Particle Filter versus Particle Swarm Optimization for Object Tracking

Howida A. Abd El-Halym*, Imbaby I. Mahmoud**, A. AbdelTawab*** and S. E.-D. Habib†

Abstract: Recently, Particle Swarm Optimization (PSO) is introduced as an alternative to Particle Filter (PF) for object tracking. In this work, we compare a PSO tracker with two PF trackers, a classical PF tracker and an Enhanced Particle Filter (EPF) tracker, introduced in this paper. The accuracy of the tracking and, in particular, occlusion handling are considered. The different trackers are implemented in MatLab environment and compared against calculated Ground Truth of the object under tracking. Experimental results indicate that the implemented EPF tracker performs better than the PSO tracker, and it additionally has a lower computational overhead.

Keywords: Particle Filter, Particle swarm Optimization, Object Tracking

I. Introduction

Object tracking is a challenging task in computer vision and has a wide range of applications. In spite of the substantial research effort expended to tackle this challenge, developing a robust and efficient tracking algorithm still remains unsolved due to the inherent difficulty of the tracking problem.

Recently, Particle filters have been extensively used in tracking field. They proved to be a robust method of tracking due to their ability of solving non-Gaussian and non-linear problems [1].

PFs perform sequential Monte Carlo estimation based on particle representation of probability densities. That is by representing the posterior density function by a set of random samples with associated weights and computing estimates based on these samples and weights. PFs propagate the particles with high weights and eliminate the particles with smallest weights by a resampling procedure. Resampling step may introduce a serious problem; namely sample impoverishment. Sample impoverishment occurs when the likelihood is very narrow or the likelihood distribution function lies at the tail of prior distribution. Under this circumstances, only a few particles with significant weights are available. To solve this sample

* Assistant Lecturer, Engineering Dept., NRC, Atomic Energy Authority, howidaaa@yahoo.com
** Prof., Engineering Dept., NRC, Atomic Energy Authority, imbabyisma@yahoo.com.
*** Demonstrator, Misr University of Science and Technology (MUST), a.abdeltawab@yahoo.com.
† Prof., Electronics and Communications Dept., Faculty of Engineering, Cairo University, seraged@ieee.org

1/12
impoverishment problem, one may enlarge the sample set so as to cover the whole state space and ensure successful estimation, the computation efficiency will be adversely affected.

To overcome the degeneracy of conventional PF, several techniques are used [2-4]. Bai and Liu [2] add the Mean-shift algorithm to the particle filter. The Mean-shift algorithm is used to move every particle to a better position near to their original position. Park et al. [3] embed a genetic algorithm into the particle filter to optimize the position of the particles considering particles as chromosomes. Kalman Particle Filter (KPF) is proposed in [4] for adaptively adjusting the number of particles during the resampling step using Kalman Filter (KF).

Also, Particle Swarm Optimization (PSO) is used to overcome the limitations of PF. Tong et al. [5] introduce PSO into generic PF. They employ PSO for intra frame optimization and PF between successive frames. Wang et al. [6] introduce another PSO procedure into PF. They combine PSO with a mutation operator which can guarantee enhancing PSO to obtain local optima. The iterative optimization procedure of the enhanced PSO is used to redistribute particles to their close local modes of the posterior.

Moreover, Zhang et al. [7] show theoretically that, in a Bayesian inference view, the sequential PSO framework is a multilayer importance-sampling-based particle filter.

These research directions combine or embed other techniques with PF to solve the drawbacks of PF. These directions improve the PF performance while increasing the computational cost and time consumed. Combining PSO as an iterated algorithm with PF tracker needs a huge computational load and time cost.

In this paper we modify the PF as a robust estimator by maintaining its advantages and solving its disadvantages, specially sample impoverishment. The modification is based on re-exciting the particles if their weights fall below a memorized weight value. The memorized weight value is calculated from the particle weights in the previous frames. The resultant algorithm is implemented and its behavior is compared with the classical PSO and PF algorithms.

II- Particle Filter Algorithm

The PFs are formulated on the concepts of the Bayesian theory and the sequential importance-sampling which are very effective in dealing with non-Gaussian and non-linear problems [8].

