PARTICLE SWARM OPTIMIZATION AND PARTICLE FILTER APPLIED TO OBJECT TRACKING

A Thesis
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Gongyi Xia

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science
Option: Operations Research

December 2015

Fargo, North Dakota

North Dakota State University
Graduate School

Title

PARTICLE SWARM OPTIMIZATION AND PARTICLE FILTER APPLIED TO OBJECT TRACKING

By

Gongyi Xia

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Simone Ludwig
Chair

Dr. Jun Kong

Dr. Yarong Yang

Approved:

12/21/2015 — Date

Dr. Brian Slator — Department Chair

ABSTRACT

The particle filter is usually used as a tracking algorithm in non-linear under the Bayesian tracking framework. However, the problems of degeneracy and impoverishment degrade its performance. The particle filter is thereafter enhanced by evolutionary optimization, in particular, Particle Swarm Optimization (PSO) is used in this thesis due to its capability of optimizing non-linear problems. In this thesis, the PSO enhanced particle filter is reviewed followed by an analysis of its drawbacks. Then, a novel sampling mechanism for the particle filter is proposed. This method generates particles via the PSO process and estimates the importance distribution from all the particles generated. This ensures that particles are located in high likelihood regions while still maintaining a certain level of diversity. This sampling mechanism is then used together with the marginal particle filter. The proposed method's superiority in performance over the conventional particle filter is then demonstrated by simulations.

ACKNOWLEDGEMENTS

First I would like to thank my advisor Dr. Simone Ludwig. Thanks for always being inspirational and encouraging to me. Then most importantly, my sincere thanks goes to my family. Your love is my safe harbor.

DEDICATION

To Emma.

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENTS iv

DEDICATION v

LIST OF TABLES viii

LIST OF FIGURES ix

1. INTRODUCTION 1

1.1. Object Tracking 1

1.2. Evolutionary Optimization 4

1.2.1. Particle Swarm Optimization 5

1.3. Motivation 6

1.4. Contribution 7

1.5. Organization of the Thesis 7

2. PARTICLE FILTER FOR OBJECT TRACKING 8

2.1. Bayesian Tracking 8

2.1.1. Monte Carlo Approximation 11

2.1.2. Sequential Importance Sampling 11

2.1.3. Particle Filter 12

2.2. Particle Filter with PSO 14

3. OBJECT TRACKING BY MARGINAL PARTICLE FILTER WITH PSO 16

3.1. Marginal Particle Filter 16

3.2. Marginal PF with Sampling by PSO 18

3.2.1. Particles in the Particle Filter 18

3.2.2. Sampling by PSO 22

3.2.3. Marginal Particle Filter with Sampling by PSO 24

4. EXPERIMENTS AND RESULTS 27
4.1. Experiments with Sampling by PSO 27
4.2. Experiments and Results 32
4.3. Discussion 34
5. CONCLUSION AND FUTURE WORK 38
REFERENCES 39

LIST OF TABLES

Table Page

1. Comparison of the fps and the particle number. 34

LIST OF FIGURES

Figure		Page
1.	Object tracking reference model.	4
2.	Bayesian dynamic system	9
3.	Sequential importance sampling path [19]	16
4.	Illustration of an importance sampling example	18
5.	Particles at initialization step	19
6.	Particles during PSO iteration-1	20
7.	Particles during PSO iteration-2	20
8.	Illustration of convergence of particles	21
9.	Distribution of all particles in PSO	23
10.	Particles distribution on the image plane	29
11.	Comparison of empirical data and Half-Normal Distribution (histogram)	29
12.	Comparison of empirical data and Half-Normal Distribution (density)	30
13.	Bhattacharrya distance for each frame	31
14.	Aggregated empirical distribution vs Half Normal (Standardized)	31
15.	Comparison of performance: Precision rate	33
16.	Comparison of performance: Success rate	33
17.	Likelihood landscape	35
18.	Comparison of particle distribution	36
19.	Number of PSO iterations	37

1. INTRODUCTION

In this chapter, the definition of visual object tracking is introduced first. The major applications of this technology and the challenges are then briefly reviewed. After that evolutionary optimization, in particular, Particle Swarm Optimization (PSO) is introduced. PSO was chosen for this work because of its capability to work on complex problems such as visual object tracking. The overall organizational structure and major contribution of this thesis are then presented at the end of this chapter.

1.1. Object Tracking

Visual object tracking is one of the most active and challenging research topics in the area of computer vision. Its goal is to estimate the target state within video frames including trajectory, orientation, position and scale, etc. [1]. More specifically, the tracker labels an area in the video frames as target based on the previous knowledge of the target and the scene and possible constraints. This area indicates the projection of real target onto the 2D image plane. It represents what the object looks like from the perspective of the camera after all the variation of the background and target between the frames. The word “visual” here emphasizes that the only guaranteed input is a visible scene in the form of a 2D video record, which was captured by the camera previously or in real-time. Other types of sensors may not be equipped, therefore, inputs other than the aforementioned 2D video record are not always available, such as the velocity or distance from a radar, image of occluded target from an infrared camera and orientation from a gyroscope, although the availability of these inputs could improve the performance of tracking.

The ability of the visual object tracking serves as a fundamental component for many computer vision applications. As one of the most promising applications, automated surveillance is now deployed to perform tasks much more complicated than the traditional plain video

recording or monitoring. For example, object tracking can be used for suspicious activities detection in public spaces such as the detection of luggage left unattended in a public transportation facility [2], as well as for traffic statistics and path optimization. In these tasks, high-level analyses such as behavior analysis are performed based on the trajectory of the target extracted only by visual object tracking. Intelligent assist driving is another emerging application of computer vision, which enables vehicles to automatically make decisions on behalf of a driver to apply the brake in a critical situation to avoid a collision with pedestrians or other vehicles, or to keep the vehicle in the lane in case of a distracted or drowsy driver, or even to drive autonomously with minimal human intervention. In order to make the appropriate yet critical decisions, the computer needs to estimate the target position or motion state from the video augmented by other available auxiliary data. A "video-based lane estimation and tracking" (VioLET) system [3] is specifically designed for driver assistance including: Lane-Departure-Warning, Automated Vehicle-Control and Driver-Attention Monitoring. The system works exactly as a visual object-tracking task by relying majorly on a vision sensor because of its accurate and real-time video information in contrast to GPS and other types of sensors, which often need extra infrastructure or prior knowledge of the road such as map data. More computer vision applications that rely on visual object tracking include human-computer interaction, mobile robot navigation, video indexing, etc.

However, depending on the deployment scenario, visual object tracking could be very complex and challenging because of the presence of so many adverse issues of real-world application. Essentially videos are captured by a camera which usually only records 2D images; i.e., information about a 3D real world is compressed into 2D images, which in turn implies great

and inevitable information loss. Besides the inherent loss of information, other adverse facts include:

- background changes in color and texture;
- scene illumination changes;
- rotation and movement of the object;
- shape change for non-rigid object;
- partial and full object occlusions;
- measurement noise.

In order to overcome these negative conditions and derive the object state as accurate as possible, multiple visual object-tracking algorithms have been developed. Basically, a tracking algorithm can be decomposed into two major parts: object appearance model and object localization.

In natural language, the appearance of an object can be described by the object's visible characteristics such as shape, color, texture, etc. For instance, a book is a rectangular shape and its color is the color of the cover. Similarly, a computer uses an appearance model to describe an object. The visible characteristics such as color, edge, optical flow, texture, and shape are basic elements used to construct the appearance model. Usually, the appearance models can be roughly categorized into two types: generative model, and discriminative model. In the generative model, the appearance of the target is modeled such that the similarity between the candidates and the target can be effectively measured. Examples include Gaussian mixture model [5], color histogram [6], and principal component analysis [7]. The discriminative model constructs a classifier that labels the candidate as either target or non-target. Examples include SVM based classifier [8], boosting-based classifier [9], etc.

With the appearance model defined, the object localization estimates the most possible target position by applying the object appearance model on the current frame. Other constraints such as a target motion model may also be taken into account to improve the accuracy. A generic procedure for doing the localization consists of two steps: generation of a set of candidates, and selection of the best one. For the generative model, the one with the highest similarity with the target is chosen, and in discriminative model, the one with the highest confidence as determined by the classification is chosen. However, it is not necessary for each tracker to go through these steps. For example, for the particle filter the estimation is made by calculating the weighted sum of all particles.

Without loss of generality, the object-tracking process can be depicted as shown in Figure 1.

