Journal of Intelligent Learning Systems and Applications, 2010, 2: 49-53
doi:10.4236/jilsa.2010.21007 Published Online February 2010 (http://www.SciRP.org/journal/jilsa)

Particle Filtering Optimized by Swarm Intelligence Algorithm

Wei Jing, Hai Zhao, Chunhe Song, Dan Liu

Institute of Information and Technology, Northeastern University, Shenyang, China.
Email: Jingw@mail.neuera.com, {zhhai, songchh, liud}@mail.neuera.com

Received October 27th, 2009; accepted January 7th, 2010.

ABSTRACT

A new filtering algorithm — PSO-UPF was proposed for nonlinear dynamic systems. Basing on the concept of re-sampling, particles with bigger weights should be re-sampled more time, and in the PSO-UPF, after calculating the weight of particles, some particles will join in the refining process, which means that these particles will move to the region with higher weights. This process can be regarded as one-step predefined PSO process, so the proposed algorithm is named PSO-UPF. Although the PSO process increases the computing load of PSO-UPF, but the refined weights may make the proposed distribution more closed to the poster distribution. The proposed PSO-UPF algorithm was compared with other several filtering algorithms and the simulating results show that means and variances of PSO-UPF are lower than other filtering algorithms.

Keywords: *Filtering Method, Particle Filtering, Unscented Kalman Filter, Particle Swarm Optimizer*

1. Introduction

Sequential signal processing has a wide range of applications in many fields such as statistical signal processing [1], target tracking [2,3], *et al*.. Currently, there are many filtering algorithm such as EKF [4], UKF [5], PF [6], UPF [7], and so on. Particle filtering is a young filtering method. Its advantage over other sequential methods is particularly distinctive in situations where the used models are nonlinear and the involved noise processes are non-Gaussian. An important feature in the implementation of particle filters is that the random measure is recursively updated. With the random measure, one can compute various types of estimates with considerable ease. Particle filtering has three important operations, sampling, weight computation, and re-sampling. With sampling, one generates a set of new particles that represents the support of the random measure, and with weight computation, one calculates the weights of the particles. Re-sampling is an important operation because without which particle filtering will get poor results. With re-sampling one replicates the particles that have large weights and removes the ones with negligible weights.

Eberhart and Kennedy (1995) proposed the Particle Swarm Optimization (PSO) algorithm is motivated from the simulation of birds' social behavior. With many advantages of computing with real number, few parameters to be adjusted, the PSO algorithm is applied in many fields such as NN-training, Optimization, and Fussy Control etc. PSO is an optimization strategy generally employed to find a global best point.

In this paper, a new filtering algorithm — PSO-UPF was proposed for nonlinear dynamic systems. Basing on the concept of re-sampling, particles with bigger weights should be re-sampled more time, and in the PSO-UPF, after calculating the weight of particles, some particles will join in the refining process, which means that these particles will move to the region with higher weights. This process can be regarded as one-step predefined PSO process, so the proposed algorithm is named PSO-UPF. Although the PSO process increases the computing load of PSO-UPF, but the refined weights may make the proposed distribution more closed to the poster distribution. The proposed PSO-UPF algorithm was compared with other several filtering algorithms and the simulating results show that means and variances of PSO-UPF are lower than other filtering algorithms.

The remaining of the paper is organized as follows: in the Section 2, a brief description of GPF is presented. In the Section 3, firstly, the PSO algorithm is introduced, and a new type PSO — one-step predefined PSO process is proposed. Then the details of the new PF this paper proposed — PSO-UPF is presented. In Section 4 the proposed algorithm is compared to other several different

filtering algorithms, and finally, some concluding remarks is given in Section 5.

2. Basic Particle Filter

The problem being addressed here is an estimating problem of the state of a system as a set of observations becomes available on-line, which can be expressed as follows:

$$\begin{aligned} x_t &= f(x_{t-1}) + w_{t-1} \\ y_t &= h(u_t, x_t) + v_t \end{aligned} \tag{1}$$

where $x_t \in \Re^{n_x}$, $y_t \in \Re^{n_y}$, $u_t \in \Re^{n_u}$, $v_t \in \Re^{n_v}$ are the state of the system, the output observations, the input observations, the process noise and the measurement noise. The mappings:

$f : \Re^{n_x} * \Re^{n_x} \mapsto \Re^{n_x}$ and $h : \Re^{n_x} * \Re^{n_v} \mapsto \Re^{n_y}$ represent the deterministic process and measurement models.

