INTERNATIONAL JOURNAL OF ADAPTIVE CONTROL AND SIGNAL PROCESSING
Int. J. Adapt. Control Signal Process. (2015)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/acs.2656

Numerical fitting-based likelihood calculation to speed up the particle filter

Tiancheng Li[1,2,*,†], Shudong Sun[2], Juan M. Corchado[1,4], Tariq P. Sattar[3] and Shubin Si[2]

[1]*BISITE research group, Faculty of Sciences, University of Salamanca, 37008 Salamanca, Spain*
[2]*School of Mechanical Engineering, Northwestern Polytechnical University, 710072 Xi'an, China*
[3]*Department of Engineering and Design, London South Bank University, SE1 0AA London, UK*
[4]*Osaka Institute of Technology, Asahi-ku Ohmiya, Osaka 535-8585, Japan*

SUMMARY

The likelihood calculation of a vast number of particles forms the computational bottleneck for the particle filter in applications where the observation model is complicated, especially when map or image processing is involved. In this paper, a numerical fitting approach is proposed to speed up the particle filter in which the likelihood of particles is analytically inferred/fitted, explicitly or implicitly, based on that of a small number of so-called fulcrums. It is demonstrated to be of fairly good estimation accuracy when an appropriate fitting function and properly distributed fulcrums are used. The construction of the fitting function and fulcrums are addressed respectively in detail. To avoid intractable multivariate fitting in multi-dimensional models, a nonparametric kernel density estimator such as the nearest neighbor smoother or the uniform kernel average smoother can be employed for implicit likelihood fitting. Simulations based on a benchmark one-dimensional model and multi-dimensional mobile robot localization are provided. Copyright © 2015 John Wiley & Sons, Ltd.

Received 29 November 2014; Revised 23 September 2015; Accepted 17 November 2015

KEY WORDS: nonlinear filtering; particle filter; robot localization; kernel density estimator

1. INTRODUCTION

This paper concerns the design of a high-speed particle filter (PF) for a variety of nonlinear state estimation problems such as localization, positioning, and tracking. Computing efficiency is a critical requirement in the industry but is particularly challenging for the application of the PF. In brief, the nonlinear filtering problem is formulated in the form of the discrete dynamic state-space model (SSM).

$$\begin{cases} x_t = f_t\left(x_{t-1}, u_t\right) & \text{(state transition equation)} \\ y_t = h_t\left(x_t, v_t\right) & \text{(observation equation)} \end{cases} \tag{1}$$

where t indicates discrete time, $x_t \in \mathbb{R}^{d_x}$ denotes the state, $y_t \in \mathbb{R}^{d_y}$ denotes the observation, u_t and v_t denote stochastic noise affecting the state transition function $f_t : \mathbb{R}^{d_x} \times \mathbb{R}^{d_u} \to \mathbb{R}^{d_x}$, and the observation function $h_t : \mathbb{R}^{d_x} \times \mathbb{R}^{d_v} \to \mathbb{R}^{d_y}$, respectively. Furthermore, let $x_{0:t} \triangleq (x_0 x_1, \ldots, x_t)$ and $y_{0:t} \triangleq \left(y_0, y_1, \ldots, y_t\right)$ be the history path of the signal and of the observation process, respectively.

*Correspondence to: Tiancheng Li, BISITE research group, Faculty of Sciences, University of Salamanca, 37008 Salamanca, Spain.
†E-mail: t.c.li@mail.nwpu.edu.cn

To estimate the state based on the noisy observations over time, a standard solution to the SSM-based filtering problem is the *Recursive Bayesian estimation*, which is based on two assumptions as follows:

(A.1) The states follow a first-order Markov process

$$p(x_t \,|x_{0:t-1}) = p(x_t \,|x_{t-1}) \tag{2}$$

(A.2) The observations are independent of the given states

$$p(y_{0:t-1} \,|x_t) = \frac{p(x_t \,|y_{0:t-1})\, p(y_{0:t-1})}{p(x_t)} \tag{3}$$

The recursive Bayes estimation comprises two steps that form one iteration:

(Step.1) Prediction (Chapman–Kolmogorov equation)

$$p(x_t \,|y_{0:t-1}) = \int_{R^{d_x}} p(x_{t-1} \,|y_{0:t-1})\, p(x_t \,|x_{t-1})\, dx_{t-1} \tag{4}$$

(Step.2) Updating or correction (Bayes' rule)

$$p(x_t \,|y_{0:t}) = \frac{p(x_t \,|y_{0:t-1})\, p(y_t \,|x_t)}{\int_{R^{d_x}} p(x_t \,|y_{0:t-1})\, p(y_t \,|x_t)\, dx_t} \tag{5}$$

where $p(y_t \,|x_t)$ is the likelihood, which is calculated based on the observation function

However, these two steps are only conceptual, and the involved integration is generally incomputable except in the linear Gaussian system and a few finite state-space hidden Markov models. To solve the integration, we have to resort to suboptimal simulation-based methods, such as the point-mass (PM) filter [1, 2], the unscented filter [3], and the Monte Carlo methods. The Monte Carlo method has become one of the most prevailing tools for sophisticated models, typically including particle flow [4], Markov chain Monte Carlo [5], and sequential Monte Carlo (often called PF) [6, 7], etc.

Specifically, the PF approximates the posterior density that can be of any distribution by a set of particles with associated weights, and its computation complexity is proportional to the number of particles used. Therefore, it suffers from heavy computation when a vast number of particles are necessarily used for complicated systems. To speed up the PF, we propose a numerical fitting approach to calculate the likelihood of particles that does not need to reduce the number of particles. Numerical fitting has been proven a powerful and a universal method for data prediction and has been used in a range of statistical applications where adequate analytical solutions may not exist. In the context of Kalman filtering, the unscented transform technique is a proven type of statistical linear regression [3]. To our knowledge, this is the first attempt to employ the numerical fitting tool for likelihood calculation for PF.

The remainder of this paper is organized as follows. A brief review of the state of the art development of fast processing PF is presented in section 2. The conceptual framework and implementation details of the proposed approach are given in section 3. Simulations on typical one/multi-dimensional models and further discussion are given in section 4 and 5, respectively. The conclusion is offered in section 6.