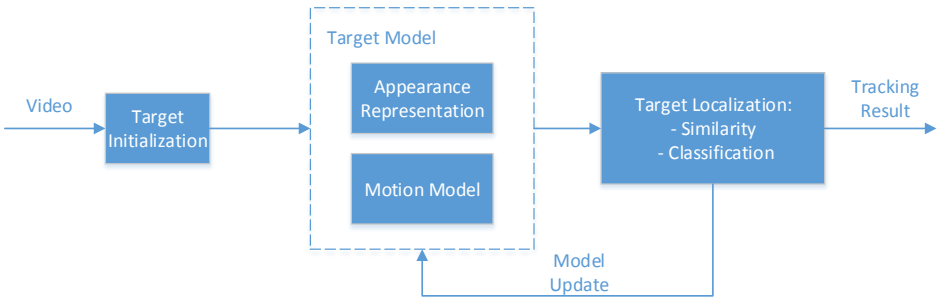

Figure 1. Object tracking reference model.

1.2. Evolutionary Optimization

Object localization in the tracking process can also be formulated into an optimization problem, where the objective function is defined as similarity or confidence as described in the appearance model. The goal is to find an area that maximizes the objective function, i.e. yields peak similarity or confidence. However, because of the complexity of the appearance model, the objective function is usually very complex and cannot be described in analytical form. Therefore,

the applicability of the conventional optimization methods such as linear programming and goal programming is not possible. This situation draws attention to the evolutionary optimization methods that are based on stochastic processes applied to object tracking.

Evolutionary algorithms borrow ideas from natural evolution and adaptation to solve complex often non-linear or non-differentiable optimization problems. Evolutionary algorithms identify themselves by its two distinct features [10]: population of individual solutions, and individual solutions renew themselves towards optimality by exchanging information with their peers. Its output (fittest individual solution) is generated by iterations of natural selection upon the population. Typical evolutionary algorithms used for optimization include genetic algorithm (GA), particle swarm optimization (PSO), differential evolution (DE), etc. In this thesis, PSO is employed to help to optimize the object localization problem.

1.2.1. Particle Swarm Optimization

Particle swarm optimization (PSO) was first introduced in [11] as a population based stochastic optimization method. As other evolutionary algorithms, PSO was inspired by some natural phenomenon. PSO mimics the social behavior that is often observed in bird flocking by adopting the concept of swarm in which individuals can communicate with their peers. In a swarm (bird flock), each individual (bird) represents a solution (a position) and each solution corresponds to a fitness value (distance to the food). An individual's movement is influenced by both its own experience and its peers' knowledge. Eventually, the whole swarm is likely to converge at an optimal position.

Formally, the social behavior within PSO can be represented as follows:

$$v_{id}^{t+1} = w v_{id}^{t} + c_1 p_1 (p_{id}^{t} - x_{id}^{t}) + c_2 p_2 (p_{gd}^{t} - x_{id}^{t}) \quad (1)$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \quad (2)$$

where v_{id}^{t} represents the velocity of individuals, p_{id}^{t} represents the best known position an individual has explored, p_{gd}^{t} represents the global best known position exchanged between the peers, w is the inertia coefficient, and c_1, c_2 are weights.

It has been shown that PSO can search very large candidate spaces for solution with very few assumptions about the problem being optimized. Moreover, since PSO has few parameters, it can be used in a broad range of applications without major modifications once it has been set up.

1.3. Motivation

Although intensive research has been done on the topic of visual object tracking, the application of evolutionary optimization on this issue has not been studied thoroughly. For example, both the particle filter and PSO perform very well in on-linear non-Gaussian scenarios, however, researchers tend to use only either of them for performing the object localization.

Several papers have attempted to use particle filter together with PSO, but failed to make the best use of PSO [16][17][18]. In these attempts, particle filter was applied on the joint space of target states from all the frames up to the current, and the sampling mechanism still follows the classic method and fails to take full advantage of PSO since it only moves particles that are drawn from the importance distribution. Additionally, after finishing a PSO iteration, most of the particle generations are discarded except the last generation. In this thesis, a novel visual object-tracking method is proposed to take full advantage of both the particle filter and PSO for object localization. The issues mentioned above are addressed by the proposed algorithm.

1.4. Contribution

In this thesis, an object-tracking algorithm based on the Bayesian statistical inference framework is proposed. It differentiates itself from other algorithms in the literature in the following ways:

- It adopts marginal particle filter for visual object-tracking where the state space is only for the current frame instead of using the joint space for all the frames.
- PSO is used as a sampling mechanism, which means the sampling is no longer drawn from an arbitrary distribution.
- In contrast to the traditional use of the particle filter, all generations of particles during the PSO procedure are used for the calculation of the optimal solution.

1.5. Organization of the Thesis

Chapter 1 provides an overview of visual object-tracking and introduces evolutionary optimization. In Chapter 2, the object-tracking algorithms employing particle filter and PSO simultaneously are discussed as well as their shortcomings. Chapter 3 describes in details the proposed algorithm introducing a marginal particle filter that is incorporated within PSO. The simulations and results are presented in Chapter 4. Finally, the conclusion and future work are explained in Chapter 5.

2. PARTICLE FILTER FOR OBJECT TRACKING

In this chapter, the Particle Filter is discussed. We first review the Bayesian tracking framework and mention that it is impossible to develop an analytical form of the Bayesian tracking in non-linear situations. Then, techniques such as Monte Carlo approximation and sequential importance sampling are introduced as corner stone of the later introduced particle filter. Finally, major steps for applying the particle filter to object tracking are discussed as well as the review regarding the usage of the particle filter combined with PSO.

2.1. Bayesian Tracking

In this section, the object-tracking problem is modeled in a Bayesian dynamic system, where two models are defined: system model and measurement model. In which the system model represents dynamic transition of the state as the time proceeds; and the measurement model represents the observation of the state through the noisy measurement process. Formally, the system model is given by:

$$x_k = f_k(x_{k-1}, v_{k-1}) \tag{3}$$

where the state sequence $\{x_k,\ k \in N\}$ denotes the target state, the sequence $\{v_k,\ k \in N\}$ denotes i.i.d. process noise, and f_k is a non-linear transition function of x_{k-1}. N is the set of natural numbers. The measurement model is given by:

$$z_k = h_k(x_k, n_k) \tag{4}$$

where the sequence $\{z_k,\ k \in N\}$ denotes the measurement of the state sequence, the sequence $\{n_k,\ k \in N\}$ denotes i.i.d. measurement noise, and h_k is a non-linear measurement function of x_k. The complete Bayesian dynamic system is depicted in Figure 2.

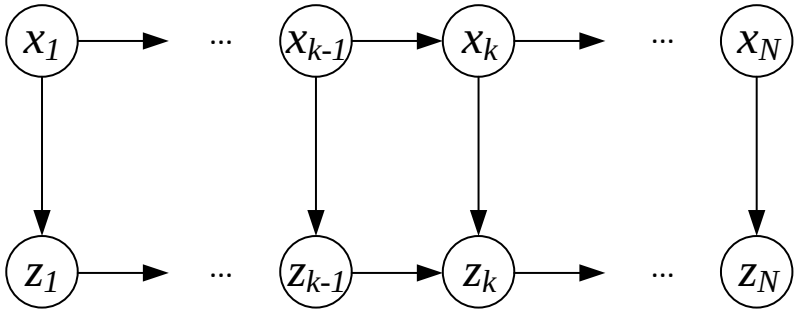

Figure 2. Bayesian dynamic system

The goal is to estimate x_k based on the prior knowledge about the measurement and the process noise and all measurements $z_{1:k} = \{z_i, i = 1,2, \dots, k\}$ up to time k. In the Bayesian tracking context, that can be expressed as building the posterior probability density function (PDF) of the state x_k based on the prior knowledge and measurements. Then, an optimal estimation of x_k is inferred by calculating the expected value from the posterior probability density constructed as above.

Usually, Bayesian tracking is performed in a recursive manner such that the computer does not have to always store the entire data set. Instead, upon receiving new data, a new estimation of the posterior probability density is obtained and this estimation is in turn used for the next data. In details, in order to obtain the posterior probability density denoted as $p(x_k|z_{1:k})$ at time *k* given the measurement data up until *k*, $z_{1:k}$, the whole Bayesian tracking is essentially divided into two steps: prediction and update.

In the prediction stage, as the name suggests a prediction of the state x_k is obtained in this step denoted as $p(x_k|z_{1:k-1})$. Particularly, it can be expressed via the Chapman-Kolmogorov equation as:

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

where $p(x_{k-1}|z_{1:k-1})$ is the posterior distribution from the previous time-step and $p(x_k|x_{k-1})$ is derived from the system transition model. It is assumed that the initial PDF $p(x_0|z_0)$ is available where z_0 is defined as no measurement.

In the update stage, upon receiving the measurement z_k at time step k, the posterior probability density $p(x_k|z_{1:k})$ is updated by the Bayes' theorem:

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

where $p(z_k|z_{1:k-1})$ is a constant given z_k has determined, thus, it functions as the normalization denominator. $p(z_k|x_k)$ measures the likelihood that the current observations is the true measurement of the predicted state, and is used to update the prior distribution. Usually this likelihood can be derived from the similarity or confidence level output obtained by the appearance model.

