Object-tracking based on Particle Filter using Particle Swarm Optimization with Density Estimation

Gongyi Xia and Simone A. Ludwig
North Dakota State University
Fargo, ND, USA
{gongyi.xia,simone.ludwig}@ndsu.edu

***Abstract*—Visual object-tracking is a fundamental task applied in many applications of computer vision. Many different tracking algorithms have been used ranging from point-tracking, to kernel-tracking, to silhouette-tracking based on different appearance models chosen. This paper investigates the particle filter that is used as a tracking algorithm based on the Bayesian tracking framework. The problems that the particle filter tracking technique suffers from are degeneracy and the impoverishment degradation. These two issues are addressed by the use of Particle Swarm Optimization (PSO) as the sampling mechanism. In particular, particles are generated via the PSO process in order to estimate the importance distribution. Two density estimation methods are used, one is a parametric method using the Half-Normal distribution fitting, and the other is a non-parametric method using kernel density estimation. The experiments revealed that the non-parametric density estimation method combined with PSO outperforms the other comparison algorithms.**

I. Introduction

Visual object-tracking is a very active research topic within the area of computer vision. One of the reasons is the increasing need for the automated analysis of videos. There are usually three steps involved in video analysis namely detection of interesting moving objects, tracking such objects from frame to frame, and the analysis of object tracks to recognize their behavior.

The aim of object-tracking is to estimate the target state within a video sequence including the trajectory, orientation, position and scale, etc. [1]. In particular, the tracker labels the area of interest in a video frame as the target based on the previous knowledge of the target.

Object-tracking is applied to many areas including [2]:

1) motion-based recognition whereby human identification is performed based on gait, etc.;
2) automatic surveillance whereby a scene is monitored to detect suspicious activities or unlikely events;
3) video indexing, whereby the videos in multimedia databases are automatically annotated;
4) human-computer interaction such as gesture recognition, eye gaze tracking, etc.;
5) traffic monitoring in order to gather real-time statistics of traffic to direct traffic flow;
6) vehicle navigation using video-based path planning and obstacle avoidance.

Object-tracking is defined as the problem of estimating the trajectory of an object in the image plane moving around in a scene. Thus, a tracker has to assign a consistent label to the tracked objects in each frame of a video. The challenges of the tracking of objects arise due to the following issues:

- noise in images,
- complex object motion,
- partial and/or full occlusion,
- scene illumination changes,
- complex object shapes,
- non-rigid or articulated nature of objects.

Object-tracking can be simplified by placing some constraints on the tracking task, which are based on the motion and appearance of objects. Most trackers assume that the object motion is smooth without abrupt changes. Furthermore, the object motion can be set to be of constant velocity or constant acceleration based on prior information. Moreover, prior knowledge about the number and size of the objects as well as the object appearance and shape can further simplify the tracking task.

Numerous object-tracking approaches have been proposed. These approaches differ from each other in the following ways:

- object representation;
- image features used;
- model to be used for motion, appearance and shape of object;

Based on the appearance models, object-tracking algorithms can be categorized as either generative [3]-[7] or discriminative [8]-[12]. The generative algorithms usually learn a model to represent the target object, and then use this model to search for the image region with minimal reconstruction error. On the other hand, the discriminative algorithms convert the tracking problem to a binary classification task in order to find the decision boundary for separating the target object from the background.

Object-tracking is a challenging task since many different circumstances play a role. For example, some trackers might be good in handling different illuminations, but might have

978-1-4799-7492-4/15/$31.00

difficulties in tracking objects when the appearance of the object changes due to variations of object movement and rotation. Another set of trackers might predict motion in order to better track speed, but then others might have difficulty following a bouncing object. Furthermore, a tracker may be based on the detailed assumption of the appearance, but then might fail on an articulated object [13].

This paper investigates particle filter based trackers, which belong to the generative algorithm category. Given that in order to model the underlying dynamics of a physical system such as the object-tracking task, it is important to include both elements of non-linearity and non-gaussianity. Thus, for such cases particle filters can be used. Particle filters are sequential Monte Carlo methods that are based on the point mass representations of probability densities that are applied to any state model. Therefore, particle filters can be used for object-tracking by approximating the filtered posterior distribution by a set of weighted particles. The particles are weighted based on a likelihood score, and thus, propagates these particles according to a motion model. One drawback the use of particle filters in object-tracking exhibits is the inability of samples to explore the probability distribution effectively and efficiently. This is due to the particles that do not move according to their former experience nor do they consider the relationship to other particles. Thus, in this paper we propose enhancements to the marginal particle filter by applying Particle Swarm Optimization (PSO) [14] and kernel-based estimation.

The paper is structured as follows. Section II describes related work in the area of object-tracking. In Section III the two proposed approaches are described. Section IV describes the experiments conducted and discusses the results and findings. The conclusion and future work is given in Section V.

