorithm as described in Section 4.2, we take a two-stage sampling strategy to generate samples that approximate to the 'optimal' proposal distribution: first, the particles are sampled from the state transition distribution $p(x_t|x_{t-1})$; second, the sampled particles evolve through the PSO iterations to obtain the final importance sampling.

From the particle filtering view, we can see that our strategy is essentially a hierarchical importance sampling. In the coarse importance sampling stage, the particles are firstly sampled from the state transition distribution as in conventional particle filters to enhance their diversity.

$$x_t^{i,0} \sim p(x_t|\tilde{p}_{t-1}^i) \tag{15}$$

In the fine importance sampling stage, the particles evolve through PSO iterations, and are updated according to the newest observations. In fact, this is essentially a latent multi-layer importance sampling process with an implicit proposal distribution. Suppose $x_t \in \mathbb{R}^d$ be *d*-dimensional state, let's focus on one PSO iteration in Section 4.2, suppose $x_t \in \mathbb{R}^d$ is a *d*-dimensional state, the distribution of the *l*th element in the vector $|r_1|(p_t^i - x_t^{i,n})$ is as follows:

$$|r_1|(p_t^i - x_t^{i,n})_l, \sim \begin{cases} 2\mathcal{N}(0, (p_t^i - x_t^{i,n})_l^2)\ [0, +\infty), \ \textbf{if}\ \ (p_t^i - x_t^{i,n})_l \geq 0 \\ 2\mathcal{N}(0, (p_t^i - x_t^{i,n})_l^2)\ (-\infty, 0), \ \textbf{else} \end{cases}$$

where $l = 1, \cdots, d$, so the distribution of $|r_1|(p_t^i - x_t^{i,n})$ is

$$|r_1|(p_t^i - x_t^{i,n}) \sim R_1 = 2\mathcal{N}(0, \Sigma_1),\ \Sigma_1 = \begin{pmatrix} (p_t^i - x_t^{i,n})_1^2 & \mathbf{0} \\ & \ddots \\ \mathbf{0} & (p_t^i - x_t^{i,n})_d^2 \end{pmatrix}$$

Similarly available,

$$|r_2|(g_t - x_t^{i,n}) \sim R_2 = 2\mathcal{N}(0, \Sigma_2),\ \Sigma_2 = \begin{pmatrix} (g_t - x_t^{i,n})_1^2 & \mathbf{0} \\ & \ddots \\ \mathbf{0} & (g_t - x_t^{i,n})_d^2 \end{pmatrix}$$

Together with $\eta \sim R_3 = \mathcal{N}(0, \Sigma_\eta)$, the implicit proposal distribution behind a PSO iteration is $R = R_1 * R_2 * R_3$[1] with a $x_t^{i,n}$ translation. Here $*$ stands for convolution operator.

[1] Since the analytical form of R is not available, we called it latent sampling process.

In this way, the PSO iterations can naturally take the current observation y_t into consideration, since $\{p_t^i\}_{i=1}^N$ and g_t are updated to their observations. Therefore, with coarse importance sampling stage from the state transition distribution $p(x_t|\tilde{p}_{t-1}^i)$, the hierarchical sampling process can approximate to the optimal sampling from $p(x_t|x_{t-1}^i, y_t)$.

As shown in Fig.1, when the transition distribution is situated in the tail of the observation likelihood, the particles directly drawn from this distribution do not cover a significant region of the likelihood, and thus the importance weights of most particles are low, resulting to unfavorable performance. In contrast, through hierarchial sampling process in our algorithm, the particles are moved towards the region where the likelihood of observation has larger values, and are finally relocated to the dominant modes of the likelihood, demonstrating the effectiveness of our sampling strategy.

5 Experimental Results

We compare the performance of our algorithm to several non-linear filters on two estimation problems: 1) a synthetic state estimation problem; 2) real world visual tracking problem. All of the experiments are carried out on a CPU Pentium IV 3.2GHz PC with 512M memory[2].

5.1 State Estimation

The algorithm is firstly tested on a non-linear state estimation problem, which is described as benchmark in many papers [9]. Consider the following nonlinear state transition model given by

$$x_t = 1 + \sin(w\pi(t-1)) + \phi_1 x_{t-1} + v_{t-1}, \quad x_t \in \mathbb{R} \tag{16}$$

where v_{t-1} is a Gamma $\mathcal{G}a(3,2)$ random variable modeling the process noise, and $w = 4e-2$ and $\phi_1 = 0.5$ are scalar parameters. A non-stationary observation model is as follows

$$y_t = \begin{cases} \phi_2 x_t^2 + n_t, & t \leq 30 \\ \phi_3 x_t - 2 + n_t, & t > 30 \end{cases} \tag{17}$$