In the bayes filtering paradigm, the posterior distribution is updated recursively over the current state x_t given all observations $Z_t = \{z_i\}_{i=1}^{t}$ up to and including time t as follows [6]:

$$p(x_t \mid Y_{t-1}) = \int p(x_t \mid x_{t-1}) p(x_{t-1} \mid Y_{t-1}) dx_k \tag{2}$$

$$p(x_t \mid Y_t) = \frac{p(y_t \mid x_t) p(x_t \mid Y_{t-1})}{p(y_t \mid Y_{t-1})} \tag{3}$$

$$p(x_t \mid Y_t) = \frac{p(y_t \mid x_t) p(x_t \mid Y_{t-1})}{p(y_t \mid Y_{t-1})} \tag{4}$$

Using Monte Carlo sampling points, particle filter executes the filtering process by generating weighted sampling points of state variances recursively. Generic particle filter algorithm can be predicted as follows [6]:

Initialization

For each particle

draw the states $x_0^{(i)}$ from the prior $p(x_0)$;

End

For each loop

importance sampling step

For each particle

sample $\hat{x}_0^{(i)} \sim q(x_t \mid x_{0:t-1}^{(i)}, y_{1:t})$

End

For each particle

evaluate the importance weights up to a normalizing constant:

$$w_t^{(i)} = w_{t-1}^{(i)} \frac{p(y_t \mid x_t^{(i)}) p(\hat{x}_t^{(i)} \mid x_{t-1}^{(i)})}{q(\hat{x}_t^{(i)} \mid \hat{x}_{0:t-1}^{(i)}, y_{1:t})} \tag{5}$$

For each particle, normalize the importance weights:

$$\tilde{w}_t^{(i)} = w_{t-1}^{(i)} [\sum_{j=1}^{N} w_t^{(j)}]^{-1} \tag{6}$$

// Re-sample

Eliminate the samples with low importance weights and multiply the samples with high importance weights, to obtain N random samples $x_{0:k}^{(i)}$ approximately distributed according to $p(x_{0:k} \mid z_{1:k})$.

Assign each particle an equal weight: $w_t^i = 1/N$.

When executing particle filtering, the choice of the proposal distribution is very important. Usually, the transition prior distribution is chosen to be the proposal distribution:

$$q(x_t \mid X_{t-1}, Y_t) = p(x_t \mid x_{k-1}) \tag{7}$$

But as not considering the recent observation, when using the transition prior distribution as the proposal distribution, the filtering result is usually poor, especially when the noise is heavy. In this paper, a kind of semi-iterative unscented transformation was proposed to address this issue.

3. The PSO-UPF Algorithm

3.1 Particle Swarm Optimizer Algorithm

PSO is an optimization strategy generally employed to find a global best point. At each time step, a particle updates its position and velocity by the following equations:

$$\begin{aligned} v_{ij}(t+1) = w * v_{ij}(t) &+ c_1 r_{1j}(t)(p_{ij}(t) - x_{ij}(t)) \\ &+ c_2 r_{2j}(t)(p_{gj}(t) - x_{ij}(t)) \end{aligned} \tag{8}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{9}$$

$$w_t = w_{\max} - (w_{\max} - w_{\min}) \frac{t}{\mathrm{g}_{\max}} \tag{10}$$

where $j \in \{1, 2, ..., Dn\}$, $i \in \{1, 2, ..., n\}$, n is the size of the population and Dn is the Dimension of the space searched; w is the inertia weight, generally updated as equation 3 [10], and $\mathrm{g}_{\max}$ is the maximal evolution generations. c_1 and c_2 are two positive constants; r_1 and r_2 are two random values into the range $[0, 1]$.

3.2 PSO in the PF Process

As depicted in the Section 2, in the re-sampling process of regular particle filtering, the particles with bigger weights will have more offspring, while the particles with smaller weights will have few or even on offspring. It inspires us to find more particles with big weights and reduce the number of particles with small weights, which will make the proposal distribution more closed to the poster distribution. And it is the aim of using particles swarm optimization in the particles filtering process.