II. Related Work

The visual object-tracking problem in the context of Bayesian inference can be formulated by two models: system model and measurement model. The system model given in Eq. (1) represents the dynamic transition of the state as the time proceeds; and the measurement model (Eq. (2)) represents the observation of the state through the noisy measurement process:

$$x_k = f_k(x_{k-1}, v_{k-1}) \tag{1}$$

$$z_k = h_k(x_k, n_k) \tag{2}$$

where the sequence $\{x_k, k \in N\}$ denotes the target state, the sequence $\{z_k, k \in N\}$ denotes the measurement of the state sequence, and v_k and n_k are system noise and measurement noise, respectively. In the context of object-tracking, x_k is the target position. Depending on the specific application, the target position could include coordinates of the target center, in-plane rotation, or aspect ratio, etc. z_k is the measurement for the image at frame k in the video sequence.

The goal of Bayesian inference is to estimate x_k based on the prior knowledge about the system and the measurement process and all measurements $z_{1:k} = \{z_i, i = 1, 2, \ldots, k\}$ up to time k. By the criteria of the Minimum Mean-Squared Error (MMSE), the optimal estimation is the expected value from the posterior probability density [15].

In the case of Gaussian noise and linear system dynamics, Kalman filters are optimal estimators [16]. For most of the applications in visual object-tracking, however, the system and measurement models can hardly be approximated as linear. In such cases, the particle filter successfully solves the problem of non-linearity by using particles to approximate a distribution, i.e., Monte Carlo approximation to avoid the ever-increasing dimension of the sampling space by sequential importance sampling. In the particle filter, the weights for particles are calculated recursively as in Eq. (3), where $p(z_k|x_k)$ denotes the likelihood, $p(x_k|x_{k-1})$ reflects the system motion, and $q(\cdot)$ is the importance density. After obtaining weights for particles, the estimation of the posterior density is approximated as in Eq. (4).

$$w_k^i \propto w_{k-1}^i \frac{p(z_k|x_k)p(x_k|x_{k-1})}{q(x_k|x_{0:k-1}, z_{1:k})} \tag{3}$$

$$p(x_{0:k}|z_{1:k}) = \sum_{i=1}^{N} w_n^i \delta_{X_{1:n}^i}(x_{1:n}) \tag{4}$$

where w_k^i is the weight for particle i at time k and $\delta(\cdot)$ is Dirac delta function.

A generic particle filter suffers from several problems that are inherent in the particles. The first issue resulting from the generic particle filter is degeneracy [17][18], where a large portion of the particles have negligible weight after a few iterations. Thus, these particles contribute very little towards the distribution approximation because of their small weight.

An intuitive yet effective solution is Sampling Importance Resampling (SIR), where particles of small weights have a smaller chance to be forwarded to the next iteration. Typical implementations of SIR include bootstrap particle filter [17], and reject particle filter [19]. While alleviating the degeneracy problem, however, the resampling may lead to the impoverishment problem, where the diversity of particles are reduced especially in the case of small process noise. Multiple research has been carried out to alleviate the impoverishment problem. The Regularized Particle Filter proposed in [21] creates a continuous posterior density by applying the kernel smoothing to the discrete posterior density. Then, particles resampled from the continuous posterior density are distributed more evenly. The Auxiliary particle filter in [20] avoids resampling from particles that have low likelihood by constructing an importance density that has high conditional likelihood.

There are other methods which cope with the degeneracy problem without giving rise to the impoverishment problem. These methods move particles away from low posterior probability regions towards regions with higher probability. The Kernel Particle Filter proposed in [22] employs Mean-Shift to migrate particles towards higher probability locations on the posterior probability landscape. However, the Mean-Shift, as a deterministic search algorithm, needs a continuous

probability distribution, which is constructed by the kernel density estimation [22].

PSO, as an optimization algorithm, is also employed by some researchers formulating the object-tracking as an optimization problem. A hierarchical version of the PSO algorithm (HPSO) is developed in [23] to track articulated human motion. The authors claim that HPSO is superior to the particle filter and related approaches in terms of tracking results due to the elimination of the needs for a specific motion model. Feng et al. in [24] proposed an enhanced PSO for object-tracking where particles are assigned different inertia parameters according to their local best quality avoiding getting stuck in a local optimum.

Recently, there are research investigations trying to integrate PSO with the particle filter to overcome the degeneracy problem mentioned above. The authors in [25] use particles sampled from the importance density using PSO. With the set of particles moved to high likelihood regions, the authors claim that the particle impoverishment is avoided. A similar algorithm was proposed in [26], in which PSO was used to move samples to a region where both the likelihood of the target and the prior density are significant. Another similar PSO based particle filter is proposed in [27], where the so-called multi-layer importance sampling method is used to move particles towards the region with high likelihood of observation. However, all these algorithms suffer from the same problem even though these samples are originally propagated forward through the system motion model. Since the particles are moved by PSO, their distribution does not fit the system motion model anymore. In other words, after applying the PSO process to the particles the underlying system motion model is modified to accommodate such transitions of the particles.

It is important to note, if the importance density $q(\cdot)$ in Eq. (3) is always identical to the system motion $p(x_k|x_{k-1})$, then above mentioned PSO based particle filters fall back to the PSO-only tracking algorithm similar to the ones in [23] and [24]. This operation effectively eliminates the system motion model from the tracking process, thus, could potentially increase the tracking performance according to [23]. However, eliminating the system motion model does not necessarily improve the performance. This is only true if the system motion model is vague and can not reflect the object motion effectively. In this case, the system motion has a negative influence. On the other side, discarding the system motion from the tracking process means that this information is completely lost, which would positively contribute to a better tracking performance if a more refined system motion can be derived.