The PF approximates recursively the posterior distribution using a finite set of weighted samples. The idea is to represent the required posterior density function by a set of random samples with associated weights and to compute estimates based on these samples and weights. PF uses the probabilistic system transition model \( p(X_t|X_{t-1}) \), (which describes the transition for state vector \( X_t \)) to predict the posterior at time \( t \) as:

\[
p(X_t | Z_{t-1}) = \int p(X_t | X_{t-1}) p(X_{t-1} | Z_{t-1}) dX_{t-1}
\]

Where \( Z_{1: t-1} = \{Z_1, Z_2, ..., Z_{t-1}\} \) are available observations at times 1, 2, ..., t-1, \( p(X_t|X_{t-1}) \) expresses the motion model, \( p(X_{t-1}|Z_{1: t-1}) \) is posterior probability density function at time t-1 and \( p(X_t|Z_{1: t-1}) \) is the prior Probability Density Function (PDF) at time t. At time t, the observation \( Z_t \) is available, then the state can be updated using Bayes's rule as:
\[ p(X_t | Z_{1:t}) = \frac{p(Z_t | X_t) p(X_t | Z_{1:t-1})}{p(Z_t | Z_{1:t-1})} \]  

(2)

Where \( p(Z_t | X_t) \) is described by the observation equation. The posterior PDF \( p(X_{t-1} | Z_{t-1}) \) is approximated recursively as a set of \( N \) weighted samples \( \{X^{(s)}_{t-1}, W^{(s)}_{t-1}\}_{s=1}^{N} \), and \( W^{(s)}_{t-1} \) is the weight for particle \( X^{(s)}_{t-1} \). Using a Monte Carlo approximation of the integral, we get:

\[ p(X_t | Z_t) = p(Z_t | X_t) \sum_{s=1}^{N} W^{(s)}_{t-1} p(X_t | X^{(s)}_{t-1}) \]  

(3)

The \( N \) samples \( X^{(s)}_{t} \) are drawn from the proposal distribution:

\[ q(X) = \sum_{s=1}^{N} W^{(s)}_{t-1} \cdot p(X_t | X^{(s)}_{t-1}) \]  

(4)

Then it is weighted by the likelihood.

\[ W^{(s)}_{t} = p(Z_t | X^{(s)}_{t}) \]  

(5)

This produces a weighted particle approximation \( \{X^{(s)}_{t-1}, W^{(s)}_{t-1}\}_{s=1}^{N} \) for the posterior PDF \( p(X_t | Z_t) \) at time \( t \).

2-1 Object motion model

We represent the object being tracked by a rectangular window 8x8 pixel centered at the centroid of the object. We define the state vector of the object as five parameters:

\[ X = (x, y, \dot{x}, \dot{y}, I) \]

Where \( x \) and \( y \) are positions in \( x \) and \( y \) directions, \( \dot{x} \) and \( \dot{y} \) are velocities in \( x \) and \( y \) directions and \( I \) is the mean gray intensity level of the rectangular window centered at \((x, y)\) position. The motion between two consecutive frames is approximated as a simple transformation. The motion transition equation is:

\[ X_t = A X_{t-1} + G \]  

(6)

Where \( A \) is the state transition matrix and \( G \) is a Gaussian distribution with zero mean and covariance matrix whose diagonal entries are the corresponding variance of the state variable \( \sigma_x, \sigma_y, \sigma_{\dot{x}}, \sigma_{\dot{y}}, \sigma_I \).

2-2 Likelihood model

To perform the similarity measurements, we use three different likelihoods: position likelihood \( LP \); gray level intensity likelihood \( LI \); and similarity likelihood \( LS \).

2-2-1 Position likelihood

To compare the motion information of the tracked object between the particles state vectors and the object state vector, we use the following likelihood model:

\[ LP(Z^{(i,j)}_t) = \exp\left(-\frac{(i - y)^2 + (j - x)^2}{2\sigma_p^2}\right) \]  

(7)
where \((i, j)\) is the particle position, \((x, y)\) is the previous object position and \(\sigma_p\) is the position standard deviations.

2-2-2 Intensity likelihood

The intensity likelihood ratio of particle at pixel \((i, j)\) for an object at state \(X_t^{(s)}\) is defined as:

\[
LI(Z_{t}^{(i,j)}) = \exp\left(-\frac{(I_{t}^{(i,j)} - Z_{t}^{(i,j)})^2}{2\sigma_{i}^2}\right)
\]  

(8)

Where \(I_{t}^{(i,j)}\) is the mean gray intensity level estimated at particle position \((i, j)\) at time \(t\), \(Z_{t}^{(i,j)}\) is the observation gray level intensity at position \((i, j)\) at time \(t\), and \(\sigma_{i}\) is the intensity gray level standard deviation.

2-2-3 Similarity likelihood

To overcome the problem produced by changes in object shape, size, contour, .. etc due to translation, rotation or sizing, we use a block of size 8x8 pixels around the centroid of the object to measure the similarity likelihood for each particle. We assume that, if the object translate, rotate or change in size, the similarity of this small block around its centroid between two consecutive frames does not change so rapidly because of the inertia. The similarity Likelihood \(LS\) is given by:

\[
LS\left(Z_{t}^{(i,j)}\right) = \frac{\sum_{n=-d}^{m=d} \sum_{m=-d}^{m=d} A_{(i+n,j+m)} B_{(x+n,y+m)}}{\sqrt{A^2 \cdot B^2}}
\]

(9)

Where \(A_{(i,j)}\) is the block 8x8 centered at particle position \((i, j)\) and \(B_{(x,y)}\) is the object kernel centered at \((x, y)\) position.