2. RELATED WORK: FAST PARTICLE FILTERING

The primary challenge for the application of the PF is from its computational inefficiency. When the rate of incoming sensor data is higher than the updating rate of the filter, then the sensor data cannot be used fully but has to be abandoned some. To avoid this, considerable efforts have been devoted to improve the computing speed of the PF to meet the real-time computing requirement. One of the most effective solutions is to reduce the number of particles used, e.g. [8–10]. However, caution has to be exercised on reducing the number of particles as a smaller number of particles make it hard to approximate the underlying probability distribution function (PDF) properly and to cope with the information imprecision. Several advanced forms of the particle filter have been proposed to work with fewer samples, but it is seldom possible to obtain a win-win situation in general. For example, the importance density has been modified to pull the particles close to the observations in order to reduce the number of particles required in [11]. The box particle [12] occupies a small and controllable rectangular region in the state space, which reduces the number of particles required in high-dimensional problems and is suitable for parallel processing.

In many real-life applications, the weight updating that is sensor-sensitive is much more computationally intensive than the state prediction, especially when map or image processing is involved in the observation model. This is particularly true for robot localization/tracking [13, 14], high-dimensional tracking [11], and visual tracking and prediction [15]. In these cases, the filter accuracy and computing speed are both restricted heavily by the updating step. Because of this, efforts have been made to increase the filtering speed by simplifying the updating step, especially for complex observation models that involves images/maps and creates the main computational burden, in two ways. One way is to reduce the number of updating cycles and the other is to reduce the computation of each updating cycle. In the first way, only observations that fall inside a specific scope around the particle have significant impact to the weight of the particle, while those outside of the scope have negligible impact and are therefore not taken into account [16]. In the second way, the weight of each particle is determined by employing the incremental likelihood calculation in [17].

Real-time techniques such as parallel processing and dimension decomposition provide possibilities for fast processing of the PF. Given the fast development and popularity of computers, multicore platforms and general purpose graphics processing units, parallel computing becomes prevailing for the PF, for example, [18, 19]. One of the biggest challenges for developing parallel PF is the resampling operation that requires the joint processing of all particles and therefore prevents parallelization of PFs. Therefore, various solutions for parallel/fast resampling and parallel PF have been proposed, [20, 21]. For the particle implementation of the probability hypothesis density filter for multi-target tracking, normalization is not needed in resampling which provides much convenience for parallel implementation [22].

The dimension decomposition idea of *Rao–Blackwellization* (RB) [23] is to divide the state so that the Kalman filter is used for the part of the state that is linear, and PF is used for the other part that is nonlinear, which inspires many similar developments. For example, the proposed method estimates orientation by using a particle filter, while the position and velocity are estimated by using Kalman filter [13]. Furthermore, to remove the linearity limitation imposed by the Kalman filter, the *Decentralized PF* ([24] splits the filtering problem into two nested sub-problems and handles each individually using PFs. This differs from RB as two parts are all approximated by conditional PFs.

Furthermore, one may partition the state space into more subspaces to combat the curse of dimensionality and run separate PFs in each subspace [25]. A similar idea is implemented in [6], which represents each component as a single chain Bayesian network and uses PF to track each component for multi-component tracking. Splitting the state space is also an appealing and even necessary way to deal with high dimensionality [26]. Meanwhile, the time scale separation exhibited in [27] allows two simplifications of the PF: (i) to use the averaging principle for the dimensional reduction of the dynamics for each particle during the prediction step and (ii) to factorize the transition probability for the RB of the update step. The resulting PF has improved computing speed and reduced variance. It is fair to say that high dimensionality remains challenging for the application of PFs [15, 28]. More effective and efficient speed-up solutions for the general SSM, whether low-dimensional or high-dimensional, are still required, especially when it has a complicated observation model.

3. NUMERICAL FITTING FOR LIKELIHOOD CALCULATION

3.1. The conceptual framework

The PF approximates the posterior PDF by a set of particles $x_t^{(i)}$ with associated non-negative weight $w_t^{(i)}$ that internally employs the strong law of large numbers, that is,

$$p(x_t \,|y_{0:t}) \propto \sum_{i=1}^{N} w_t^{(i)} \delta \left(x_t - x_t^{(i)} \right) \tag{6}$$

where $\delta(\cdot)$ is the Dirac delta impulse and N is the number of particles.

The essence of the PF is to evaluate how well each particle conforms to the dynamic model and explains the measurements, using this assessment to generate a weighted particulate approximation to the posterior distribution, and hence form state estimates. More precisely, given a proposal importance density $q(\cdot)$ (that shall incorporate the state process model and the newest observations [6, 7]), the weight is determined as

$$w_t^{(i)} \propto w_{t-1}^{(i)} \frac{p\left(y_t \,|x_t^{(i)}\right) p\left(x_t^{(i)} \,|x_{t-1}^{(i)}\right)}{q\left(x_t^{(i)} \,\middle|\, x_{0:t-1}^{(i)}, y_{0:t}\right)} \tag{7}$$

where $p\left(y_t \,|x_t^{(i)}\right)$ is the likelihood of the particle $x_t^{(i)}$ given observation y_t, which is critical to the PF and is the core of our approach.

Moreover, resampling is often applied to reset the weights [20] to be equal or approximately to combat weight degeneracy, that is, sampling importance resampling (SIR) or sequential importance sampling and resampling, which shall not significantly change the posterior distribution [21].

In this paper, we concern the SSM in which the likelihood function is computationally intensive, or even intractable [29] and has to be approximated. For this reason, we propose a numerical fitting approach to speed up the likelihood calculation of particles, as referred to as likelihood fitting (Li-fitting). Basically, the task of numerical fitting is to recover $y = f(x; C)$, where C are the parameters that are to be determined by using given data based on the belief that the data contains a slowly varying component, which captures the trend about, y, and a varying component of comparatively small amplitude that is the error/noise in the data. We here implicitly assume that the likelihood function is continuous and has a smooth distribution, which is the case in most applications. Our approach is based on the situation that the direct likelihood calculation is computationally more intensive than the numerical fitting employed.

As the first step, a number of fulcrums are created as detailed in Section 3.2. Then, the *Likelihood function* (Li-function) $L_t(\cdot)$ is constructed based on the likelihoods of fulcrums that are directly calculated by utilizing the observation at time t, which will be used to infer the likelihood of the particles, i.e.

$$p\left(y_t \,\middle|\, x_t^{(i)}\right) = L_t\left(x_t^{(i)}\right) \tag{8}$$

Here, the Li-function can be thought of as a sufficient statistic for calculation of the likelihood of each particle. In the implicit Li-fitting form, the likelihoods of particles are fitted directly from that of the nearby fulcrums via smoothing and the Li-function will not be obtained explicitly; see section 3.4 for details.