III. Proposed Approach

The Marginal Particle Filter is used instead of the conventional particle filter to allow arbitrary particle sampling within the state space without modifying the effective system motion model.

A. Marginal Particle Filter

As discussed in [28], the importance sampling scheme draws samples, i.e., particles, according to the importance density in a joint state space. In practice, the sampling is done in a sequential manner such that direct sampling in the high dimensional joint space can be avoided. Each sample x_k^i drawn in the current state space x_k by sequential importance sampling together with its previous sample $x_{0:k-1}^i$ constitute a sample in the joint space χ^k, where $i = 1, 2, \ldots N$ and N is the number of particles. It is clear that as the path grows, the dimension of the sampling space also increases. This situation usually leads to the problem of degeneracy of the weights, and this leads to the variance of the weights to increase without bound [28].

The curse of high dimensionality of the sampling space is inherited from the joint posterior density $p(x_{0:k}|z_{1:k})$ that is being approximated by the particles. If we can reduce the dimension of the posterior density, then we could avoid problems induced by the high dimensional sampling space. The authors in [28] presented a Marginal Particle Filter (MPF), in which only the marginal posterior density $p(x_k|z_{1:k})$ is estimated. Thus, this essentially eliminates the need for sampling in a high dimensional joint space.

The prediction and update steps in the marginal particle filter are essentially the same as in the particle filter, which are shown in Eq. (5) and Eq. (6).

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})\mathrm{d}x_{k-1} \quad (5)$$

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{(p(z_k|z_{1:k-1})} \quad (6)$$

Substituting Eq. (5) into Eq. (6), the update step becomes:

$$p(x_k|z_{1:k}) \propto p(z_k|x_k) \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})\mathrm{d}x_{k-1} \quad (7)$$

Since $p(x_{k-1}|z_{1:k-1})$ is already approximated by a set of particles $\{x_{k-1}^i, w_{k-1}^i\}$, the approximation of $p(x_k|z_{1:k-1})$ is derived from:

$$p(x_k|z_{1:k-1}) = \sum_{i=1}^{N} w_{k-1}^i p(x_k|x_{k-1}^i) \quad (8)$$

Then, the posterior density can be estimated by:

$$p(x_k|z_{1:k}) \propto p(z_k|x_k) \sum_{i=1}^{N} w_{k-1}^i p(x_k|x_{k-1}^i) \quad (9)$$

And the importance weights are updated by:

$$w_k = \frac{p(x_k|z_{1:k})}{q(x_k|z_{1:k})} \quad (10)$$

where $q(x_k|z_{1:k})$ is an arbitrarily chosen importance density to draw samples. By deliberately designing the importance density, the particles drawn can be distributed in the region where the likelihood is significant.

B. Sampling by PSO

As suggested above, to mitigate the problem of degeneracy the key is to derive particles that have significant posterior probability. In [25] and [26], this is done by moving particles through the PSO process. However, such abrupt manipulation of particles completely eliminates the connection of particles established by particle propagation between consecutive frames, thus, results in ignorance of system motion information during the tracking process of the particle filter.

An ideal importance density should yield particles that minimize the variance of the importance weights [29]. That is equivalent to finding an importance density that matches the posterior density. The marginal particle filter does not necessarily need particle propagation between consecutive frames. Thus, it can freely sample particles from the current state space x_k based on any importance density.

An intuitive solution is to define an importance density that is identical to the posterior density. Since the posterior density is unknown while sampling, and usually is impossible to fully derive, we use the density of the likelihood as the basis to perform the importance sampling. Even though it is still costly to obtain the density of the likelihood for the entire state space. In this paper, we choose a "reverse direction" to accomplish this. First, we locate the high likelihood regions and scatter the particles in such regions. Then, we estimate the underlying importance density from the current particles and in turn calculate the importance probability for each particle based on the estimated importance density.

1) Particles sampling: In this section, we present the method to distribute particles in high likelihood regions for the use with the particle filter. We use the canonical PSO process to explore the state space and use the likelihood as the fitness value for the PSO process. Upon convergence, the particles should move towards and finally gather around the convergence position.

Let us assume the posterior $p(x_{k-1}|z_{1:k-1})$ for time step $k-1$ is available, and we are proceeding to the next frame. The particle sampling at time step k is done by the following steps:

Initialize: Particles are distributed across the possible region in the search space as in general PSO. Depending on the implementation, this region could be around the optimal estimation of the target position at time step $k-1$.

Iterate: Let the particles move according to the PSO iteration equations, whereby the particles in each iteration are denoted as $\{x_k^{i,j}\}_{j=1}^N$, i is the index of the current iteration, and N is the swarm size.

Converge: If the convergence criteria are met (the details will be discussed later in this paper), then the iteration is stopped. If the maximum number of generations has been reached before the convergence criteria are met, then this particular execution of sampling has failed.

Combine: Place particles from all iterations into the same set, denoting the set as $\{x_k^i\}_{i=1}^T$, where $T = M * N$ is the number of total particles, M is the number of iterations, and N is the swarm size of the population.

2) Importance Density Estimate: The importance density is empirically estimated from the aggregated particles $\{x_k^i\}_{i=1}^T$ obtained above, which is then used to update the particle weights. This can be done either in a parametric or non-parametric manner.