The prediction and update represents the generic and analytical solution to the Bayesian dynamic system. However, in many practical cases of visual object tracking, the system model or measurement model are highly non-linear because of the complexity of the video capturing process and the object movement. This makes the posterior density function very difficult to derive. It could be even worse in cases where the posterior density function is non-standard, thus, there is no analytical PDF representation at all.

To cope with such situations, an original particle filter was proposed in [12]. It is a sequential Bayesian filtering method based on random sampling known as Sequential Importance Sampling (SIS). The major steps in SIS are discussed in the following sections.

2.1.1. Monte Carlo Approximation

First, a PDF is no longer always in analytical form, instead it can be approximated by the Monte Carlo approximation using a group of samples called particles. For a generic PDF $\pi_n(x_{1:n})$ with a given *n*, its Monte Carlo approximation can be expressed as:

$$\hat{\pi}_n(x_{1:n}) = \frac{1}{N}\sum_{i=1}^{N} \delta_{X_{1:n}^i}(x_{1:n}) \tag{7}$$

where $x_{1:n}^i \sim \pi_n(x_{1:n})$, $i = 1,2, \dots N$ are N independent random variables drawn from the generic PDF $\pi_n(x_{1:n})$, and δ_x denotes the Dirac delta mass located at *x*.

The Monte Carlo approximation avoids the need for the analytical form, which is often difficult to obtain for a distribution. However, this comes with two major problems [14]:

- Sometimes it is difficult to directly sample from $\pi_n(x_{1:n})$, for example, it may not have an explicit form.
- The computational complexity of such a sampling scheme is ascending along with the dimensional increase of the joint space. At the latter phases of the tracking, *n* could become very large.

2.1.2. Sequential Importance Sampling

Importance Sampling is introduced to handle the first issue. Generally, it is circumvented by sampling from an importance density $q_n(x_{1:n})$ instead. Suppose $x_{1:n}^i \sim q_n(x_{1:n})$, $i = 1,2, \dots N$ are N independent random variables drawn from the $q_n(x_{1:n})$, then the generic PDF $\pi_n(x_{1:n})$ is approximated as:

$$\hat{\pi}_n(x_{1:n}) = \sum_{i=1}^{N} W_n^i \delta_{X_{1:n}^i}(x_{1:n}) \tag{8}$$

where

$$W_n^i = \frac{w_n(X_{1:n}^i)}{\sum_{j=1}^{N} w_n(X_{1:n}^j)} \tag{9}$$

$$w_n(x_{1:n}) = \frac{\pi_n(x_{1:n})}{q_n(x_{1:n})} \tag{10}$$

By carefully selecting the importance density $q_n(x_{1:n})$, the samples can now be drawn even though $\pi_n(x_{1:n})$ is not explicitly described.

The authors in [14] point out that the computational complexity to perform sampling on $\pi_n(x_{1:n})$ is at least linear with *n*, which is the dimension of the joint state space, which is analogous of the number of frames up to now in the context of object tracking. It has been shown that the method to solve this problem stems from the factorization of the importance density. Factorizing the importance density and $w_n(x_{1:n})$ such that:

$$\begin{aligned} q_n(x_{1:n}) &= q_{n-1}(x_{1:n-1}) q_n(x_{\mathrm{n}}|x_{1:n-1}) \\ &= q_1(x_1) \prod_{k=2}^{n} q_k(x_{\mathrm{k}}|x_{1:k-1}) \end{aligned} \tag{11}$$

and

$$\begin{aligned} w_n(x_{1:n}) &= \frac{\pi_n(x_{1:n})}{q_n(x_{1:n})} \\ &= \frac{\pi_{n-1}(x_{1:n-1})}{q_{n-1}(x_{1:n-1})} \frac{\pi_n(x_{1:n})}{\pi_{n-1}(x_{1:n-1}) q_n(x_{\mathrm{n}}|x_{1:n-1})} \\ &= w_{n-1}(x_{1:n-1}) * \alpha_n(x_{1:n}) \\ &= w(x_1) \prod_{k=2}^{n} \alpha_k(x_{1:\mathrm{k}}) \end{aligned} \tag{12}$$

The decompositions above show that both the importance density and weights can be derived recursively.

2.1.3. Particle Filter

Applying Monte Carlo approximation and sequential importance sampling as well as selecting the appropriate importance distribution, a basic particle filter is defined as shown above. One only needs to recursively sample from the importance density and calculate the weight for the samples with appropriate initial values, then a density $\pi_n(x_{1:n})$ can be optimally estimated using the particle filter.

Referring back to the object tracking case, in order to estimate the posterior density $p(x_{0:k}|z_{1:k})$ in the joint space, let $\pi_n(x_{1:n})$ in Equation 6 be $p(x_{0:k}|z_{1:k})$, and $q_n(x_{1:n})$ in Equation 10 be $q(x_{0:k}|z_{1:k})$, then

$$w_k^i = \frac{p(x_{0:k}|z_{1:k})}{q(x_{0:k}|z_{1:k})} \tag{13}$$

where w_k^i is the weight of the *i* th particle at time *k*. Similar to Equation 11 [15],

$$q(x_{0:k}|z_{1:k}) = q(x_k|x_{0:k-1}, z_{1:k})q(x_{0:k-1}|z_{1:k}) \tag{14}$$

$$p(x_{0:k}|z_{1:k}) = \frac{p(z_k|x_{0:k}, z_{1:k-1})p(x_{0:k}|z_{1:k-1})}{p(z_k|z_{1:k-1})}$$

$$= \frac{p(z_k|x_{0:k}, z_{1:k-1})p(x_k|x_{0:k-1}, z_{1:k-1})p(x_{0:k-1}|z_{1:k-1})}{p(z_k|z_{1:k-1})}$$

$$= \frac{p(z_k|x_k)p(x_k|x_{k-1})p(x_{0:k-1}|z_{1:k-1})}{p(z_k|z_{1:k-1})}$$

$$\propto p(z_k|x_k)p(x_k|x_{k-1})p(x_{0:k-1}|z_{1:k-1}) \tag{15}$$

Substitute (14) and (13) into (12),

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

$$= w_{k-1}^i \frac{p(z_k|x_k)p(x_k|x_{k-1})}{q(x_k|x_{0:k-1}, z_{1:k})} \tag{16}$$

By recursively calculating the weights, the estimation of the posterior density $p(x_{0:k}|z_{1:k})$ can then be obtained using

$$p(x_{0:k}|z_{1:k}) = \sum_{i=1}^{N} w_n^i \delta_{X_{1:n}^i}(x_{1:n}) \tag{17}$$

If the importance density becomes only dependent on z_k and x_{k-1}, then $p(x_k|z_{1:k})$ is obtained similarly, except where

$$w_k^i \propto w_{k-1}^i \frac{p(z_k|x_k)p(x_k|x_{k-1})}{q(x_k|x_{k-1}, z_k)} \tag{18}$$

2.2. Particle Filter with PSO

Particle Filter successfully solves the problem of non-linearity of an optimization scenario by using particles to approximate a distribution, i.e., Monte Carlo approximation, and avoiding the ever-increasing dimension of the sampling space by sequential importance sampling. However, a generic particle filter suffers from several problems that are inherent from the particles.

The first issue resulting from the generic particle filter is degeneracy, where a large portion of the particles will have negligible weight after a few iterations. These particles contribute very little towards the distribution approximation because of their small weight. A popular technique to reduce such a situation is to resample where particles of small weights have a smaller chance to be forwarded to the next iteration. While alleviating the degeneracy problem, however, resampling on the other hand may lead to the impoverishment problem, where the diversity of particles are reduced especially for the case of small process noise.

Recently, there are research investigations trying to incorporate PSO with particle filter to overcome the limitations mentioned above. The authors in [16] merged PSO into particle filter by replacing sampling from importance density with PSO. By using PSO to generate the set of samples around high likelihood regions, the authors claim that the particle impoverishment is avoided. A similar algorithm was proposed in [17], in which PSO was used to move samples to a region in which both the likelihood of state and the prior density are significant. This algorithm however suffers from the same problem as in [16], even though these samples are originally drawn from the importance density. Since they are moved by PSO, their distribution does not fit the importance density anymore. Similar to the PSO based algorithm in [18], where the so-called

multi-layer importance sampling method is also used to move particles towards the region with high likelihood of observation.

3. OBJECT TRACKING BY MARGINAL PARTICLE FILTER WITH PSO

In this chapter, the problem of increasing dimension of the sampling space of the sampling mechanism in the generic particle filter is reviewed. Then, a novel marginal particle filter with PSO is proposed and discussed.