For illustration purposes, the idea of the Li-fitting approach can be described in Figure 1. The horizontal axis and the vertical axis represent the state and likelihood respectively. Circles represent the particles while boxes represent the specific fulcrums and their likelihoods (the red dotted lines) are known. In the explicit Li-fitting form, the likelihoods of the fulcrums are fitted with their states to obtain the Li-function, as represented by the red curve. This curve is then used to obtain the

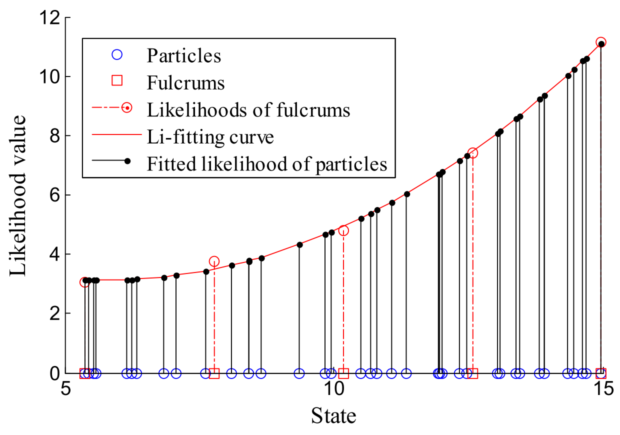

Figure 1. Schematic diagram of the fitting-based particle likelihood calculation.

particles' likelihood i.e. the height of black lines. In the implicit Li-fitting way, we may not obtain the red curve but just use the nearby fulcrums to infer the likelihood of the particle by utilizing a smoothing estimator. In both, no matter how many particles there are, we can fit or smooth all of them to obtain their likelihoods from the fulcrums.

The framework of the explicit Li-fitting based particle filter is described in Algorithm 1. The details of the algorithm are given in the following subsections.

Remark 1
Our approach does not have to be based on the SIR filter but it can be also combined with advanced PFs such as Auxiliary PF [30] and Gaussian PF [31].

Remark 2
It has become a common idea to develop analytic technologies to improve the PF, including the regularized PF (RPF) [32], kernel PF (KPF) [33], convolution PF [34] and feedback PF [35] but with very different implementations and purposes. In addition, Gaussian mixtures (GMs) are used [36] to represent the posterior PDF and the observation likelihood function. However, these PFs do not run faster than the basic SIR PF if the same number of particles is used.

Algorithm 1*:* Explicit Li-fitting based PF (one iteration)

```
Input: $S_{t-1}=\left\{x_{t-1}^{(i)}, w_{t-1}^{(i)}\right\}_{i=1}^{N}$, Output: $S_t=\left\{x_t^{(i)}, w_t^{(i)}\right\}_{i=1}^{N}$
  1. Selective Resampling
     Resample if the variance of normalized weights is greater than a pre-specified threshold; (Asymptotically)
     unbiased resampling that equally weights resampled particles and computes fast e.g. minimum-sampling
     variance resampling that maintains the original distribution the best [28] is preferred.
  2. Prediction
     $\forall: i = 1 \to N$, sample from the proposal:
        $x_t^{(i)} \sim q\left(x_t^{(i)} \middle| x_{t-1}^{(i)}, y_{0:t}\right)$
  3. Li-function construction
     3.1 Construct $M$ fulcrums (see subsection 3.2): $(X_1, X_2, \dots X_M)$
     3.2 Calculate their likelihoods: $Y_m = p(y_t|x_t)$ of $X_m$, $m = 1,2,\dots,M$
     3.3 Numerically fit the likelihood of fulcrums with their states to get the Li-function $L_t(\cdot)$ (see subsections
         3.3, 3.4), which satisfies:
         $(Y_1, Y_2, \dots Y_M) \cong L_t(X_1, X_2, \dots X_M)$
  4. Updating
     Update the weights by using their fitted value of $L_t(\cdot)$, i.e., $\forall: i = 1 \to N$, update the weight:
        $w_t^{(i)} \propto w_{t-1}^{(i)} \dfrac{L_t\left(x_t^{(i)}\right) p\left(x_t^{(i)} \middle| x_{t-1}^{(i)}\right)}{q\left(x_t^{(i)} \middle| x_{0:t-1}^{(i)}, y_{0:t}\right)}$
     Normalize the weights: $w_t^{(i)} = w_t^{(i)} / \sum_j^N w_t^{(j)}$
```

3.2. Non-negligible support fulcrums

There are two methods to construct fulcrums: One method is to simply use some particles from the particle set (non-uniformly distributed) as fulcrums and the other is to create new data-points in the state space (uniformly distributed). The more fulcrums are more likely to obtain a better fitting approximation, but at the higher cost of computation. The fulcrums should be distributed appropriately so that they can correctly infer the likelihoods of all the particles. In our approach, a grid-based method is adopted to generate uniformly distributed fulcrums that cover the non-negligible region.

By partitioning the state space into rectangular cells, the fulcrums can be easily created by the centers or the corners of those cells. This method of approximating the probability density by rectangular delimited data-points is flexible and convenient for implementation (the data structure created is easy to handle in computers). This data model has also been employed in the PM filter. Specifically, the anticipative boundary-based grid and the non-negligible support principle proposed in [1, 2] are readily suitable for our approach by a sensible albeit convenient conversion from the predictive PDF in the PM filter to the PF.

In contrast to interpolation, which predicts within the range of values in the dataset used for model fitting, prediction outside this range of the data is known as extrapolation. The further the extrapolation goes outside the data, the more likely it is for the model to fail due to differences between the fitting assumptions and the sample data or the true values. To avoid this, the fulcrums are constructed in the state space I_t to cover the complete state-space of particles with a boundary margin r

$$\forall l \in [1, d_x] : I_t^{[l]} = \left[\min x_t^{[l]} - r^{[l]}, \max x_t^{[l]} + r^{[l]}\right] \tag{9}$$

where $x_t^{[l]}$ is the state value in the lth coordinate, min and max functions are calculated with respect to all particles, d_x is the total dimensionality of the state, $r^{[l]}$ is the lth dimension boundary margin which will be determined on the state noise covariance matrix as in [1]

$$r^{[l]} = a\sqrt{\bar{Q}_t^{[l]}} \tag{10}$$

where a is a parameter that determines the non-negligible level, $\bar{Q}_t^{[l]}$ is the lth diagonal element of the transformed $\bar{Q}_t = T_t^T Q_t T_t$, Q_t is the covariance of state noise u_t and T_t is an orthogonal matrix composed of the eigenvectors of the estimate of the predictive covariance matrix; for detailed explanation see [1, 2]. In fact, as long as $r^{[l]}$ is a small positive value, it does not affect much the fitting result. For simplicity, we suggest to determine it simply based on the width of the particle distribution, e.g.