Parametric density estimation:

Here we assume that the distance of particles from their center conforms to the Half-Normal distribution. Its probability density function is given by:

$$p(y) = \frac{\sqrt{2}}{\sigma\sqrt{\pi}}\exp\left(-\frac{y^2}{2\sigma^2}\right) \qquad \text{y} > 0 \tag{11}$$

The reason we chose the Half-Normal distribution is because it only depends on one parameter, which introduces less deviation during the method of finding the maximum likelihood. First, the center of particles is obtained by retrieving the mean of all the particles. Then, the Euclidean distance between the particles and the center are calculated. Finally, the parameter σ can be estimated by:

$$\hat{\sigma} = \sqrt{\frac{1}{2}\sum_1^n d_i^2} \tag{12}$$

where d_i is the distance to the center for particle i, and n is the number of particles.

Since the fitting of the Half-Normal distribution depends on only one parameter (distance to the center), the posterior density is also symmetrical around the center. Which might not always be true, however, in this paper we hold on to such approximation in the case of the parametric estimation.

Non-parametric density estimation:

In order to relieve the restriction for the assumption of some known distribution and symmetrical posterior, non-parametric density estimation is also employed in this paper. This approach makes less rigid assumptions about the distribution of the observed data. In this paper, we use the *kernel density estimation* with the Gaussian kernel function [30]. Given the particle set $\{x_k^i\}_{i=1}^T$, its kernel density estimation is obtained by:

$$q(x_k|z_{1:k}) = \sum_{i=1}^{T} K_\lambda(x_t - x_k^i) \tag{13}$$

where K_λ is the scaled kernel, and λ is the kernel width.

Compared to the parametric approach, the kernel density estimation makes no assumption about the potential distribution of the particle set. Therefore, it adapts better to the multi-model likelihood situation.

In addition, the given particle set is further tailored before the estimation stage. For example, the particle set could be reduced by discarding particles whose likelihood are less than a certain threshold. This operation is meaningful in terms of both cost and performance. Usually, particles with small likelihood contribute to the posterior density little. Excluding them from the subsequent process saves unnecessary computations. On the other hand, a smaller particle set reduces the variance of the particle weights.

C. Marginal Particle Filter with PSO

Our PSO based particle filter differs from other PSO based particle filters (mentioned in Section II) in two ways:

- marginal particle filter is used to allow a fixed-dimension state space;
- particles are drawn first in the high likelihood region before the importance density is derived.

These differences bring performance improvements to our particle filter in the following ways:

1) instead of using only the last generation of particles, our approach considers all generations, which increases the number of available particles dramatically by several folds without extra computational expense;
2) the particle set covers a much broader area including the area around and further away from the convergence position. In contrast, the last generation of PSO usually gathers close around the convergence position;
3) adaptive to multi-model likelihood density;
4) able to globally search for high likelihood particles instead of only deriving particles from the last generation.

Algorithm 1 shows the pseudo code of general form of our particle filter.

Algorithm 1 Marginal Particle Filter with PSO

Input: Particle set from previous frame $\{x_{k-1}^{(\cdot)}, w_{k-1}^{(\cdot)}\}$, measurement z_k
Output: Target estimation $\hat{p}(x_k|z_{1:k})$

Generate particle set $\{x_k^i\}_{i=1}^T$ by performing PSO
if $Parametric\ Estimation$ **then**
  Estimate $\hat{\sigma}$ according to Eq. (12)
  for each x_k^i in the particle set **do**
    Calculate likelihood $p(x_k^i|z_k)$ based on appearance model
    Calculate $p(x_k^i|z_{1:k-1})$ based on Eq. (8)
    Obtain importance density estimation $q(x_k^i|z_{1:k})$ based on Eq. (11)
    Calculate particle weights $w_k^i = \frac{p(x_k^i|z_{1:k-1}) * p(x_k^i|z_k)}{q(x_k^i|z_{1:k})}$
  end for
else $Non\text{-}Parametric\ Estimation$
  Discard particles from $\{x_k^i\}_{i=1}^T$ whose likelihood is less than a designated value
  for each x_k^i in the particle set **do**
    Calculate likelihood $p(x_k^i|z_k)$ based on appearance model
    Calculate $p(x_k^i|z_{1:k-1})$ based on Eq. (8)
    Obtain importance density estimation $q(x_k^i|z_{1:k})$ based on Eq. (13)
    Calculate particle weights $w_k^i = \frac{p(x_k^i|z_{1:k-1}) * p(x_k^i|z_k)}{q(x_k^i|z_{1:k})}$
  end for
end if
Return target estimation $\hat{p}(x_k|z_{1:k}) = \sum w_k^i x_k^i$

IV. Experiments and Results

To demonstrate the performance of our object tracker based on marginal particle filter with PSO (MPFPSO), we evaluate the tracker on a video dataset comprised of a wide range of videos. In this section, we first examine the validity of our sampling method to verify that particles are sampled as expected and the importance density estimation results are also evaluated. Then, the performance of our object trackers is evaluated against other trackers including the particle filter with PSO (PFPSO) [26], the bootstrap particle filter (BPF) [31], and the basic marginal particle filter (MPF).