where $\phi_2 = 0.2, \phi_3 = 0.5$, and the observation noise n_t is drawn from a Gaussian distribution $\mathcal{N}(0, 0.00001)$. Given only the noisy observation y_t, several filters are used to estimate the underlying state sequence x_t for $t = 1 \cdots 60$. Here, we compare our algorithm (with AGPSO) with conventional particle filter [7], extended Kalman based particle filter [10], unscented particle filter [9], auxiliary particle filter [11], and our algorithm (with traditional PSO)[3]. For each algorithm, a proposal distribution is chosen as shown in Table 1. The parameters in APSO and PSOPF are set as follows: $\Sigma = 0.8, c = 2, \varphi_1 = \varphi_2 = 1, w_{max} = 0.8, w_{min} = 0.1, T = 20$. Fig.3 gives an illustration of the estimates generated from a single run of the different filters. Compared with other nonlinear filters, our algorithm is more robust to the outlier, where the

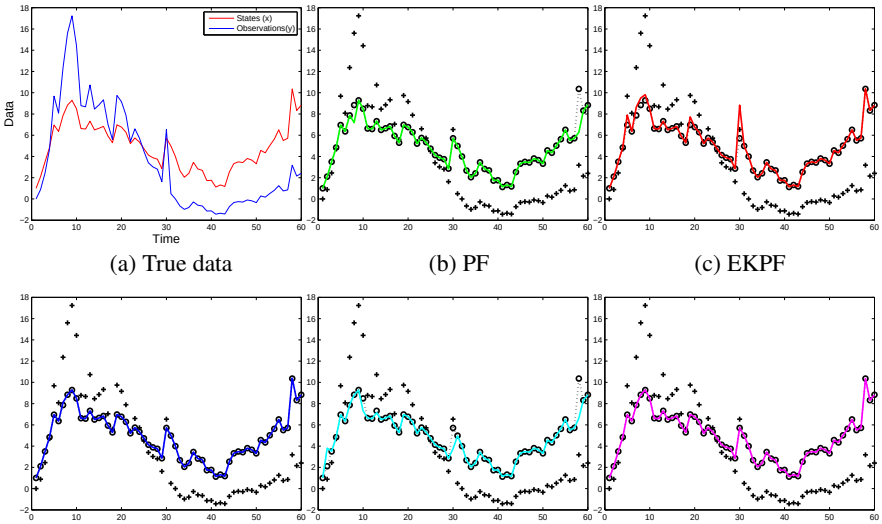
[2] The data and code used in these experiments are available by writing to the authors.

[3] We call these filters AGPSOPF, PF, EKPF, UPF, APF, PSOPF respectively for short in the following parts.

Algorithm	Proposal	MSE mean	MSE var	Time(s)
Particle filter (PF)	$p(x_t\|x_{t-1})$	0.42225	0.045589	3.6939
Extended Kalman particle filter (EKPF)	$N(\bar{x}_t, \bar{P}_t)$	0.31129	0.015167	13.014
Unscented particle filter (UPF)	$N(\bar{x}_t, \bar{P}_t)$	0.06977	0.024894	26.2815
Auxiliary particle filter (APF)	$p(x_t\|x_{t-1})$	0.55196	0.037047	7.1835
Our algorithm (with PSO)	$p(x_t\|x_{t-1})$	0.13019	0.044086	10.2087
Our algorithm (with AGPSO)	$p(x_t\|x_{t-1})$	0.060502	0.06852	6.8005

Table 1. Experimental results of state estimation

observation is severely contaminated by the noise. Since the result of a single run is a random variable, the experiment is repeated 100 times with re-initialization to generate statistical averages. Table 1 summarizes the performance of all the different filters in the following aspects: the means, variances of the mean-square-error (MSE) of the state estimates and the average execute time for one run. It is obvious that the average accuracy of our algorithm is better than generic PF, EKPF, APF and comparable to that of UPF. However, the real-time performance of our algorithm is much better than UPF as Table 1 shows. Meanwhile, we can see that AGPSOPF can achieve a much faster convergence rate than PSOPF. This is because the velocity part employed in Eq.(6) carries little information, while the annealing part in our PSO iterations enhances the diversity of the particle set and its adaptive effect enables a fast convergence rate. In summary, the total performance of our algorithm prevails over that of other nonlinear filters.



(a) True data (b) PF (c) EKPF

(d) UPF (e) APF (f) Our algorithm

Fig. 3. An illustration of a single run of different filters

5.2 Visual Tracking

In this part, we apply these filters (except EKPF and PSOPF) to a rapid motion tracking task to further demonstrate the effectiveness of the sampling strategy in our algorithm.

Fig. 4. Tracking performances of a human face with rapid motion (green: PF, blue: UPF, cyan: APF, magenta: our algorithm)

This video sequence[4] contains a human face with a rapid motion (see Fig.4). In tracking application, $p(x_t|x_{t-1})$ is used to model the object motion, so when $p(x_t|x_{t-1})$ is not coincident with the actual motion, the sampling directly from $p(x_t|x_{t-1})$ will not be efficient. Therefore, although this sequence seems simple, its rapid and arbitrary motion is a challenge for the different improvements of sampling strategy.