$$\forall l \in [1, d_x] : r^{[l]} = \frac{\max x_t^{[l]} \text{-} \min x_t^{[l]}}{2 \times M_t^{[l]}} \tag{11}$$

where $M_t^{[l]}$ is the number of grids to partition in dimension l. It will finally determine the number of fulcrums to obtain. A lower bound of it is given in [1] as follows

$$M_t^{[l]} \geqslant \frac{\left\| I_t^{[l]} \right\|}{2\gamma} \left(\bar{Q}_t^{[l]}\right)^{-1/2} \times \max_{x_t \in \Omega_t} \left| \det \frac{\partial h(x_t)}{\partial x_t} \right| \tag{12}$$

where Ω_t is a significant support of the predictive PDF $p(x_t | y_{t-1})$, $h(x)$ is the observation function, $\gamma > 0$ is the second design parameter. This is determined according to the user's preference of the accuracy.

Since the fulcrum is the crossing point of the partitioning of each dimension, the total number of fulcrums is just the product of the partitioning times of all dimensions

$$M_t = \prod\nolimits_{l=1}^{d_x} M_t^{[l]} \tag{13}$$

If we delimit the fulcrums with a fixed interval $d^{[l]}$, i.e.

$$d^{[l]} = \frac{I_t^{[l]}}{M^{[l]} - 1} \tag{14}$$

Then, the fulcrums can be defined at the space crossing of the following coordinates:

$$\forall m \in \left[1, M_t^{[l]}\right] : x_{t,m}^{[l]} = \min x_t^{[l]} - r^{[l]} + (m-1)\, d^{[l]} \tag{15}$$

Choosing a sensible number of fulcrums with respect to the state noise is important in our approach. For simplicity and fast online computation in multiple dimension situations, the following number $M^{[l]}$is suggested in our application such that

$$\forall l \in [1, d_x] : M_t^{[l]} = p \text{ or } 1 \tag{16}$$

where p is a specified value loosely satisfying (13), $M^{[l]} = 1$ means that the lth dimension is not partitioned.

To note, fulcrums can be added into the particle set. This will not increase additional likelihood computation as it has already been calculated. However, the total number of particles will be increased, which can be reduced in the process of resampling [10, 20, 21].

Remark 3
The primary limitation of the grid partitioning is due to the sensitivity to the dimensionality of the state space. To mitigate this limitation, partitioning the state space only in primary dimensions is recommended. For example, in the case of the state that consists of position, velocity, etc., the grid partitioning can be realized only in the position space (as shown in Section 4.2).

3.3. *Least squares numerical fitting*

Numerical fitting is accomplished in practice by selecting a linear or nonlinear function

$$y = f\left(x; c_1, c_2 \ldots, c_k\right) \tag{17}$$

that depends on certain parameters $c_1, c_2, \ldots, c_k$. There are two forms of numerical fitting: regression and interpolation, which are distinguished from one another based on whether the function works on the data (that is interpolation) or not (regression). In this paper, we concern with the regression for which the fitting data may not strictly work on the function, instead a fitting error generally exists, i.e.

$$y_m = f\left(x_m; c_1, c_2 \ldots, c_k\right) + e_m \tag{18}$$

where y_m is the measured value of the dependent variable, $c_1, c_2, \ldots, c_k$ are the required parameters. In our approach, the given data (x_m, y_m), $m = 1, 2 \ldots, M$ are fulcrums, x is the state, y is the likelihood, and $f(\cdot)$ is the required Li-function. The dependence of the likelihood function on the parameters can be either linear or nonlinear. For the nonlinear likelihood function, solutions include approximate linearization with tolerable errors (see Appendix) and conversion methods of the nonlinearity (see subsection 3.5). Otherwise, a nonlinear regression method is required, such as the Gauss-Newton method.

In what follows, we first consider the basic univariate variable fitting, while the intractable multivariate fitting will be described in subsection 3.4 where a local smoothing strategy (implicit Li-fitting approach) is proposed for convenient implementation.

Normally, one will try to select a function $L(x)$ that depends linearly on the parameters, in the form of

$$L(x) = c_1\phi_1(x) + c_2\phi_2(x) + \ldots + c_k\phi_k(x) \tag{19}$$

where $\{\Phi_i(x)\}$ are a priori selected sets of functions, for example, the set of monomials $\{x^{i-1}\}$ or the set of trigonometric functions $\{\sin \pi i x\}$, and $\{c_i\}$ are parameters which must be determined. In this paper, we call k the order of the fitting function. In over-determined systems, as in our case, k is much smaller than the number M of fulcrums

$$M >> k \tag{20}$$

To specify the form of the functions in (19), the best case is when the function is known in advance. Otherwise, reasonable assumptions and offline searching for the optimal fitting model is necessary. To find the optimal fitting model, offline study might be helpful. Once the approximating function form and fulcrums have been defined, the next step is to determine the population parameters $c_1, c_2, \ldots, c_k$ to obtain a "good" approximation. As a general idea, the residuals

$$d_m = f_m - L\left(x_m; c_1, c_2 \ldots, c_k\right), \quad m = 1, 2, \ldots M \tag{21}$$

are simultaneously made as small as possible. One tries to make some norm of the M-vector $\mathrm{d} = [d_1, d_2, \ldots, d_M]^T$ as small as possible - typically such as the 2-norm

$$\|\mathrm{d}\|_2 = \left(\sum_{m=1}^{M} |d_m|^2\right)^{1/2} \tag{22}$$

This leads to a linear system of equations to determine the minimum $\hat{c}_k$'s. The resulting approximation $L(x; \hat{c}_1, \hat{c}_2, \ldots, \hat{c}_k)$ is known as the least squares approximation to the given data and $\hat{c}_k$'s are called least squares estimates of the population parameters.

An appropriate fitting model and proper distributed fulcrums are two critical factors needed to achieve good fitting results. The Goodness-of-fit could be tested to decide whether it is possible to proceed or search for a more suitable Li-function model, one that will better represent the true observation. Available Goodness-of-fit tests include the Kolmogorov-Smirnov test, Anderson-Darling test, Chi-Square test, etc. [37].

3.4. Piecewise fitting and smoothing: Implicit Li-fitting

Often, however, it is difficult or even impossible to find a single function to represent the likelihood function in the entire state space, especially for intractable multivariate fitting (namely, Hypersurface problem). As such, a flexible piecewise constant form could be chosen where the fitting function is of lower order. Accompanied with the piecewise fitting/local smoothing strategy, the linearization of the nonlinear dependence on parameters will be more theoretically tenable and easier to implement. This can also reduce the required fitting function order that promises a smaller linearization error; see the analysis given in Appendix.