A. Experimental Setup

The experiments are implemented based on the object-tracking framework developed in [32]. The video set is also provided by this framework, which includes 51 videos of different scenes. All of the code is run in Matlab R2015 on a Windows machine equipped with Core i5@2.8GHz*4 Cores, 16GB RAM.

For other settings of the experiments, we use the appearance model and similarity measurement as defined in [33] because of its high resolution and peaked likelihood landscape. In this appearance model, objects are described by sparse representation of a dictionary which contains the vectorized pixels. The pixels are represented by its intensity and similarity is measured through l_1 minimization. To simplify the problem, we only track the target location in the image plane. The state is defined as $x_t = (x, y)$, where $\{x, y\}$ are the coordinates of the target center. The system motion model is assumed to be of Gaussian distribution $p(x_k|x_{k-1}) \sim N(x_{k-1}, \Sigma)$, where Σ is the diagonal covariance matrix.

The PSO process in the tracker is set up as a 'Standard PSO' [34] with the population size of the particle swarm set to 50. In this paper, we use this set of parameters throughout our experiments due to the observation that changes in the PSO parameters do not change the tracking performance significantly. The swarm of the particles are initialized uniformly within ± 32 pixel on both the horizontal and vertical axis around the previous target position. The stopping criteria used are:

1) the PSO process stops after 20 iterations, or
2) if the overlap ratio of the best bounding box of the previous 4 frames as compared to the current frame is at least 98%.

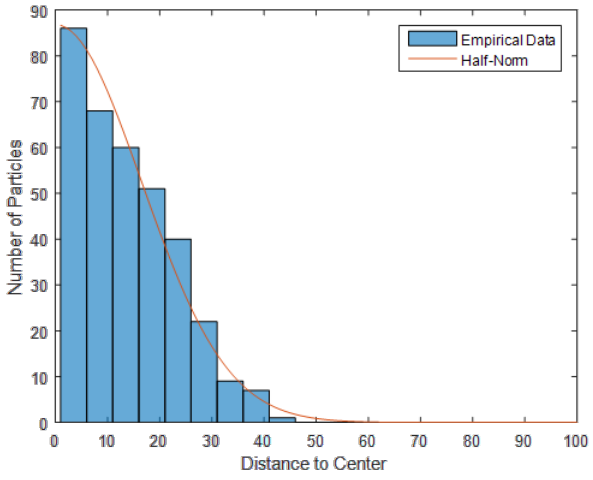
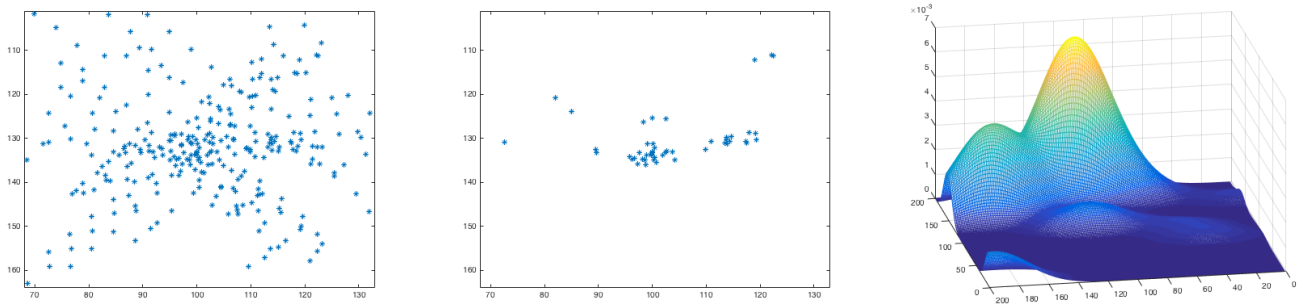


Fig. 1. Comparison of empirical data and Half-Normal Distribution.



(a) Aggregated particles on one frame. (b) Tailored particle set. (c) Results of kernel density estimation.

Fig. 2. Illustration of kernel density estimation.

B. Results

We run our object tracker on the video sequence named "CarDark", and we randomly choose a frame during the tracking process. For that particular frame, PSO converges at iteration 7, so there are in total 350 particles in the data set. Figure 1 shows that the histogram of the distance of particles to the center fits the Half-Normal distribution very closely while using parametric density estimation.

The non-parametric density estimation is performed with the kernel density estimation tool kit developed in [35]. Figure 2 shows how the density estimation is performed on particles drawn by the PSO process. The particles are drawn first by the PSO process as shown in Figure 2a, then the particle set is tailored according to the particle weights as displayed in Figure 2b. Figure 2c is an illustration of the kernel density estimation of the tailored particle set. It can be observed in Figure 2c that the tailored particle set consists of particles from all over the frame because their likelihood is above the threshold. It is worth to mention that some of the particles are not from the current iteration of PSO, and thus are relatively further away from the convergence position. This is particularly desirable in the case of a multiple-peaked likelihood where the last generation of particles is usually attracted to the highest peak. This is how the tailoring operation exploits the exploratory power of PSO.