3.1. Marginal Particle Filter

As shown in Equation 11, the sequential importance sampling scheme draws samples, i.e., particles, from the importance density $q(x_{0:k}|z_{1:k})$. In practice, the sampling has to be done in a sequential manner, such that one does not have to perform sampling directly in a high dimension joint space. That means at each step the samples of the current state x_k, denoted as x_k^i where $i = 1,2, \ldots N$ and N is the number of particles, are drawn directly from $q\left(x_k^i \middle| x_{0:k-1}^i, z_{1:k}\right)$. Then, every such sample x_k^i together with its previous sample $x_{0:k-1}^i$ constitute a sample in the joint space of $q(x_{0:k}|z_{1:k})$. The trajectory of $x_0^i, x_1^i, \ldots\ x_k^i$ can be treated as a path from the first time step to the current time step k, as shown in Figure 3 as given in [19].

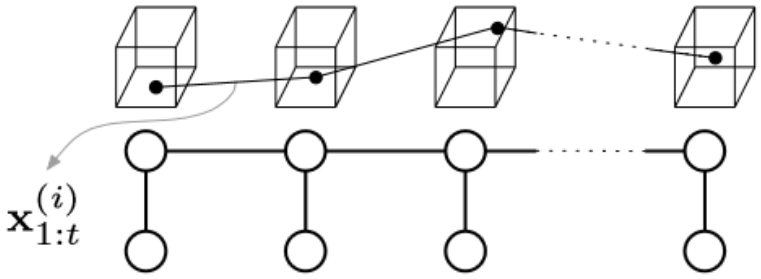

Figure 3. Sequential importance sampling path [19]

It is clearly shown 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 [19].

The high dimension 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 [19] developed a Marginal Particle Filter (MPF), which estimates the marginal posterior density $p(x_k|z_{1:k})$ only, thus, this essentially eliminates the need for sampling in a high dimensional joint space.

The marginal prediction step and update step is essentially the same as in Equation 5 and 6, rewritten as such:

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1} \tag{19}$$

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})} \tag{20}$$

Substitute Equation 19 into 20, the update step becomes:

$$p(x_k|z_{1:k}) \propto p(z_k|x_k) \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1} \tag{21}$$

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

$$p(x_k|z_{1:k-1}) = \sum_{i=1}^{N} w_{k-1}^i p(x_k|x_{k-1}^i) \tag{22}$$

Then posterior density can be estimated by

$$p(x_k|z_{1:k}) \propto p(z_k|x_k) \sum_{i=1}^{N} w_{k-1}^i p(x_k|x_{k-1}^i) \tag{23}$$

And the importance weights are updated by:

$$w_k = \frac{p(x_k|z_{1:k})}{q(x_k|z_{1:k})} \tag{24}$$

where $q(x_k|z_{1:k})$ is an arbitrarily chosen importance distribution to generate samples.

3.2. Marginal PF with Sampling by PSO

As discussed in Section 3.1, the marginal particle filter estimates the marginal posterior density $p(x_k|z_{1:k-1})$, thus its sampling space is always the state space at the current time x_k whose dimension is 1. The sampling space can be dimensionally reduced because samples in the current time step are disconnected from their predecessors by the marginalizing operation in Equation 19. This helps to alleviate the problem of degeneracy since the probability of samples is not vanishing along with the advance of the time step. However, the problem of degeneracy does not disappear completely. As shown in Figure 4, if the samples by sampling from the importance distribution are located in the region where the likelihood happens to be small, these particles will only negligibly contribute to the posterior density because of their small importance weights.

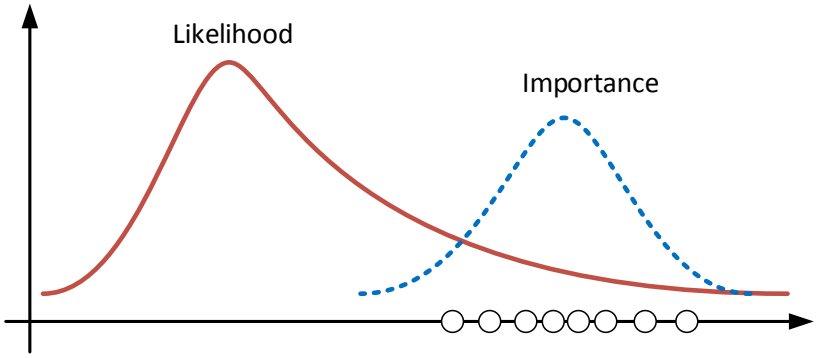

Figure 4. Illustration of an importance sampling example

3.2.1. Particles in the Particle Filter

Recalling that in the PSO procedure the particles are "travelling" all over the search space while being attracted by their personal best and global best as well as driven by their own inertia. These particles are usually scattered quite dispersedly within the search space during the initialization process in consideration of the large coverage. Eventually all particles will gather around the convergence point in case of convergence.

We use a standard PSO as given in [20] applied to 2D images as an example to demonstrate this convergence process. In this optimization problem, the fitness function evaluates the similarity between a predefined object and the candidates, which are represented by particles. The swarm population size is 30.

- **Initialization**. Particles are distributed across the possible region in the search space. Usually, the particles are spread out relatively broad to ensure a large search space coverage as shown in Figure 5.

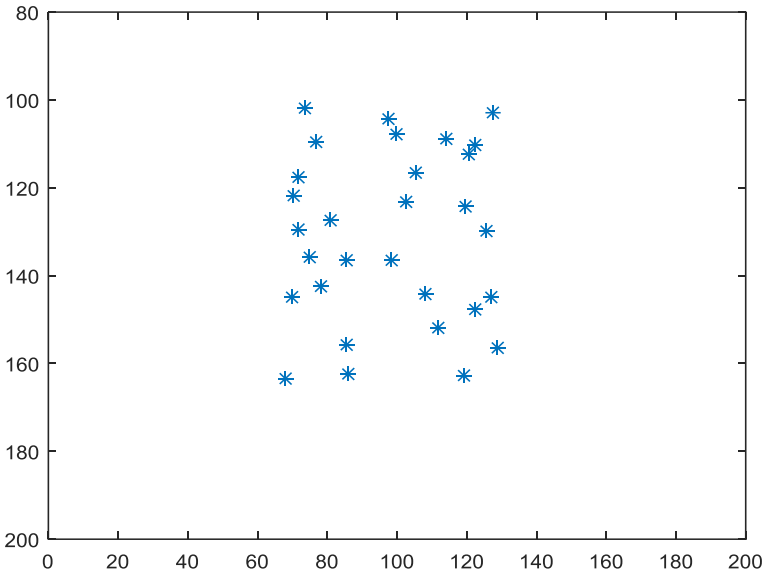

Figure 5. Particles at initialization step

- **During iterations**. Particles are attracted by both their own best position up to now and the global best position. While exploring the new position within the search space, particles start concentrating around the future convergence point if convergence was achieved at the end of the iteration. This is essentially the core philosophy of PSO. The movement of the particles during this stage of the process is show in Figure 6 and Figure 7.

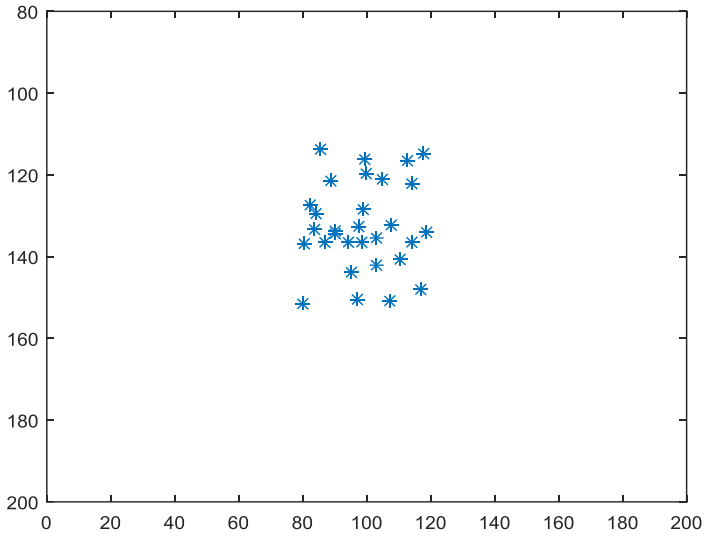

Figure 6. Particles during PSO iteration-1

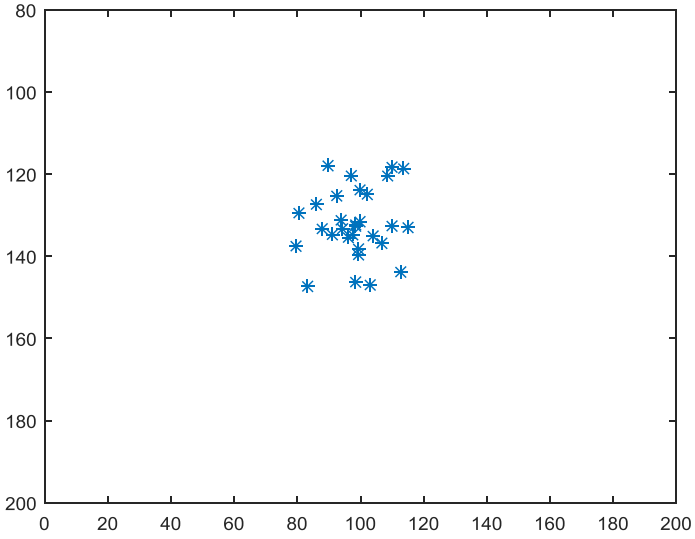

Figure 7. Particles during PSO iteration-2

- **Convergence**. Eventually, if successful, most, if not all, the particles will gather at the convergence point as shown in Figure 8.