To perform the *Piecewise/Segmented fitting*, the independent variable is partitioned into intervals, and then a separate segment is fitted to each interval and the boundaries between the segments. It is shown in [38] that the piecewise constant approximations are the best when the densities are reasonably smooth in the scale of the grid. This indicates that the piecewise intervals should be partitioned such that the likelihood function in each interval is reasonably smooth.

We reiterate that our goal is to calculate the likelihood of particles but not the Li-function, which is only an intermediate process. If we can calculate the likelihood directly and accurately, we do not need to burden the Li-function. Thus, in the piecewise fitting, we can use nonparametric local smoothing techniques, e.g. Kernel Density Estimator (KDE), to derive the likelihood of particles by using the fulcrums without explicitly obtaining the Li-function. This method, termed the implicit Li-fitting approach, will greatly simplify the multivariate fitting for convenient implementation. Next, we will illustrate how it works. For a particle with state $x_t^{(j)}$, denoting its nearest M_j fulcrums in a limited scale and their likelihoods as $\{x_i, L_i\}_{i=1,2,\ldots,M_j}$, the required likelihood $p\left(y_t^{(j)}|x_t^{(j)}\right)$ can be defined as the Nadaraya-Watson kernel-weighted average of the likelihoods of these fulcrums

$$p\left(y_t^{(j)}\left|x_t^{(j)}\right.\right) = \frac{\sum_{i=1}^{M_j} K_{h_\lambda}\left(x_t^{(j)}, x_i\right) L_i}{\sum_{i=1}^{M_j} K_{h_\lambda}\left(x_t^{(j)}, x_i\right)} \tag{23}$$

where $K(\cdot)$ the kernel function. Two quite convenient kernel smoothers are available: the nearest neighbor smoother and the uniform kernel average smoother. The idea of the nearest neighbor (NN) smoother is as follows. For each point $x_t^{(j)}$, take M_j nearest neighbor fulcrums x_i and estimate the likelihood of the particle $p\left(y_t^{(j)}|x_t^{(j)}\right)$ by averaging the values of these neighbors likelihood. Formally,

$$K_{NN}\left(x_t^{(j)}, x_i\right) = h \tag{24}$$

In contrast to this, the uniform kernel function can be defined as

$$K_{uniform}\left(x_t^{(j)}, x_i\right) = \frac{h}{\left\|x_t^{(j)} - x_i\right\|} \tag{25}$$

As the estimate of this uniform kernel smoother, every fulcrum x_i in the bounded interval h contributes to the likelihood of particle $x_t^{(j)}$ in a manner inversely proportional to their distance from the particle. The NN smoother and the uniform kernel smoother will be both applied in our simulations in Section 4.

3.5. *A polynomial fitting example*

While boosting the processing speed of the PF, it is important to guarantee the estimate accuracy. In order to have an intuitive understanding of the numerical fitting process and its results, the following popular univariate SSM is considered. The system dynamic and observation equations are, respectively,

$$x_t = \frac{x_{t-1}}{2} + \frac{25x_{t-1}}{1 + x_{t-1}^2} + 8\cos(1.2(t-1)) + u_t \tag{26}$$

$$y_t = 0.05x_t^2 + v_t \tag{27}$$

where u_t and v_t are zero mean Gaussian random variables with variance 10 and 1 respectively.

Assuming the unknown observation equation is $y_t^{(i)} = g\left(x_t^{(i)}\right)$, the likelihood function can be obtained through one more step, i.e. the following Gaussian model

$$L_t\left(y_t^{(i)}\right) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(y_t - y_t^{(i)}\right)^2\right) \tag{28}$$

where y_t is the real observations and $y_t^{(i)}$ is the observation of $x_t^{(i)}$. As shown, the likelihood function is in fact nonlinear (most commonly it belongs to the exponential family). Instead of

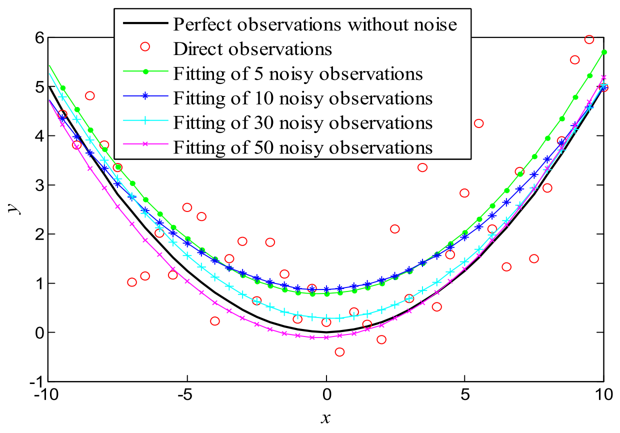

Figure 2. Observations and Equation (30)-based fitting function using different number of fulcrums.

using nonlinear fitting methods that are sometimes quite complex to implement, one may choose to linearize the nonlinearity. Equation (28) can be linearized by taking its natural logarithm to yield

$$\ln L_t\left(y_t^{(i)}\right) = \ln \frac{1}{\sqrt{2\pi}} - \frac{1}{2}\left(y_t - y_t^{(i)}\right)^2 \tag{29}$$

Thus, the function $\ln L_t(x)$ with independent variable x has a linear dependence on the parameters (substituting $y_t^{(i)} = g\left(x_t^{(i)}\right)$ into the equation); refer to (19). However, for this model we only fit the observation function $y_t^{(i)} = g\left(x_t^{(i)}\right)$. Fulcrums can be uniformly distributed with parameter $r = 1$ and the 2-order polynomial in the following trinomial form is assumed as the observation equation

$$y = c_3 x^2 + c_2 x + c_1 \tag{30}$$

Then we obtain the Li-function $(L_t \circ g)\left(x_t^{(i)}\right)$, which is

$$L_t\left(x_t^{(i)}\right) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(y_t - c_3 x_t^{(i)2} - c_2 x_t^{(i)} - c_1\right)^2\right) \tag{31}$$

This is known as a nonlinear conversion that changes the nonlinear fitting function to a linear one, which has high potential to apply to SSM with Gaussian observation noise. In Figure 2, the 'ideal' true observations($y = 0.05x^2$) without noise are shown in 'black' curve and its noisy observations in (27) of 100 random samples are shown in red circles. The least squares fitting results of the noisy observations in (30) of M fulcrums ($M = 5, 10, 30, 50$ separately; for 2-order function, $M >> 2$ according to the condition (20)) are shown with respective colored lines.