The most important issue of using particles swarm

optimizer is the choice of fitness function. In the proposed algorithm, we want to find more particles with bigger weights, so the fitness can be chosen as the value of weights directly. As usually the aim of PSO algorithms is to minimize the fitness function, so in the PSO-PF, the fitness is the minus value of particle weight as follows:

$$fitness_t^{(i)} = \frac{p(y_t \mid x_t^{(i)})p(\hat{x}_t^{(i)} \mid x_{t-1}^{(i)})}{q(\hat{x}_t^{(i)} \mid \hat{x}_{0:t-1}^{(i)}, y_{1:t})} \tag{11}$$

Secondly, as the computing consumption of particle filtering is already very large, so the computing consumption of new introduced PSO process should be reduced. In the PSO-PF algorithm, for every particle, at first, a random number in the range of 0 and 1 will be generated, and only when the number is smaller than a predefined threshold, the PSO process can be conducted. A new type of PSO process — one step predefined PSO is introduced. In this process, only when the new location is better than the original one, the particle will move to the new one, and the location updating process will be conducted only one time in each particle filtering generation.

Finally, the direction of particle updating is determined by the location of best particle in current generation and itself location. And as in the one-step predefined PSO process, particle need not remember its individual best location of history; the updating mode uses the social only mode of PSO algorithm.

3.3 The PSO-UPF Algorithm

In this section, the mentioned one-step predefined PSO process is used in the UPF algorithm. The information of UPF algorithm can be found in [9,10]. And he PSO-UPF algorithm can be decried as follows:

For each particle

draw the states $x_0^{(i)}$ from the prior $p(x_0)$;

set $\overline{x}_0^{(i)} = E[x_0^{(i)}]$

$P_0^{(i)} = E[(x_0^{(i)} - \overline{x}_0^{(i)})(x_0^{(i)} - \overline{x}_0^{(i)})^T]$

$x_0^{(i)a} = E(x_0^{(i)a}) = [(x_0^{(i)})^T \ 0 \ 0]^T$

$P_0^{(i)a} = E[(x_0^{(i)a} - \overline{x}_0^{(i)a})(x_0^{(i)a} - \overline{x}_0^{(i)a})^T]$

For each loop

// Generate proposal distribution and resample

For each particle

// calculate sigma points:

$\chi_{t-1}^{(i)a} = [\overline{x}_{t-1}^{(i)a} \ \ \overline{x}_{t-1}^{(i)a} \pm \sqrt{(na+\lambda)P_{t-1}^{(i)a}}]$

// update particles into next time:

$\chi_t^{(i)x} = f(\chi_{t-1}^{(i)x}, \chi_{t-1}^{(i)v}), \overline{x}_t^{(i)x} = \sum_{j=0}^{2n_a} W_j^{(m)} \chi_{j_{t|t-1}}^{(i)x}$

$P_{t|t-1}^{(i)x} = \sum_{j=0}^{2n_a} W_j^{(m)} [\chi_{j_{t|t-1}}^{(i)x} - \overline{x}_{j_{t|t-1}}^{(i)x}][\chi_{j_{t|t-1}}^{(i)x} - \overline{x}_{j_{t|t-1}}^{(i)x}]^T$

$\gamma_{t|t-1}^{(i)x} = h(\chi_{t|t-1}^{(i)x}, \chi_{t|t-1}^{(i)n}], \overline{y}_{t|t-1}^{(i)x} = \sum_{j=0}^{2n_a} W_j^{(m)} \gamma_{j_{t|t-1}}^{(i)x}$

// Measurement update:

$P_{\tilde{y}_t \tilde{y}_t} = \sum_{j=0}^{2n_a} W_j^{(m)} [\gamma_{j_{t|t-1}}^{(i)x} - \overline{y}_{j_{t|t-1}}^{(i)x}][\gamma_{j_{t|t-1}}^{(i)x} - \overline{y}_{j_{t|t-1}}^{(i)x}]^T$

$P_{x_t y_t} = \sum_{j=0}^{2n_a} W_j^{(m)} [\chi_{j_{t|t-1}}^{(i)x} - \overline{x}_{j_{t|t-1}}^{(i)x}][\gamma_{j_{t|t-1}}^{(i)x} - \overline{y}_{j_{t|t-1}}^{(i)x}]^T$