The overall likelihood is defined as the product of the three likelihoods:

\[
\text{Likelihood} = \text{LP} \cdot \text{LI} \cdot \text{LS}
\]

(10)

III- Particle Swarm Optimization Algorithm

The particle swarm optimization (PSO) method is an optimization method based on natural behavior of birds flocking or fish schooling [9]. Each potential solution is assigned a randomized position vector, and the potential solutions called particles, move through the search space seeking the function optima. Particles change their direction based on the combination of their own experience and the best experience of the group. In each iteration, the particles are updated with swarm motion equations that combines 'cognitive' and 'social' forces to move each particle towards its own best location and towards the globally best location, with added random entropy:

\[
V_t = WV_{t-1} + c_1 R_1 (P_b - X_{t-1}) + c_2 R_2 (p_g - X_{t-1})
\]

(11)

\[
X_t = X_{t-1} + V_t
\]

(12)

Where \(W\) is an ‘inertial’ weights, \(P_b\) is the particle’s previous best state, \(p_g\) is the globally best location of the entire population, \(R_1\) and \(R_2\) are randomly generated noise, \(c_1\) is the cognitive weight and \(c_2\) is the social weight.
IV- Excitation Particle Filter Tracker

The particle impoverishment problem of the PF occurs when the likelihood is very narrow or the likelihood lies in the tail of the prior distribution. This case leads to few particles having significant importance weights. Thus, the sample set contains only few dissimilar particles. To solve this problem, we propose to re-excite the particles, similar to the particle re-excitation of PSO. If we re-excite the particles every frame, we lose the sequential propagation of the particles. Also, we lose the PF robustness specially in the occlusion situations. The question now is: When we must re-excite the particles? The answer is simply: re-excite if the sample impoverishment occurs; i.e., if all particles have low weight values. We memorize the maximum calculated weight of the particles from the previous tracking frames. Then, comparing the current maximum weight value to this memorized value enables us to decide if it is suitable to excite the particles or not. We excite the particles according to an excitation model similar to the used motion model:

\[ X_E = A X_{last} + G \]  

(13)

Except that \( X_{last} \) is the state vector with the previous position, velocity and mean intensity of the tracked object. G is a Gaussian distribution with zero mean and covariance matrix whose diagonal entries are the corresponding variance of the state variable \( \sigma_x, \sigma_y, \sigma_x', \sigma_y', \sigma_I \).

A pseudo code description of the proposed EPF tracker algorithm is given in Table 1.

V- Implementation and Comparison Test

To evaluate EPF tracker algorithm performance against PF and PSO tracker, we implemented the EPF tracker, the PF and the sequential PSO based tracker algorithms in MatLab. The implementation of the three algorithms used the same number of particles (100 particles), the same state vector, the same likelihood functions (fitness functions in the case of sequential PSO), and the same motion model. For sequential PSO, we used equal values for \( c_1 \) (the cognitive weight), and \( c_2 \) (the social weight). W, the inertial weights, decreased from 0.9 to 0 during 10 iterations for each frame. PSO tracker is implemented such that it iterates for 10 iterations for each new frame by exciting the particles at each incoming frame and iterating towards the globally best location.

5-1 Comparison Test

Several real video sequences are used to compare the three implemented tracking algorithms. We illustrate our results with three test sequences A, B, and C. Sequence A [10] is a video clip for a remotely-controlled toy helicopter as it is manually landed against a high texture background. The helicopter is partially occluded for few frames within this sequence. Fig. 1 shows the behavior of the different trackers when applied to sequence A. The three trackers PSO, PF and EPF almost have the same performance as the object has high contrast with the background in the first frames. As the object is occluded with a similar background around frame 650, the PSO tracker fails. This failure is due to the occurrence of local maximum which happened because PSO excites the particles at each frame and optimize the fitness functions through the iterations. EPF has a different behavior, as the particles weights fall below the memorized weights; the excited particles are forced to redistribute to catch the object.

For sequence B, Fig 2 shows that performance of the PF and EPF is similar. In this sequence, a moving man disappears behind a tree for few frames. In sequence C, the man disappears for larger number of frames. Fig. 3-(a) shows that the PF and EPF trackers fail around frame 220,
but EPF performs in different manner to catch the tracked object. EPF excite the particles from frame 220 searching for the object until the object appears again in frame 240. Fig. 3-(b) delineates why EPF displayed more robust performance in this case. EPF excites the particles to redistribute. It searches not only for the tracked object but also for an object with a meaningful weight with respect to the memorized weight.