The results show that the proposed fitting approach obtains more accurate observations than direct observation. These good results benefit from our pre-knowledge that the observation equation is a 2-order polynomial, although this is a fairly weak assumption. Obviously, the fitting results can grow stronger as the knowledge of the observation model improves. This will be further shown in our simulation section 4.1. For example, if we know the fitting function is in the monomial form

$$y = c_3 x^2 \tag{32}$$

then more precise fitting results can be obtained (as shown in Figure 3) by using the same fulcrums. For example, in one trial, we obtain: $c_3 = 0.0570$ (5 fulcrums), $c_3 = 0.0518$ (10 fulcrums), $c_3 = 0.0530$ (30 fulcrums), $c_3 = 0.0487$ (50 fulcrums).

The previous example has just exhibited a potential application of the numerical fitting to estimate the observation function if it is unknown and needs to be estimated. Furthermore, if the observation

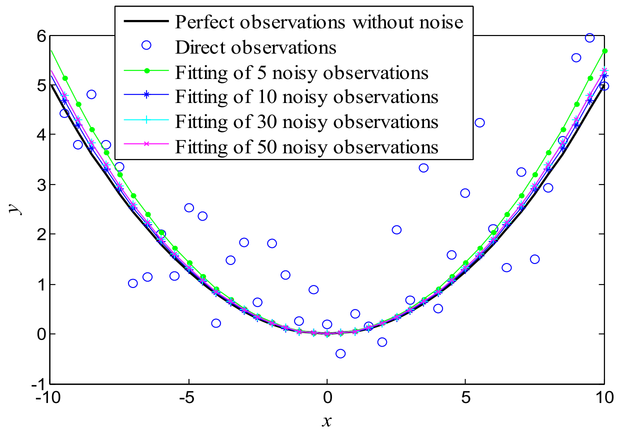

Figure 3. Observations with noise and Equation (32)-based fitting function using different numbers of fulcrums.

equation is known (as the case of most SSMs), the numerical fitting method can be applied in a batch manner in which it does not need to online fit the equation of (30) or (32) for each step. The fitted observation function or even the exact function $y = 0.05x^2$ can be taken as granted in the subsequent filtering steps to infer the likelihood of particles for saving computation. We refer to this form of fitting method as the batch fitting in which the function of interest is constant. We will demonstrate this in our simulations. However, we reiterate that, the Li-fitting approach is not intended to apply for such a model with known and simple observation function for which the likelihood calculation is extremely simple and is hard to be speeded up further. Instead, it is primarily targeted for cases such as robot localization (and visual tracking) where the likelihood calculation often involve complicated map/image processing which is much more computational intensive.

4. SIMULATIONS

For the sake of verifying the validity of our approach, typical SSMs including the aforementioned one-dimensional (1D) model and multiple dimensional robot localization are considered in this section.

4.1. *One-dimensional model: Fitting observation function*

For the nonlinear system described in (26)–(27), the root mean square error (RMSE) in the time series is used to evaluate the estimation accuracy, which is calculated by

$$\text{RMSE} = \left(\frac{1}{T}\sum_{t=1}^{T}(x_t - \hat{x}_t)^2\right)^{1/2} \tag{33}$$

where $\hat{x}$ is the estimate of the state, T is the sum of iterations. A big T =10,000 is chosen and a sequence number of particles from 10 to 500 with interval of 10 is separately used.

First, different orders of polynomial with 10 fulcrums are used in the regression model (30) to fit the observation function. The RMSE results of the Li-fitting based PFs are given in Figure 4, which shows that the 1st-order polynomial fitting result is really poor whereas 2nd-order and higher form polynomials obtain much better estimation accuracy. This indicates that a proper (no smaller than the true order) fitting function is critically important for our approach.

Secondly, the RMSE of the basic PF and the Li-fitting based PFs using different numbers of fulcrums (for 2nd-order fitting polynomial) is given in Figure 5. The results indicate that the Li-fitting PF can obtain a comparable estimation accuracy with the SIR filter, as long as an enough number of fulcrums are used.

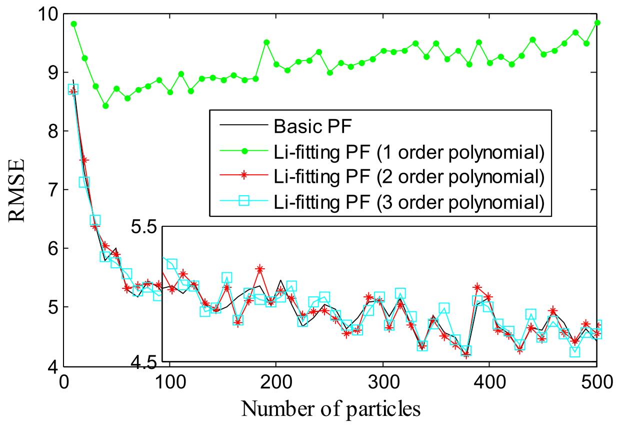

Figure 4. Root mean square error of the basic sampling importance resampling particle filter and the Li-fitting based PFs that use different orders of polynomials.

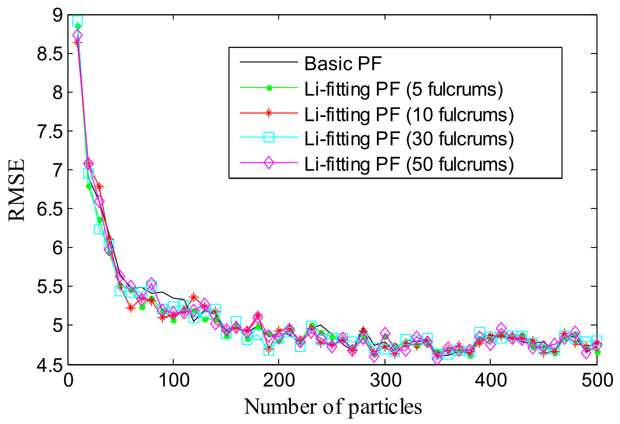

Figure 5. Root mean square error of the basic sampling importance resampling particle filter and the Li-fitting based PFs that use different numbers of fulcrums.