$K_t = P_{x_t y_t} P_{\tilde{y}_t \tilde{y}_t}^{-1}$

$\overline{x}_t^{(i)} = \overline{x}_{t|t-1}^{(i)} + K_t(y_t - y_{t|t-1}^{(i)})$

$\hat{P}_t^{(i)} = P_{t|t-1}^{(i)} + K_t P_{\tilde{y}_t \tilde{y}_t} K_t^T$

// Sample

$\hat{x}_t^{(i)} \sim q(x_t^{(i)} \mid x_t^{(i)}, y_{1:t}) \triangleq N(\overline{x}_t^{(i)}, \hat{P}_t^{(i)})$

Set $\hat{x}_{0:t}^{(i)} = (x_{0:t}^{(i)}, \hat{x}_t^{(i)})$ and $\hat{P}_{0:t}^{(i)} = (P_{0:t}^{(i)}, \hat{P}_t^{(i)})$

For each particle:

Calculate fitness as equ.11, denoted by F;

Find the particle with best fitness, suppose the location is L*;

For each particle

Generate a random number c in the range of [0 1];

If c<C // C is the predefined threshold.

newLocation* = originalLocation + rand*(L*-orginalLocaion);

calculate the fitness of newLocation* , denoted by F*;

if F*<F

orginalLocation = newLocation;

end if

end if

end for

For each particle

Normalize the importance weights as equ.4

Re-sample

Eliminate the samples with low importance weights and multiply the samples with high importance weights, to obtain N random samples $x_{0:k}^{(i)}$ approximately distributed according to $p(x_{0:k} \mid z_{1:k})$.

Assign each particle an equal weight: $w_t^i = 1/N$.

4. The Simulation Experiments

In this paper, the proposed algorithm was compared to other seven filtering algorithm — EKF, UKF, GPF, GPFMCMC, EKFPF, EKFPFMCMC, UPF, Experimental settings are shown in the Table 1. The settings of

other six filtering algorithms are the same as [4–8].

4.1 Benchmark Function

In this paper, the following benchmark function [10] was chosen to test the proposal algorithm.

Bechmark 1:

$$x_t = 1 + \sin(w\pi(t-1)) + \frac{1}{2}x_{t-1} + u_t$$

$$y_t = \begin{cases} \frac{1}{5}x_t^2 + v_t, & t \le 30 \\ \frac{1}{2}x_t - 2 + v_t & t > 30 \end{cases}$$

Bechmark 2:

$$x_t = 1 + \sin(0.04 * \pi * (t-1)) + 0.5 * x_{t-1} + w_{t-1}$$

$$y_k = 0.2 * x_k^2 + \cos(x_t)/10 + v_k$$

where $w_t \sim \gamma(3,1)$, $v_k \sim N(0,1e-2)$, particle number N=200, sample time T=100, whicle the results were the average of 50 times of experiments. $C = 0.5$, $R = 1e-2$, $Q = 0.75$; $\alpha = 0.5$, $\beta = 0.5$, $k = 1$。

4.2 Experimental Results and Some Remarks

Experimental results are shown in Figure 1–Figure 2 and Table 1~Table 1. All results are the means of 100 runs, as the results of UPF and SIUPF are close and far different with others, an enlargement figure was drawn as Figure 3.

As shown in the experimental results, it is clear that, EKF has the worst results, but has the best running time. While the proposed algorithm has the best results, but has longer running time. In theory, the running time of PSO-PF is between one and two times of UPF, due to the refining step, and the simulation results has proved it.