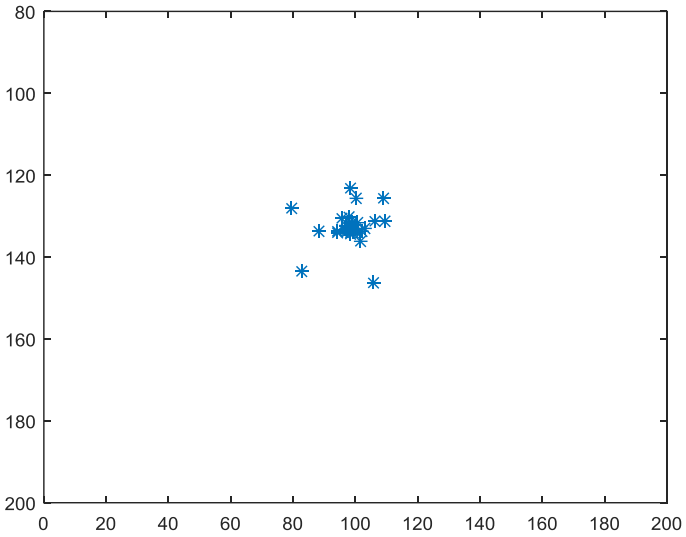

Figure 8. Illustration of convergence of particles

As illustrated by the figures above, after an appropriately configured PSO process, the particles from the randomly assigned position have moved to the high likelihood region. This is basically how PSO is used together with the particle filter in literature as discussed in Section 2.2.

However, the current application and utilization of PSO in the context of particle filter is far from adequate. In [16], particles are first drawn from the importance distribution and then driven by PSO towards a high likelihood region. Though, the authors claim by using PSO the problem of impoverishment is avoided, the usage of PSO outlined in their paper is inadequate. First, only the last generation of particles are used in the approximation of the posterior. Second, PSO upon convergence trends to gather all particles in a small zone around the convergence point. This somehow introduces certain level of concentration and reduces diversity between particles, which is the main origin of the impoverishment problem. Another drawback, which however is not quite apparent, is the inapplicability in the case of multiple separated high

likelihood regions. This drawback stems from the inherent property of the PSO for which a single swarm can only have one convergence position.

The algorithm in [17] fails to update the weights of the particles after they are moved by the PSO process. In that paper, the weights of samples are calculated according to the initial particle positions, yet particles by then have been moved. The meaning of the weight for samples is thus not the ratio of the importance density and posterior density at the point of the particles anymore.

The particle filter investigated in [18] is merely used as an initialization method by locating the position with maximal similarity. The complete tracking process can be hardly treated as Bayesian tracking.

3.2.2. Sampling by PSO

In this section, we introduce a novel sampling mechanism for use with the particle filter. 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 sampling of particles for the calculation of the approximation of posterior for time step *k* is done by the following steps:

- **Initialize**. Particles are distributed across the possible region in the search space as in basic 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 per Equation 1 and Equation 2 whereby denoting particles in each iteration as $x_{1:n}^{i}$, where *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 in the Section 4.1), then stop the iteration. Once convergence is achieved, the particles

stay closely around the convergence position with potential small variation, if any. If the maximum generation has been reached before the convergence criteria is met, then this particular execution of sampling has failed.

- **Combine**. Put particles from different iterations into the same set, denoting the set as $x_{1:n}^{1:m} = \{ x_k^i,\ k = 1,2, \dots, n, i = 1,2, \dots m\}$, where *m* is the number of iterations and *n* is the swarm size of the population. The combined particle set is distributed as shown in Figure 9.

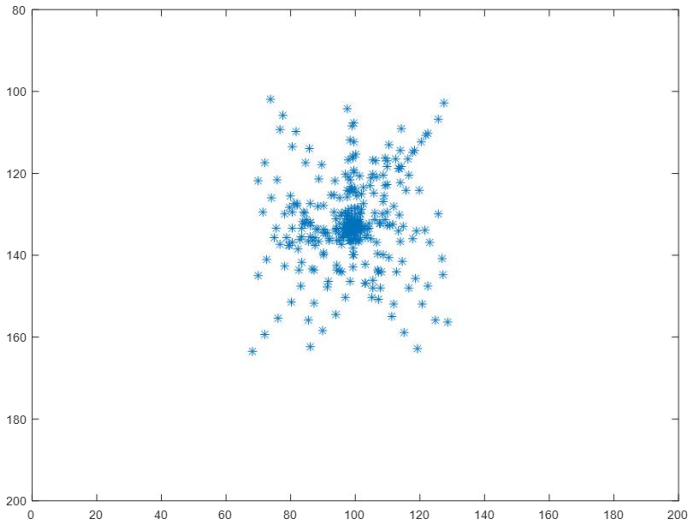

Figure 9. Distribution of all particles in PSO

- **Estimate Importance Distribution**. Empirically estimate the distribution of $x_{1:n}^{1:m}$, which is then used as the importance distribution. This can be done by either distribution fitting or non-parametric estimation.
- **Calculate Importance Probability**. For each particle in the set of $x_{1:n}^{1:m}$, calculate its importance probability. This probability is then used towards the estimation of the posterior as given in Equation 23.

Although the estimation of the importance distribution from the set of particles may introduce a certain level of variance into the system, since the particles are not drawn directly from this importance distribution. Our sampling mechanism using PSO still differentiates the other methods discussed in Section 3.2.1 in the following aspects:

- Instead of using only the last generation of particles, our method considers all generations, which increases the number of particles dramatically by several folds without extra computational expense, and in turn provides more robust approximation of the posterior density.
- By putting all generations of particles together, the final particle set focuses both on the concentration and diversity thanks to the latter generations of particles around the convergence position, and the scattering of earlier generations of particles in the search space, respectively.
- The balance between diversity and concentration can be easily adjusted via the parameters of PSO. By adjusting the termination criteria, the particles piling up in a high likelihood region can be controlled. By tuning the initializing parameters, the coverage of the sampling can be changed correspondingly.
- Compared to the particle filter used in [17], the empirically estimated importance distribution provides reasonable importance density for the particles.

3.2.3. Marginal Particle Filter with Sampling by PSO

To summarize our proposed marginal particle filter with sampling by PSO, let us revisit the marginal particle filter discussed in Section 3.1. The marginal particle filter approximates only the marginal posterior distribution, thus, the dimension of the sampling space is limited and the problem of degeneracy of the weights is alleviated. However, the sampling distribution (the

part that influences the performance significantly) is up to the researcher's arbitrary choice. This is where our PSO based sample mechanism comes in. The complete description of our proposed algorithm is presented as follows:

Algorithm 1: Marginal Particle Filter with Sampling by PSO

Input:

$\{x_{k-1}^i, w_{k-1}^i\}_{i=1}^N$, which are *N* particles and their weights at time *k-1*.

z_k, which is the measurement at time *k*.

Steps:

- **Perform PSO**. Initialize the first generation of particles for PSO, denoting them as $x_k^{1:m} = \{ x_k^i, i = 1,2, \dots m\}$, where *m* is the swarm size of the population. Perform the PSO iterations according to the PSO procedure until convergence or the maximum number of generations is reached.

1) If PSO terminates due to convergence, then aggregate the particles from all iterations to obtain a particle set: $\{x_k^i, pl_k^i\}_{i=1}^{m*n} = \{ x_k^i, i = 1,2, \dots m * n\}$, where *n* is number of iterations, *m* is the swarm population size and pl_k^i is the likelihood ($p(z_k|x_k)$) of particle x_k^i.

2) If PSO terminates due to reaching the maximum number of generations, then use the global best x_{gbest} and $\{x_k^i, w_k^i\}_{i=1}^N = \{x_{gbest}, 1\}$ as the final output. Where x_{gbest} is the estimation of the object position for this frame, and $\{x_k^i, w_k^i\}_{i=1}^N$ is used as the input for the next frame. Then, proceed to the next frame.

- **Prediction.** For each x_k^i in $\{x_k^i, pl_k^i\}_{i=1}^{m*n}$, estimates its prior as:

$$p(x_k^i | z_{1:k-1}) = \sum_{j=1}^N w_{k-1}^j p(x_k^i | x_{k-1}^j) \quad (25)$$

to obtain pp_k^i , which is the prior probability ($p(x_k|z_{1:k-1})$) for particle x_k^i.