Thirdly, the comparison of the Li-fitting PF (including the batch form) with several other known nonlinear filters including the SIR PF, auxiliary PF (APF) [30], Gaussian PF (GPF) [31], Kernel PF (KPF) [33] and unscented Kalman filter (UKF, with the unscented transform parameter set as $\alpha = 1, \beta = 0, \kappa = 2$) [39] are given in Figures 6 and 7 for the RMSE and processing time, respectively. Here, the Li-fitting PF uses 10 fulcrums and fitting function (32) at all steps, while the batch Li-fitting PF uses 100 fulcrums and fitting function (32) at the first step. The average performance over different numbers of particles is given in Table I where the weight updating is not vectorized in Matlab as explained later.

The results show that the UKF does not work for this highly nonlinear model as its RMSE is high (the same as [31] shows) while all PFs perform very similarly. In detail, the GPF and the APF is somewhat inferior to the (batch) Li-fitting PFs, the SIR, and the KPF on average. When the number of particles is small, the batch Li-fitting PFs performs better than others.

The Matlab programming itself can highly affect the computing speed. Especially, the vectorization, that is, the data are processed in the unit of matrix can highly speed up the computation. The processing time of all filters are given in Figure 7 for the case without vectorization of the weight updating step and are given in Figure 8 for the case of using vectorization for updating. To note, when dimensions are correlated, vectorization might be infeasible. Then, the computational demand of the PFs (including GPF, KPF, APF, and SIR) will unsurprisingly increase in proportion with the growth of the number of particles used as shown in Figure 7; see our next simulation. For this general case, the UKF is obviously the fastest. The batch Li-fitting based PF is the second, GPF is the third (no resampling is needed), and the Li-fitting PF is the fourth. All of them are highly faster than

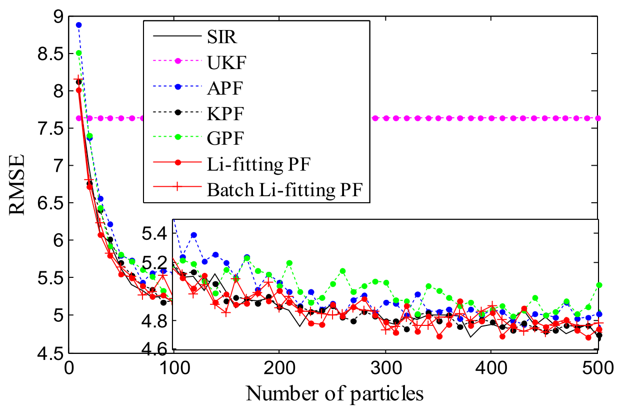

Figure 6. Root mean square error of different filters against the number of particles used.

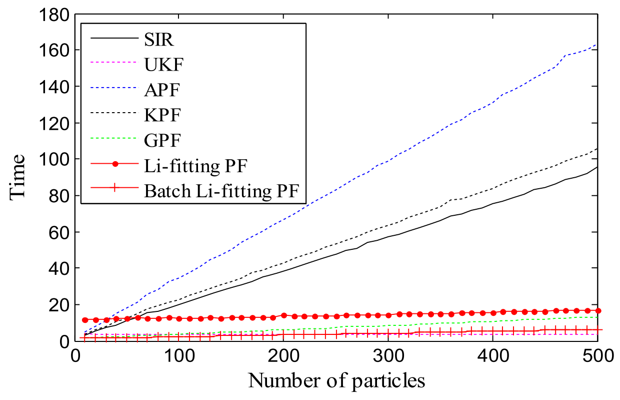

Figure 7. Computing time of different filters for 10,000 steps (the weight updating of particles is not vectorized).

the KPF, APF, and SIR especially when the number of particles is large. In contrast, in the Li-fitting PFs, the number of fulcrums is constant, and the calculation does not necessarily increase with the number of particles. This exactly demonstrates the superiority of our approach that releases the high dependence of the computational time on the number of particles used.

To note, the processing speed of the Li-fitting PF can only be improved when the time cost for the fitting is lesser than the likelihood computation it has saved. Because the updating step (27) when the vectorization is employed is nothing more than the job of solving (30), it is not surprising that the computing speed of the Li-fitting PF has not been improved but instead reduced in such a simple simulation when vectorization is utilized or when the number of particles is very small. However, in multiple dimensional model, the multi-dimensional state is often unable for the vectorization of the weight updating; see our next simulation.

In particular, in such a 1D model, the resampling will take a large part of the computation in the PF, and therefore, the GPF is very fast, which does not need to perform resampling. However, in high-dimension models where the weight updating is much more computation-consuming than the resampling and the state prediction, the computation advantage of the GPF will not be so obvious, but the Li-fitting will further speed up the PF. This will be demonstrated in our next simulation.

4.2. Robot localization

In general, the SIR PF seems to be the fastest computationally while existing variant PFs are often more computationally intensive except for the GPF which does not need to perform resampling. Therefore, in our second simulation, our comparison has not included all of these advanced PFs except SIR and GPF.

Table I. Average performance of filters (without vectorization).

	$N = 50$		$N = 200$	
	RMSE	Time	RMSE	Time
UKF	7.641	3.491	7.641	3.427
SIR	5.623	10.840	4.895	38.196
APF	5.805	18.044	5.060	66.885
KPF	5.706	12.514	4.896	42.910
GPF	5.821	2.412	5.038	5.955
Li-fitting	5.546	12.210	5.004	13.906
Batch Li-fitting	5.548	1.704	4.883	3.170

RMSE, root mean square error; UKF, unscented Kalman filter; SIR, sampling importance resampling; APF, auxiliary PF ; KPF, kernel PF; GPF, Gaussian PF.

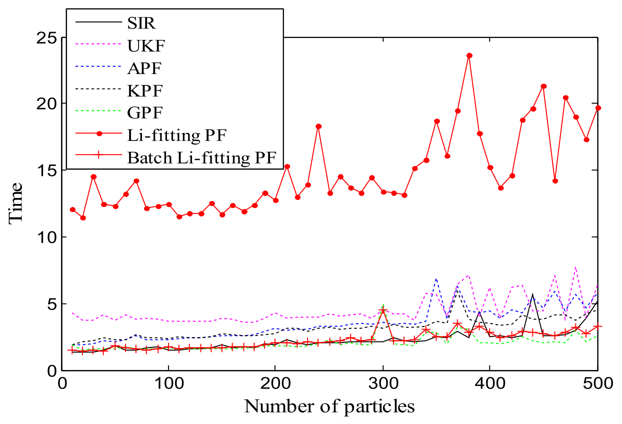

Figure 8. Computing time of different filters for 10,000 steps (the weight updating of particles is vectorized).