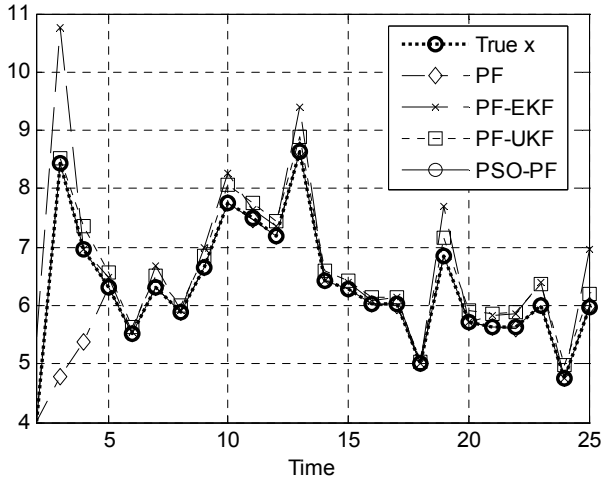

Figure 1. Results on benchmark 1

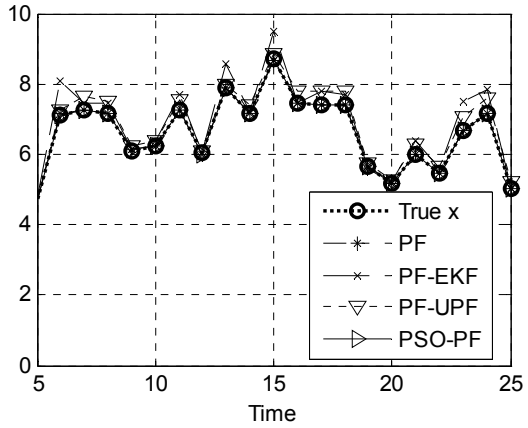

Figure 2. Results on benchmark 2

Table 1. Results on benchmark 1

	MSE		MeanRunTime
	mean	variance	
EKF	0.6975	0.1248	-
UKF	0.3234	0.1297	-
PF	0.18001	0.1650	0.4733
PF-EKF	0.1426	0.2147	4.1292
PF-UKF	0.1239	0.1107	7.2136
PSO-PF	0.0268	0.0359	8.2324

Table 2. Results on benchmark 2

	MSE		MeanRunTime
	mean	variance	
EKF	0.3375	0.1438	-
UKF	0.2347	0.1277	-
PF	0.2301	0.6247	0.3903
PF-EKF	0.3181	0.1147	5.1652
PF-UKF	0.2339	0.1347	7.1336
MPF	0.0968	0.0959	8.1224

5. Conclusions

Basing on the concept of re-sampling, particles with bigger weights should be re-sampled more time, this paper has proposed a new type of particle filtering algorithm — PSO-UPF. In the PSO-UPF, after calculating the weight of particles, some particles will join in the refining process, which means that these particles will move to the region with higher weights. This process can be regarded as one-step predefined PSO process, so the proposed algorithm is named PSO-UPF. Although the PSO process increases the computing load of PSO-UPF, but the refined weights may make the proposed distribution more closed to the poster distribution. In the following experiment, the proposed algorithm has better performances than other several types of filtering methods.

REFERENCES

[1] D. Guo and X. Wang, "Quasi-monte Carlo filtering in nonlinear dynamic systems," IEEE Transactions on Sig-

nal Process, Vol. 54, No.6, pp. 2087–2098, 2006.

[2] M. S. Arulampalam, S. Maskell, N. Gordon, *et al*., "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking [J]," IEEE Transactions on Signal Processing, Vol. 20, No. 2, pp. 174–188, 2002.

[3] D. Crisan and A. Doucet, "A survey of convergence results on particle filtering methods for practitioners [J]," IEEE Transactions on Signal Processing, Vol. 50, No. 3, pp. 736–746, 2002.

[4] B. D. Anderson, and J. B. Moore, "Optimal filtering," Prentice–Hall, New Jersey, 1979.

[5] S. J. Julier and J. K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," Proceedings of AeroSense: The 11th International Sysmpsium on Aerospace/Defence Sensing, Simulation and Controls, Orlando, Florida, Muti Sensor Fusion, Tracking and Resource Management II, pp. 182–193. 1997.

[6] Y. Shi, and R. C. Eberhart, "A modified particle swarm optimizer," in Proceedings of the IEEE International Conference on Evolutionary Computation, Piscataway, NJ: IEEE Press, pp. 69–73, 1998.

[7] J. A Riget, "Diversity-guided particle swarm optimizer —the ARPSO," EVALife Technical Report, Department of Computer Science, University of Arhus, 2002.

[8] D. Guo, X. Wang, and R. Chen, "New sequential monte carlo methods for nonlinear dynamic systems," Statistics and Computing, Vol. 15, No. 2, pp. 135–147, 2005.

[9] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, "Estimation with applications to tracking and navigation: Theory, Algorithm and Software [M]," New York: Wiley, 2001.