- **Estimate Importance Probability:** First, estimate an empirical importance distribution for the particle set of PSO. Then, the importance probability for each particle q_k^i is calculated by the estimated importance distribution as described in Section 3.2.2.
- **Calculate weight**. The weight for each particle w_k^i is computed according to

$$w_k^i = \frac{p\left(z_k \middle| x_k^i\right) \sum_{j=1}^{N} w_{k-1}^j p(x_k^i | x_{k-1}^j)}{q\left(x_k^i \middle| z_{1:k}\right)} \quad (26)$$

Specifically $w_k^i = \frac{pl_k^i pp_k^i}{q_k^i}$. After normalization of the weights, the particles and their weights, denoted as $\{x_k^i, w_k^i\}_{i=1}^{m*n}$, are obtained.

- **Object Position Estimation:** As discussed in [14][15], the best estimation of the object position is the expected value of the posterior which can be obtained by:

$$\hat{p}(x_k | z_{1:k}) = \sum_{i=1}^{N} w_k^i x_k^i. \quad (27)$$

4. EXPERIMENTS AND RESULTS

In this chapter, we perform a series of experiments to verify our hypothesis and validate our algorithm. First, we will demonstrate that our tracking algorithm sampling by PSO is feasible and robust with regards to the parameter setting. Then, we compare the proposed marginal particle filter with PSO (MPFPSO) against the conventional particle filer (PF).

4.1. Experiments with Sampling by PSO

Before starting the experiment of sampling by PSO as described in Section 3.2.2, our experimental environment is set up as follows:

- The dataset for demonstration of sampling by PSO is gathered from an actual MPFPSO tracking process.
- The population size of the particle swarm is 50.
- The convergence criterion we use here is: the global bests in 5 consecutive frames are similar at least 98%, where similarity is measured as:

  overlap (5^{th} frame, 1-4 frames combined) / combine (5^{th} frame, 1-4 frames combined), where:

  overlap (5^{th} frame, 1-4 frames combined) means the overlapping area between the 5^{th} frames and the combined area of 1-4 frames, and

  combine (5^{th} frame, 1-4 frames combined) means the combined area of the last frames and the previous 4 frames. Basically, this criterion stops the PSO process when the global best of the swarm only improves very little.
- Particles in the swarm are initialized uniformly within ±32 pixel on both *horizontal* and *vertical* axis around the previous target position.

- The code is run based on the framework developed in [21], and the test video is the one named “CarDark”.
- We use the appearance model and similarity measurement defined in [22] because of its high resolution and peaked likelihood landscape.

We first make the assumption about the distribution to which the combined particles would conform, then, we estimate its parameters using the method of maximum likelihood as in Equation 29.

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

$$p(y) = \frac{\sqrt{2}}{\sigma\sqrt{\pi}} \exp\left(-\frac{y^2}{2\sigma^2}\right) \quad y > 0 \tag{28}$$

The reason we choose the Half-Normal Distribution is because it only depends on one parameter, which introduces less deviation during the method of 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 only parameter σ can be estimated by:

$$\hat{\sigma} = \sqrt{\frac{1}{n}\sum_1^n d_i^2} \tag{29}$$

where d_i is the distance to the center for particle *i*.

Let us take the particle dataset output by sampling by PSO on one frame as an example. The PSO converges at iteration 9, so there are in total 450 particles in the dataset. Figure 10 shows how these particles are scattered on the image plane. Figure 11 shows the histogram of distance of particles to the center fitting the Half-Normal Distribution very well. We can as well

treat the histogram presented in Figure 11 as a probability density function by normalization. These two probability density functions are very similar to each other as illustrated in Figure 12. Actually, the Bhattacharrya distance [1] (also a.k.a Hellinger distance) between two density functions is only 0.0746.

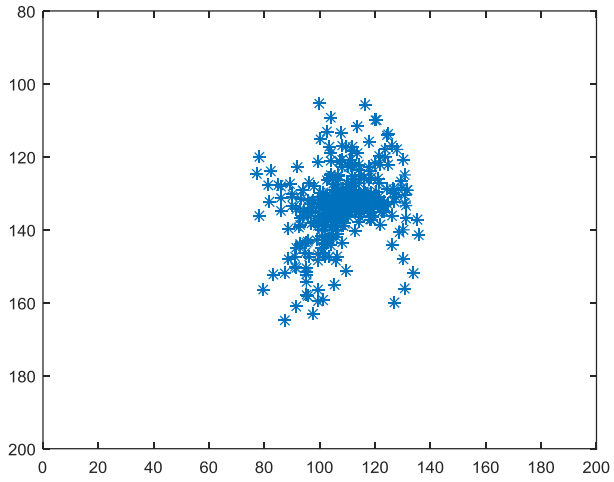

Figure 10. Particles distribution on the image plane

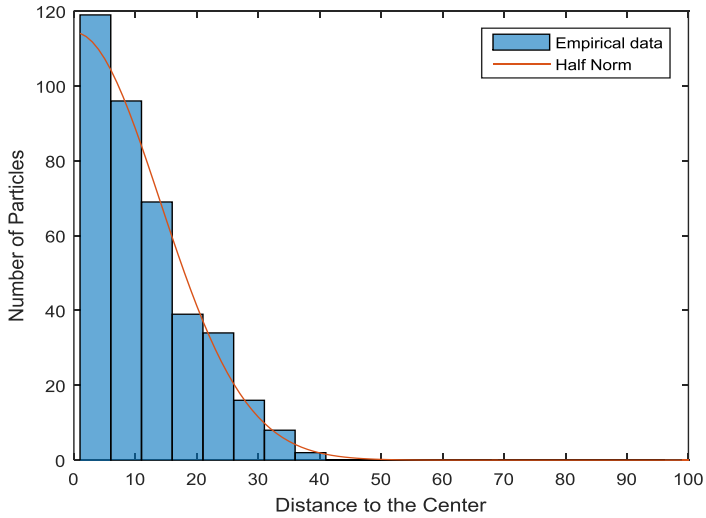

Figure 11. Comparison of empirical data and Half-Normal Distribution (histogram)

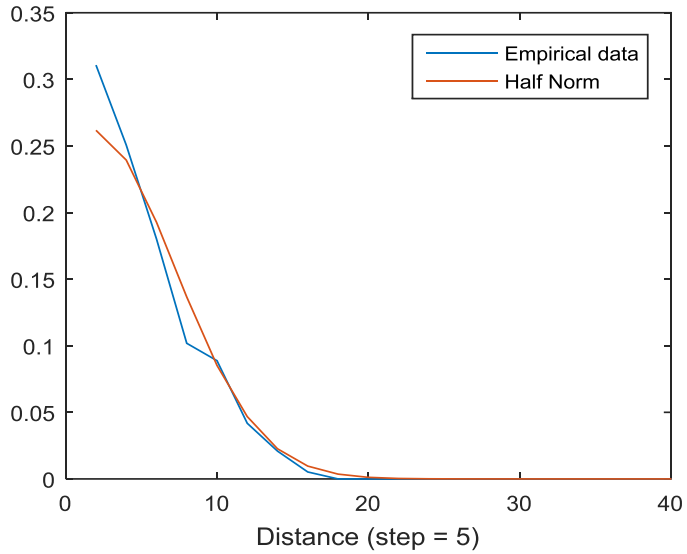

Figure 12. Comparison of empirical data and Half-Normal Distribution (density)

As PSO is a stochastic algorithm, and the distribution of particles is the result of such a stochastic process. Therefore, the particles sampled by the PSO process are non-deterministic, i.e., data from one frame is not enough to eliminate uncertainty. In order to address such concern, we collected the Bhattacharrya distances between the empirical distribution and the fitted Half-Normal Distribution for each frame in a video sequence. After deleting data for frames on which the PSO process failed to converge, we obtained the Bhattacharrya distances for a total of 383 frames in the video sequence "CarDark" as shown in Figure 13. From this figure, it can be observed that on most frames the Bhattacharrya distance is less than 0.15.

Figure 14 shows the comparison of aggregated empirical distribution and Half Normal Distribution, where the aggregated empirical distribution is obtained by averaging the standardized empirical distribution of each frame. The figure shows that the comparison of the aggregated empirical distribution and Half Normal Distribution fits closely except for a small region around the distance of 8.