1) Tracking Results: We use precision and success rate as in [32] for quantitative analysis of the tracking ability of our approach. In detail, precision measures the accuracy of tracking in terms of distance between the tracking result and the ground truth. Specifically, the distance is derived by calculating the Euclidean distance between the centers of the tracking result and the ground truth. Given $\{x, y\}_r$ is the center of the tracking window, and $\{x, y\}_g$ is the center of the ground truth, then the precision is obtained based on Eq. (14):

$$\text{precision} = \frac{N_{\|\{x,y\}_r - \{x,y\}_g\| \leq \text{threshold}}}{N_{total}} \tag{14}$$

where N_{total} is the number of total frames on which the tracking is performed, $N_{\|\{x,y\}_r - \{x,y\}_g\| \leq threshold}$ is the number of frames which is within given distance of threshold from ground truth and $\|\cdot\|$ represents the Euclidean distance. The success rate on the other hand measures the robustness of the tracking in terms of overlap between the tracking result and the ground truth. Given P_r is the set of pixels within the tracking result bounding boxes, and P_g is the set of pixels within the ground truth bounding boxes, whereby the success rate is given by Eq. (15):

$$\text{success rate} = \frac{N_{S \geq \text{threshold}}}{N_{total}} \tag{15}$$

where N_{total} is the total number of frames on which the tracking is performed, $N_{S \leq threshold}$ is the number of frames whose overlap score S is equal or greater than the given threshold, and $S = \frac{|P_r \cap P_g|}{|P_r \cup P_g|}$ is the overlap score between the tracking result and the ground truth, $|\cdot|$ measures the number of pixels in the pixel set.

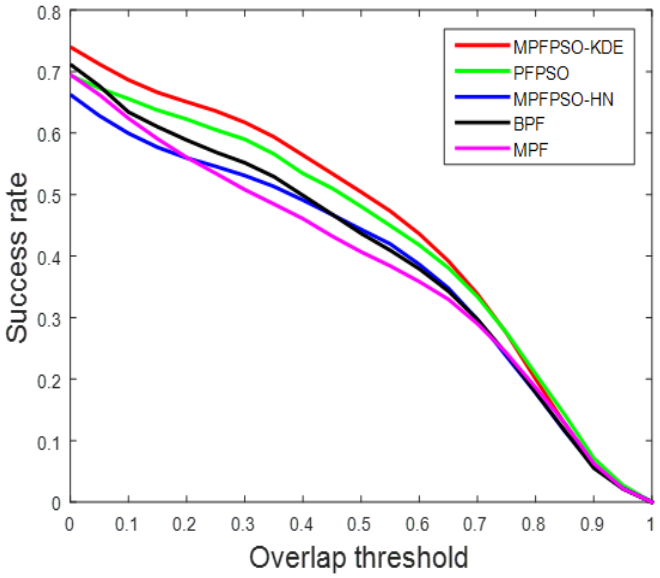


Fig. 3. Success rate versus overlap threshold.

In order to evaluate the tracking performance more thoroughly and alleviate potential dependencies on specific videos, both precision and success rate are measured on all 51 video sequences. As shown in Figure 3 and Figure 4, our MPFPSO

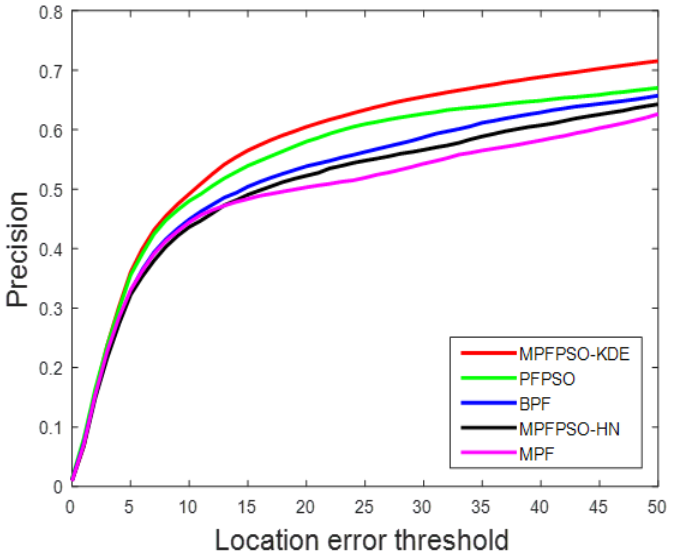


Fig. 4. Precision versus location error threshold.

with kernel density estimation (MPFPSO-KDE) performs best in terms of both precision and success rate, followed by PFPSO, MFPSO with Half-Norm (MFPSO-HN) fitting, BPF and MPF sequentially. The x-axis in both figures are thresholds as used in Eq. (14) and Eq. (15).

The performance margin that MPFPSO and PFPSO have brought against BPF and MPF comes mostly from the sampling mechanism. In MPFPSO-KDE and PFPSO, the particles are either sampled from or moved to high likelihood regions, which in turn heightens the matching degree between the importance density and the posterior density. For MPFPSO-HN it considers all the particles. Although the variance of the particle weights increases because of that, with the help of PSO it still outperforms basic MPF. MPFPSO-KDE is superior to PFPSO because of its adaptive particle sampling. In MPFPSO-KDE, only those particles with high likelihood are selected, thus, the number of selected particles is not necessarily equal to the PSO swarm size and they are not necessarily located at the convergence position. Generally, the size of the particle set varies according to the likelihood landscape and the PSO process. For example, in the case of narrow likelihood peaks and fast convergence of PSO, the size of the particle set could be relatively small. This way the particles are sampled to match the resulting importance density to the likelihood as good as possible. In contrast, PFPSO always uses the last iteration of particles from the PSO process. Depending on the convergence criterion, those particles from the last generation may distribute only around the convergence point, thus, ignoring the likelihood landscape anywhere else. As we discussed previously, this may lead to higher variance of the particle weights. Another advantage of MPFPSO compared to PFPSO is that it includes the system model during the tracking process.