Also, we evaluate the accuracy of the EPF tracker, comparing its position tracking performance against PF, PSO trackers and calculated Ground Truth as shown in Figures 4 and 5. The Ground Truth is taken every five frames. Also, we compare the error of the position in x and y coordinates from Ground Truth calculation. This error is computed according to the following equation:

$$E_{xy} = \sqrt{\left(\frac{x_g - x}{Lx}\right)^2 + \left(\frac{y_g - y}{Ly}\right)^2}.$$  

(14)

Where \((x_g, y_g)\) is Ground Truth coordinates and \((x, y)\) is the tracked centroid position. \((L_x, L_y)\) is the image width and height. Table 2 shows the maximum error in x and y positions for the three trackers in pixels.

These figures show the superior performance of PF over PSO. Moreover the time consumed to calculate the fitness function for the particle every iteration within the frame makes PSO slower than both PF and EPF algorithms.

VI- Conclusion

This work compares two powerful object tracking frameworks namely: PF and PSO for estimating parameters of non-linear and non-Gaussian models. An Enhanced Particle Filter (EPF) algorithm for object tracking is also introduced to enhance the robustness of the tracking algorithm. The EPF algorithm solves the sample impoverishment problem of the PF. Our experimental results demonstrate the superior performance of the PF and EPF over PSO in terms of robustness and occlusion handling. Our results additionally demonstrate that the EPF solves the sample impoverishment problem without any computational cost or time increase.

REFERENCES


[5] G. Tong, Z. Fang, and X. Xu, “A particle Swarm Optimization – Particle Filter for non-


<table>
<thead>
<tr>
<th>Table 1 EPF tracker Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1- Collect the prior knowledge for the moving object as centroid position, mean gray intensity level.</td>
</tr>
<tr>
<td>2- Initialize PF ( {x_i^{(s)}, w_i^{(s)}}_{k=1}^{N} )</td>
</tr>
<tr>
<td>3- For ( s=1: N )</td>
</tr>
<tr>
<td>- Sample ( X_i^{s} ) (Sample the position, intensity, and centroid block from the proposal distribution.).</td>
</tr>
<tr>
<td>- Evaluate the likelihood from equations 7, 8, 9, and 10.</td>
</tr>
<tr>
<td>4- End For</td>
</tr>
<tr>
<td>5- Evaluate importance weights.</td>
</tr>
<tr>
<td>6- For ( s=1: N ), Normalize weights ( W_i^{s} ), end.</td>
</tr>
<tr>
<td>7- Evaluate max. weights of LP and LI.</td>
</tr>
<tr>
<td>8- If max. weight &lt; 0.6 memorized max. weight.</td>
</tr>
<tr>
<td>- Excite the particles according to equation 13</td>
</tr>
<tr>
<td>- Set Excite indicator.</td>
</tr>
<tr>
<td>Else</td>
</tr>
<tr>
<td>- Clear Excite indicator</td>
</tr>
<tr>
<td>- memorized max. weight=max.(memorized max. weight, max. weight)</td>
</tr>
<tr>
<td>- Resample particles.</td>
</tr>
<tr>
<td>9- End If</td>
</tr>
<tr>
<td>10- If Excite indicator</td>
</tr>
<tr>
<td>- Output the last centroid of the tracked object</td>
</tr>
<tr>
<td>Else</td>
</tr>
<tr>
<td>- Output the centroid as the weighted mean particles position.</td>
</tr>
<tr>
<td>11- End If</td>
</tr>
<tr>
<td>12- Go to step 3.</td>
</tr>
</tbody>
</table>
Fig. 1. PSO, PF and EPF trackers performance with Sequence A
Fig. 2. PSO, PF and EPF trackers performance with Sequence B

Table 2 Comparison between PF, PSO and EPF maximum error

<table>
<thead>
<tr>
<th></th>
<th>PSO</th>
<th>PF</th>
<th>EPF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence A</td>
<td>45</td>
<td>33</td>
<td>30</td>
</tr>
<tr>
<td>Sequence B</td>
<td>8</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Sequence C</td>
<td>13</td>
<td>13</td>
<td>12</td>
</tr>
</tbody>
</table>
Fig. 3. PF and EPF trackers' performance with Sequence C. Figure (a) shows the failure of the PF tracker as the man in this sequence disappears behind a tree at frame 220. Figure (b) shows how EPF re-excites the particles until the tracked object is found after the man reappears.
Fig. 4. PSO, PF and EPF trackers against Ground Truth for Sequence A
Fig. 5. PSO, PF and EPF trackers against Ground Truth for Sequence C