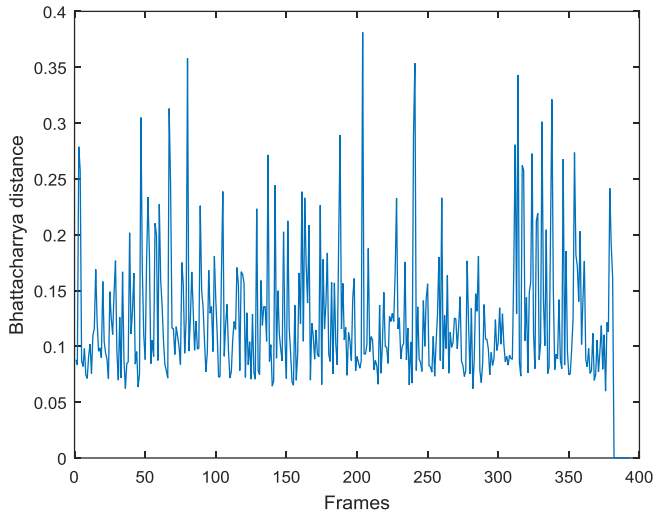

Figure 13. Bhattacharrya distance for each frame

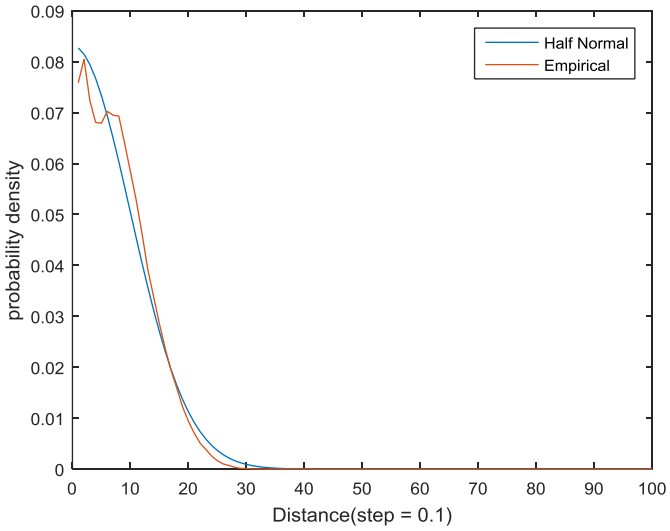

Figure 14. Aggregated empirical distribution vs Half Normal (Standardized)

It is worth to note that the factors that impact the empirical distribution of particles include: PSO convergence criteria, the particle initialization, and PSO control parameter such as particle maximal velocity, particle swarm size, etc.

4.2. Experiments and Results

In this section, a tracking performance comparison is made between the marginal particle filter with PSO (MPFPSO) and the bootstrap particle filter (BPF) [23] as well as the pure marginal particle filter (MPF).

We keep using the experimental environment set up as in the previous section, with the following extra adjustments and constraints:

- In total, 51 video sequences collected in [21] are tested.
- To simplify the problem, only the positions of the center of the target are tracked. That means the bounding box of the target does not change its shape and scale throughout the whole tracking process.
- The motion model is assumed to be $p(x_k|x_{k-1}) \sim N(x_{k-1}, R)$, where R is a covariance matrix.
- In MPFPSO, the swarm size is 50 and maximal generation is 20.
- In BPF and MPF, the size of the particles is set to 500.

1. Comparison of precision rate and success rate of the trackers

We use precision and success rate for quantitative analysis as in [21], between which the precision rate shows the percentage of frames whose estimated location is within the given threshold distance of the ground truth, and the success rate shows the percentage of successful frames whereby successful means that the tracking result bounding box overlaps with the ground truth by at least a certain amount.

The precision rate is given in Figure 15, and the success rate is given in Figure 16. We can observe that MPFPSO is performing best in terms of both precision rate and success rate.

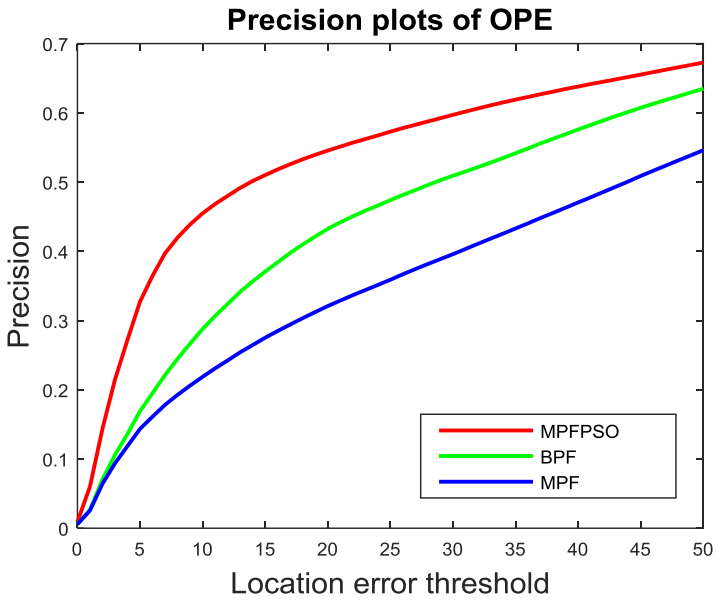

Figure 15. Comparison of performance: Precision rate

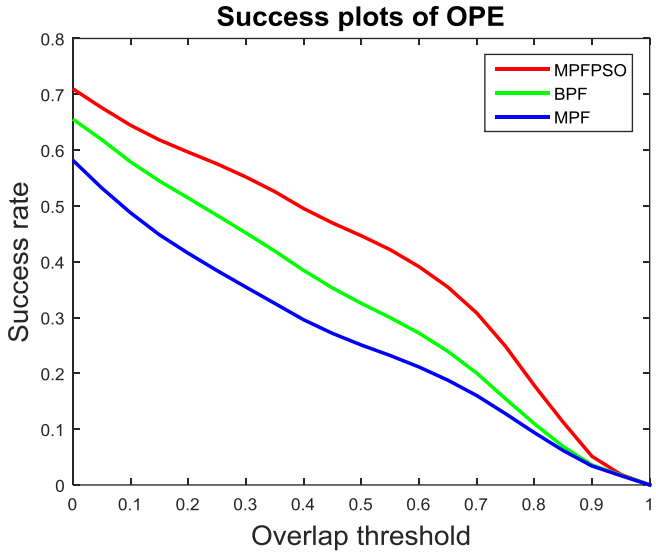

Figure 16. Comparison of performance: Success rate

2. Comparison of the running time for trackers

The running time to perform the tracking measures how fast a tracker can process the video sequences. Although the running time depends on many factors, such as appearance model, coding style, platform, etc., however, in our experiment, we fix as many factors as

possible to focus on the difference among the three trackers. Since we use the same platform and appearance model in the three trackers, the discrepancy of the running time basically boils down to how the trackers approximate the posterior density function of the particles and how many particles are used.

Table 1 shows the comparison of processed frame per second (fps) and particle number for three trackers. Please note that for MPFPSO, the particle number per frame is obtained by the average.

Table 1. Comparison of the fps and the particle number.

	fps	Particle number per frame
MPFPSO	6.7019	457
BPF	9.0882	500
MPF	5.6616	500

4.3. Discussion

In Section 4.1, the sampling by PSO is demonstrated and discussed. In this section, we focus on the experiments presented in Section 4.2.

1. Precision rate and success rate

As shown in Figure 15 and Figure 16, MPFPSO performs best compared to the other two trackers in both performance measure: precision rate and success rate. This performance margin comes mostly from the sampling mechanism. MPF itself is actually inferior to BPF as shown in both precision rate and success rate figures. The reason for this is because for BPF we used the optimal importance distribution - exactly the same as the motion model. The adaptation of optimal importance distribution [24] minimizes the variance of particles' weight [15]. But still the sampling by PSO compensates as explained in the following paragraphs.

By randomly choosing a frame as an example, the likelihood landscape around the target region is illustrated in Figure 17. The peak is where the target is located.

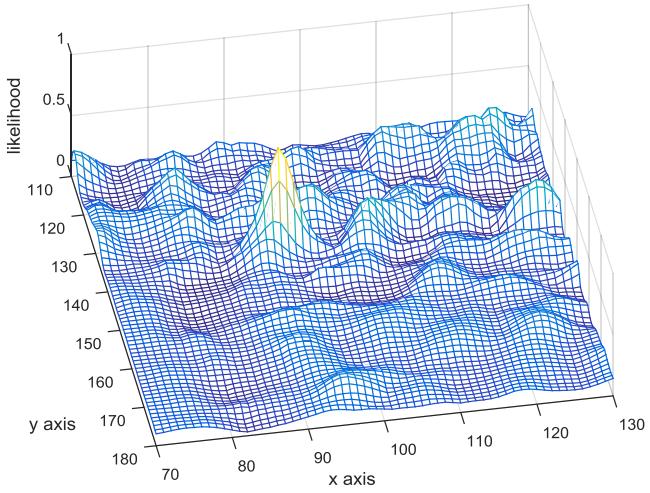

Figure 17. Likelihood landscape

Figure 18 shows how the particles distribute around the target region in different ways for this chosen frame comparing the three approaches. It can be observed that particles in MPFPSO are scattered more dispersedly than in BPF and MPF. Yet, the swarm of particles in MPFPSO can always focus on the high likelihood region. In contrast, the particles in BPF and MPF flock together with very few diversity and in turn cover a smaller area than in MPFPSO. The reason for such discrepancy lies in how these particles are generated. In BPF and MPF, the particles are drawn from the motion model or the importance distribution. The way the particles stretch out is defined only by the motion model or the importance distribution without regard to the likelihood landscape. In comparison, the particles in MPFPSO are first spread out throughout a relatively large search space and then concentrate around the high likelihood region.