Figure 5 shows the tracking results on the video sequence 'Boy'. The target in this sequence is fast moving and causes motion blur. It can be observed that both MPFPSO-KDE and PFPSO are more robust compared to the other tracking method.

2) Running Time: The running time is another performance measure for object trackers. We evaluate the running time by the number of frames per second (fps) a tracker can process in the video sequences. Although the running time depends on many factors, such as the appearance model, coding style, platform, etc., however, in our experiment, we fix as many factors as possible to focus on the difference among the trackers. Since we use the same platform and appearance model in the trackers, the discrepancy of the running time basically boils down to how the trackers sample the particles and how they are used.

TABLE I
RUNNING TIME AND AVERAGE PARTICLE NUMBER PER FRAME

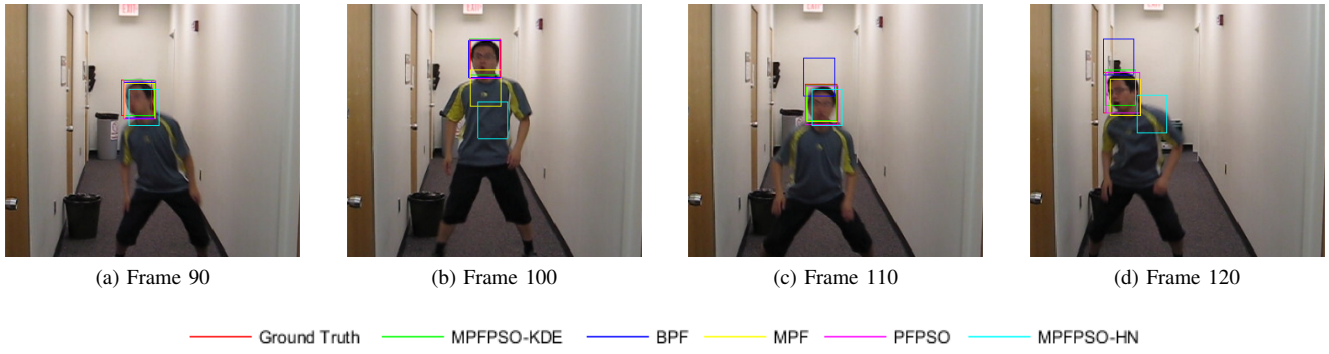
Trackers	fps	Particle number
MPFPSO-KDE	8.48	455.62
PFPSO	16.43	276.56
MPFPSO-HN	9.07	458.59
BPF	9.72	500
MPF	6.81	500

Table I shows the comparison of running time and average particle number. It can be observed that the running time is directly related to the average particle number per frame. PFPSO runs at almost double speed compared to MPFPSO because it generates only half of the particles. The reason behind this lies in the different initialization strategies for the PSO process. In PFPSO, the particles are initialized by the propagation from the previous generation which usually are more concentrated. Whereas in MPFPSO, the particles are initialized evenly in a larger area, thus, it takes more iterations to reach convergence. However, on the other hand the broader initialization inspires a wider search of the particles.

V. Conclusion

This paper first reviewed the current state of visual object-trackers with connection to evolutionary optimization, then discussed past research on the application of the particle filter and PSO applied to object-tracking and investigated their drawbacks. A novel particle filter has been proposed to overcome the problems identified by previous approaches, where a sampling by the PSO mechanism is used together with the marginal particle filter. Experiments and results demonstrated that the proposed method (MPFPSO-KDE) does improve the tracking performance without introducing extra computational cost.

Overall, it is shown that our sampling by PSO method works quite well together with the marginal particle filter. Its performance is better than the particle filter with PSO previously developed as well as the bootstrap particle filter and pure marginal particle filter. It is also demonstrated that the sampling by PSO method is an efficient yet effective sampling mechanism. This approach is capable of sampling particles in the high likelihood region while keeping the system



(a) Frame 90 (b) Frame 100 (c) Frame 110 (d) Frame 120

Fig. 5. Tracking results on video 'Boy'.

motion model the same. Our experiments also indicate that this approach works well in the case of multiple-peaked likelihood landscapes.

In the future, our marginal particle filter with sampling by PSO could be further investigated in the following ways: (1) accelerate convergence of PSO to reduce the number of particles further to save computational cost; (2) the motion model could be further enhanced to reflect the real target motion; (3) the relation between swarm size, tracking performance, and computational cost is yet to be examined in more detail.

REFERENCES

[1] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, A. V. D. Hengel. A survey of appearance models in visual object tracking. ACM transactions on Intelligent Systems and Technology (TIST), 4(4), 58, 2013.

[2] A. Yilmaz, O. Javed, M. Shah. Object tracking: A survey. ACM Comput. Surv. 38, 4, Article 13, 45 pages, 2006.

[3] M. Black, A. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. IJCV 38, 63-84, 1998.