In our implementation, we adopt an incremental learned subspace based appearance model [12] for observation evaluation, and we consider only translational motion $x = (t_x, t_y)$ for simplicity, since our goal is to test the sampling efficiency of all the non-linear filters. Here, $p(x_t|x_{t-1})$ is set to a Gaussian distribution with a covariance matrix $\Sigma = diag(8^2, 8^2)$, and the annealing const is also set to 0.3, and the particle number is set to 200. As shown in Fig.4, the PF based tracker and APF based tracker soon fail to track the object, because the particles directly sampled from the state transition distribution $\mathcal{N}(x_{t-1}, \Sigma_s)$ can not catch the rapid motion of the object, and thus the weights of most particles are low, leading to the tracking failure. More particles and an enlargement for the diagonal elements of the covariance matrix would improve its performance, but this strategy involves more noises and a heavy computational load, and it may trap in the curse of dimensionality when the dimension of the state increases. While the UPF based tracker can follow the object throughout the sequence, the localization accuracy is unsatisfactory. In comparison, our algorithm, which evolves the particles by the swarm intelligence based importance sampling, never loses the target and achieves the most accurate results. Furthermore, we have conducted a quantitative evaluation of these algorithms, and have a comparison in the following aspects: frames of successful tracking, RMSE (root mean square error) between the estimated position and the labeled groundtruth, and average tracking time of each frame. In Table 3, our algorithm outperforms the other filters based trackers in accuracy with a reasonable sacrifice of speed, which witnesses the effectiveness our sampling strategy.

Algorithm	Frames Tracked	RMSE of Position (by pixels)	Average Tracking Time (by seconds)
PF	5/31	33.4580	0.051
UPF	31/31	3.5097	80.785
APF	4/31	38.7260	0.098
AGPSO	31/31	2.0112	0.731

Table 2. Quantitative results of the tracking performance

6 Conclusion

In this paper, we propose a swarm intelligence based particle filter to overcome the sample impoverishment problem. Unlike the independent particles in the convectional

[4] The sequence is available at http://vision.stanford.edu/ birch/headtracker/seq/.

particle filters, the particles in our algorithm cooperate each other and evolve according to the *cognitive effect* and *social effect* in analogy with the cooperative and social aspects of animal populations. We conduct a theoretical analysis in a Bayesian filtering view, and find that our algorithm is essentially a convectional particle filter with a hierarchial importance sampling process which is guided by the swarm intelligence extracted from particle configuration. The experimental results demonstrate the effectiveness and promise of our approach.

Acknowledgment

This work is partly supported by NSFC (Grant No. 60825204, 60672040, 60705003) and the National 863 High-Tech R&D Program of China (Grant No. 2006AA01Z453, 2009AA01Z318).

References

1. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks. Volume 4. (1995) 1942–1948
2. Clerc, M., Kennedy, J.: The particle swarm-explosion, stability, and convergence in amultidimensional complex space. IEEE Transactions on Evolutionary Computation **6**(1) (2002) 58–73
3. Wachowiak, M., Smolikova, R., Zheng, Y., Zurada, J., Elmaghraby, A.: An approach to multimodal biomedical image registration utilizing particle swarm optimization. IEEE Transactions on Evolutionary Computation **8**(3) (2004) 289–301
4. Zhang, X., Hu, W., Maybank, S., Li, X., Zhu, M.: Sequential particle swarm optimization for visual tracking. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. (2008) 1–8
5. Zhang, X., Hu, W., Li, W., Qu, W., Maybank, S.: Multi-object tracking via species based particle swarm optimization. In: Proceedings of International Workshop on Visual Surveillance. (2009)
6. Doucet, A., Godsill, S., Andrieu, C.: On sequential monte carlo sampling methods for bayesian filtering. Statistics and Computing **10**(3) (2000) 197–208
7. Arulampalam, M., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. IEEE Transactions on Signal Processing **50**(2) (2002) 174–188
8. Ingber, L.: Simulated annealing: Practice versus theory. Journal of Mathematical and Computer Modeling **18**(1) (1993) 29–57
9. Merwe, R., Doucet, A., Freitas, N., Wan, E.: The unscented particle filter. Advances in Neural Information Processing Systems (2001)
10. Freitas, D., Niranjan, M., Gee, A., Doucet, A.: Sequential monte carlo methods to train neural network models. Neural Computation **12**(4) (2000) 955–993
11. Pitt, M., Shephard, N.: Filtering via simulation: Auxiliary particle filters. Journal of the American Statistical Association **94**(446) (1999) 590–591
12. Lim, J., Ross, D., Lin, R., Yang, M.: Incremental learning for visual tracking. Advances in Neural Information Processing Systems **17** (2005) 793–800