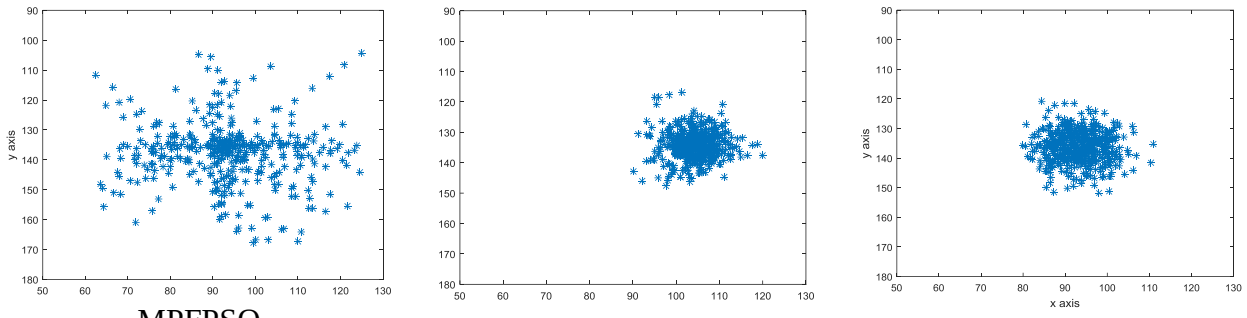

Figure 18. Comparison of particle distribution

2. Running time of the trackers

Our experiments show that the calculation of Equation 25 is very computational intensive. In particular, for MPFPSO the calculation of Equation 25 results in 43.7 % of the overall computational time. However, this part does not change even when the appearance model is changed. Considering MPFPSO generates less particles on average than BPF and MPF does, for the complex appearance model, MPFPSO is more computational efficient than BPF and MPF. This observation can also be supported by the fact that MPF is actually slower than MPFPSO, because MPF generates more particles per frame but also carries the same workload of having to do the calculation of Equation 25.

The reason that MPFPSO generates less particles on average though maintain high performance is because the PSO process stops once convergence is achieved. That saves unnecessary computation and it also means that the population size of the particles is adaptive to the image of the current frame as shown in Figure 19. For a distinct target, the PSO process terminates earlier, otherwise the PSO just iterates over more generation in order to search for the high likelihood region. On the other side, the number of particles in BPF and MPF are constant.

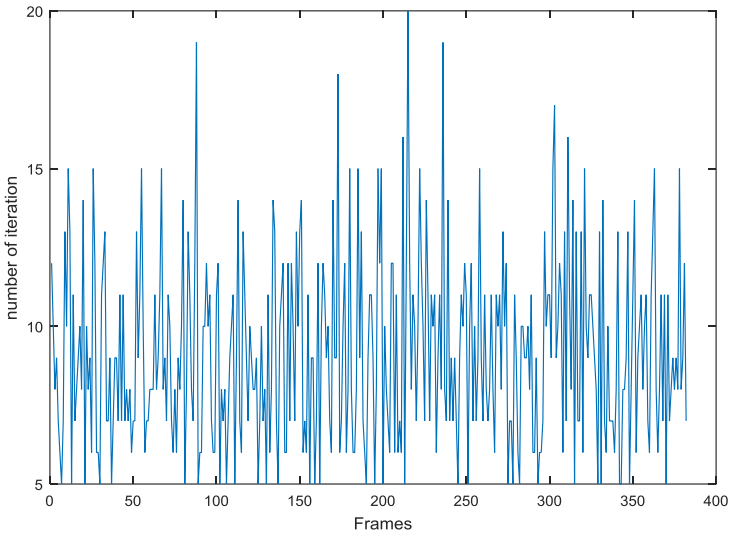

Figure 19. Number of PSO iterations

5. CONCLUSION AND FUTURE WORK

This thesis first reviewed the current state of visual object-tracking and evolutionary optimization, then discussed past research on the applications of particle filter and PSO applied to object-tracking and listed their drawbacks. In the Chapter 3, a novel particle filter has been proposed to overcome the problems including unused particles and lack of diversity as identified previously, where a sampling by PSO mechanism is used together with the marginal particle filter. Simulations and results are then presented and discussed. It is demonstrated that the proposed method does improve the tracking performance without introducing extra computation 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 both bootstrap particle filter and pure marginal particle filter. It is also illustrated that the sampling by PSO method is an efficient yet effective sampling mechanism, which adapts its computational cost to image quality and always focuses on the high likelihood region.

In the future, our marginal particle filter with sampling by PSO could be further investigated in the following ways:

- Accelerate convergence of PSO, thus, reduce the number of particles to save computation cost.
- Optimize PSO parameters to achieve better importance distribution fitting.
- The motion model could be further enhanced to reflect the real target motion.
- Relation between swarm size and tracking performance, computational cost is yet to be examined.

REFERENCES

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

[2] Smith, K. C. (2007). *Bayesian methods for visual multi-object tracking with applications to human activity recognition* (Doctoral dissertation, École Polytechnique). Retreived from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.348.3000.

[3] Skog, I., & Händel, P. (2009). In-car positioning and navigation technologies—A survey. *Intelligent Transportation Systems, IEEE Transactions on*, *10*(1), 4-21.

[4] Yilmaz, A., Javed, O., & Shah, M. (2006). Object tracking: A survey. *Acm computing surveys (CSUR)*, *38*(4), 13

[5] McKenna, S. J., Raja, Y., & Gong, S. (1999). Tracking colour objects using adaptive mixture models. *Image and vision computing*, *17*(3), 225-231.

[6] Bradski, G. R. (1998, October). Real time face and object tracking as a component of a perceptual user interface. In *Applications of Computer Vision, 1998. WACV'98. Proceedings, Fourth IEEE Workshop on* (pp. 214-219). IEEE.

[7] Black, M. J., & Jepson, A. D. (1998). Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, *26*(1), 63-84.

[8] Avidan, S. (2004). Support vector tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *26*(8), 1064-1072.

[9] Grabner, H., & Bischof, H. (2006, June). On-line boosting and vision. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on* (Vol. 1, pp. 260-267). IEEE.

[10] Sarker, R., Mohammadian, M., & Yao, X. (2002). *Evolutionary optimization* (Vol. 48). Springer Science & Business Media.

[11] Eberhart, R. C., & Kennedy, J. (1995, October). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science* (Vol. 1, pp. 39-43).

[12] Handschin, J. E., & Mayne, D. Q. (1969). Monte carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering†. *International journal of control*, *9*(5), 547-559.

[13] Isard, M., & Blake, A. (1998). Condensation—conditional density propagation for visual tracking. *International journal of computer vision*, *29*(1), 5-28.

[14] Doucet, A., & Johansen, A. M. (2009). A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, *12*, 656-704.

[15] Arulampalam, M. S., Maskell, S., Gordon, N., & Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *Signal Processing, IEEE Transactions on*, *50*(2), 174-188.

[16] Tong, G., Fang, Z., & Xu, X. (2006, July). A particle swarm optimized particle filter for nonlinear system state estimation. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on* (pp. 438-442). IEEE.

[17] Klamargias, A. D., Parsopoulos, K. E., & Vrahatis, M. N. (2008, July). Particle filtering with particle swarm optimization in systems with multiplicative noise. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation* (pp. 57-62). ACM.

[18] Zhang, X., Hu, W., Maybank, S., Li, X., & Zhu, M. (2008, June). Sequential particle swarm optimization for visual tracking. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (pp. 1-8). IEEE.

[19] Klaas, M., De Freitas, N., & Doucet, A. (2012). Toward practical N2 Monte Carlo: the marginal particle filter. *arXiv preprint arXiv:1207.1396*.

[20] Bratton, D., & Kennedy, J. (2007, April). Defining a standard for particle swarm optimization. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE* (pp. 120-127). IEEE.

[21] Wu, Y., Lim, J., & Yang, M. H. (2013, June). Online object tracking: A benchmark. In *Computer vision and pattern recognition (CVPR), 2013 IEEE Conference on* (pp. 2411-2418). IEEE.

[22] Jia, X., Lu, H., & Yang, M. H. (2012, June). Visual tracking via adaptive structural local sparse appearance model. In *Computer vision and pattern recognition (CVPR), 2012 IEEE Conference on* (pp. 1822-1829). IEEE.

[23] Gordon, N. J., Salmond, D. J., & Smith, A. F. (1993, April). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)* (Vol. 140, No. 2, pp. 107-113). IET Digital Library.

[24] Doucet, A., Godsill, S., & Andrieu, C. (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and computing*, *10*(3), 197-208.

[25] Deza, M. M., & Deza, E. (2006). *Dictionary of distances*. Elsevier.