[4] A. Jepson, D. Fleet, T. Maraghi. Robust online appearance models for visual tracking. PAMI 25, 1296-1311, 2003.

[5] D. Ross, J. Lim, R. Lin, M.-H. Yang. Incremental learning for robust visual tracking. IJCV 77, 125-141, 2008.

[6] H. Li, C. Shen, Q. Shi. Real-time visual tracking using compressive sensing. In: CVPR, pp. 1305-1312, 2011.

[7] X. Mei, H. Ling. Robust visual tracking and vehicle classification via sparse representation. PAMI 33, 2259-2272, 2011.

[8] S. Avidan. Support vector tracking. PAMI 26, 1064-1072, 2004.

[9] R. Collins, Y. Liu, M. Leordeanu. Online selection of discriminative tracking features. PAMI 27, 1631-1643, 2005.

[10] H. Grabner, M. Grabner, H. Bischof. Real-time tracking via online boosting. In: BMVC, pp. 47-56, 2006.

[11] H. Grabner, C. Leistner, H. Bischof. Semi-supervised On-Line Boosting for Robust Tracking. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 234-247. Springer, Heidelberg, 2008.

[12] B. Babenko, M.-H. Yang, S. Belongie. Robust object tracking with online multiple instance learning. PAMI 33, 1619-1632, 2011.

[13] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, M. Shah. Visual Tracking: An Experimental Survey, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 7, pp. 1442-1468, July 2014.

[14] J. Kennedy, R. Eberhart. Particle swarm optimization. In: Proc. of IEEE Int. Conf. on Neural Networks, pp. 1942-1948. IEEE Press, Piscataway, 1995.

[15] B. D. Anderson, J. B. Moore. Optimal Filtering, 1979.

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

[17] P. Green, Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination, Biometrika, vol. 82, no. 4, p. 711, 1995.

[18] J. Liu, R. Chen, Sequential Monte Carlo Methods for Dynamic Systems, Journal of the American Statistical Association, vol. 93, no. 443, p. 1032, 1998.

[19] J. Liu, R. Chen and W. Wong, Rejection Control and Sequential Importance Sampling, Journal of the American Statistical Association, vol. 93, no. 443, pp. 1022-1031, 1998.

[20] M. Pitt and N. Shephard, Filtering via Simulation: Auxiliary Particle Filters, Journal of the American Statistical Association, vol. 94, no. 446, p. 590, 1999.

[21] C. Musso, N. Oudjane and F. Gland, Improving Regularised Particle Filters, Sequential Monte Carlo Methods in Practice, pp. 247-271, 2001.

[22] C. Chang and R. Ansari, Kernel particle filter for visual tracking, IEEE Signal Processing Letters, vol. 12, no. 3, pp. 242-245, 2005.

[23] J. Vijay, E. Trucco, and S. Ivekovic. Markerless human articulated tracking using hierarchical particle swarm optimisation. Image and Vision Computing 28, no. 11 (2010): 1530-1547

[24] S. Feng, C. Bae, G. Liu, X. Zhao, Y. Y. Chung, and W. Yeh. A categorized particle swarm optimization for object tracking. In Evolutionary Computation (CEC), 2015 IEEE Congress on, pp. 2737-2744. IEEE, 2015

[25] G. F. Tong, Z. Fang, and X. H. Xu. A particle swarm optimized particle filter for nonlinear system state estimation., Evolutionary Computation, 2006. CEC 2006. IEEE Congress on, pp. 438-442, 2006.

[26] A. Klamargias, K. Parsopoulos, P. Alevizos and M. Vrahatis, Particle filtering with particle swarm optimization in systems with multiplicative noise, Proceedings of the 10th annual conference on Genetic and evolutionary computation - GECCO '08, 2008.

[27] X. Zhang, W. Hu, S. Maybank, Xi Li and M. Zhu, Sequential particle swarm optimization for visual tracking, 2008 IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[28] M. Klaas, N. D. Freitas, and A. Doucet. Toward practical N2 Monte Carlo: the marginal particle filter. arXiv preprint arXiv:1207.1396, 2012.

[29] A. Doucet, S. Godsill and C. Andrieu, Statistics and Computing, vol. 10, no. 3, pp. 197-208, 2000.

[30] B. W. Silverman, Density estimation for statistics and data analysis. Vol. 26. CRC press, 1986.

[31] N. Gordon, D. Salmond and A. Smith, Novel approach to nonlinear/non-Gaussian Bayesian state estimation, IEE Proceedings F Radar and Signal Processing, vol. 140, no. 2, p. 107, 1993.

[32] Y. Wu, J. Lim and M. Yang, Online Object Tracking: A Benchmark, 2013 IEEE Conference on Computer Vision and Pattern Recognition, 2013.

[33] X. Jia, H. Lu and M. Yang, Visual tracking via adaptive structural local sparse appearance model, 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012.

[34] D. Bratton, J. Kennedy. Defining a standard for particle swarm optimization. In Swarm Intelligence Symposium, 2007. SIS 2007. IEEE, pp. 120-127. IEEE, 2007.

[35] Kernel Density Estimation Toolbox for MATLAB. [Online]. Available: http://www.ics.uci.edu/~ihler/code/kde.html. [Accessed: 08- Jan- 2016].