The application of the PF to mobile robot localization is usually referred to as the Monte Carlo localization (MCL) [13]. In this section, we present a typical MCL application via simulation for accurate comparison and analysis. In order to develop the details of MCL, let $x_t = (x, y, \theta)^T$ be the robot's position in Cartesian space (x, y) along with its heading direction θ denotes the robot's state at time instant t, y_t is the observation at time t, and u_t is the odometer data (control observation) between time $t-1$ and t. Supposing u_{t-1} has a movement effect $(\Delta s, \Delta\theta)^T$ on the robot, Δs is the translational distance, and $\Delta\theta$ the change of robot's heading direction from time $t-1$ to t. Then, the motion model $p(x_t|x_{t-1}, u_t)$ can be easily obtained as

$$x_t = x_{t-1} + \begin{bmatrix} \cos\theta_{t-1} & 0 \\ \sin\theta_{t-1} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta s \\ \Delta\theta \end{bmatrix} + u_t \tag{34}$$

where u_t is the system uniform noise with zero mean and $\text{diag}[\Delta s \times 20\%, \Delta\theta \times 5\%]^T$ variance in our case. In particular, the particle, which falls into obstructs, will be discarded (by setting its weight to zero) in resampling.

The observation model depends on the map of the environment (given in Figure 9) and the sensor data. Here, we assume the use of a single laser radar (having an arc view field of 180°) that measures a serial of (denoted as n) distances between the sensor and the surrounding obstacles (black areas in the figure) in different bearings/azimuths, forming a fan of multiple 'distance-lines' (as shown by blue lines in Figure 9 for one position). This involves map processing when calculating the observation $\hat{y}_t^{(i)}$of each particle $x_t^{(i)}$, which needs to calculate the lengths of all these 'distances' in n bearings. Then one can compute the likelihood for each particle based on $\hat{y}_t^{(i)}$ and y_t. Here, we assume the observation noise is Gaussian, and the nearest-neighbor data association is used for scan data matching in our simulation, which can be described as

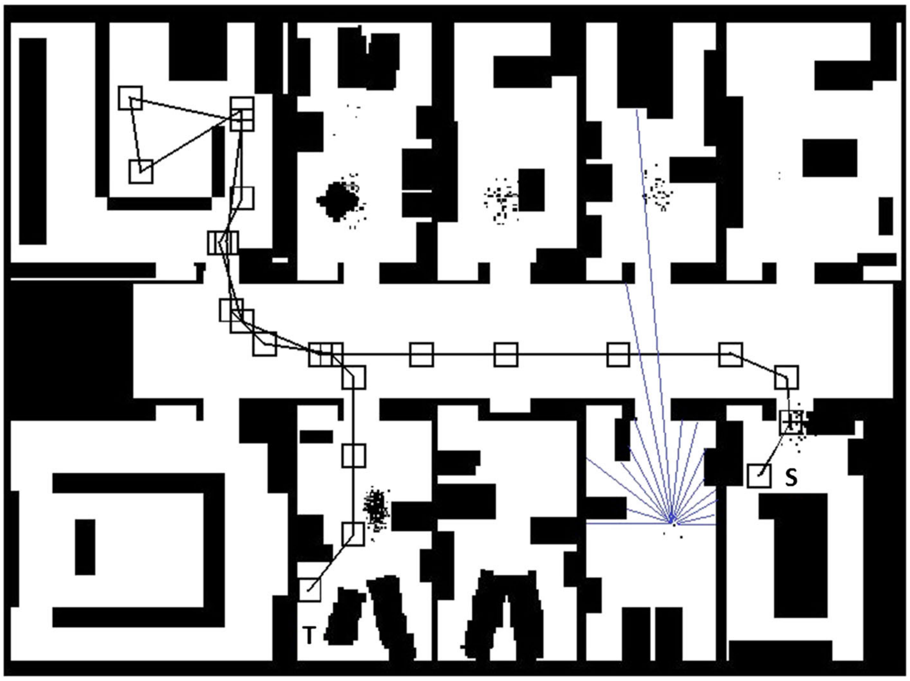

Figure 9. The robot trajectory and the distribution of particles (black point) when the robot is at the second stop; A fan of blue lines represent the scanning distances of a particle (simulated) in a view field of [0, 180°].

$$p\left(y_t \left| x_t^i\right.\right) \propto \frac{1}{\sqrt{(2\pi)^n \left|S_{i,t}\right|}} \exp\left(-\frac{1}{2}\left(\hat{y}_t^{(i)} - y_t\right)^T S_{i,j}^{-1}\left(\hat{y}_t^{(i)} - y_t\right)\right) \quad s.t.\ \hat{y}_t^{(i)} = g\left(x_t^i\right) \tag{35}$$

where $S_{i,j}$ is the covariance matrix of the difference $\hat{y}_t^{(i)} - y_t$; the Gaussian observation noise is $s \sim \mathcal{N}(0, 5)$ for each scanning distance. We choose $n = 36$ and 180, respectively, in our case. A bigger n indicates a higher resolution and is more time-consuming. The map processing as well as (35) indicate that it is impossible to apply vectorization for weight updating as is done in the one-dimensional SSM like (26)–(27).

The Li-functions could facilitate the entire state space $(x, y, \theta)^T$ or simplify the Cartesian space $(x, y)^T$ only. This simplification is possible because the direction θ relies strongly on its position $(x, y)^T$ if the covariance matrix $S_{i,j}$ can be known based on the observations. In this case, we set a constant boundary margin $r^{[l]} = 1$ in (10), $p = 10$ in (16) so that 100 fulcrums are used in the position space $(x, y)^T$, and the linear uniform fitting method is adopted. In addition, the Li-fitting approach is applied after $t = 3$, since at the starting stage, ($t \leqslant 3$) particles are very widely distributed (e.g., the case is plotted in Figure 9 for $t = 2$) which is unsuitable for constructing the Li-function. To determine whether it becomes suitable for fitting, one suggestion here is to measure the variance of the particle distribution. Here, we use the threshold method.

To evaluate the filtering performance, the Euclidean distance (ED) is defined between the position estimate $(\hat{x}, \hat{y})^T$ and the ground truth $(x, y)^T$ that is calculated by

$$\mathrm{ED} = \sqrt{(x - \hat{x})^2 + (y - \hat{y})^2} \tag{36}$$

The path of the robot is from 'S' to 'T' in Figure 9. The points represent particles, and the rectangle boxes represent the stops of robot. To gain more insights, the distribution of particles, and the surfaces of Li-function are given in Figure 10 for one trial when 500 particles and 100 fulcrums are used and $n = 36$. As shown, the particles concentrate in local areas, and their likelihood distribution is relatively smooth in the planar position space.

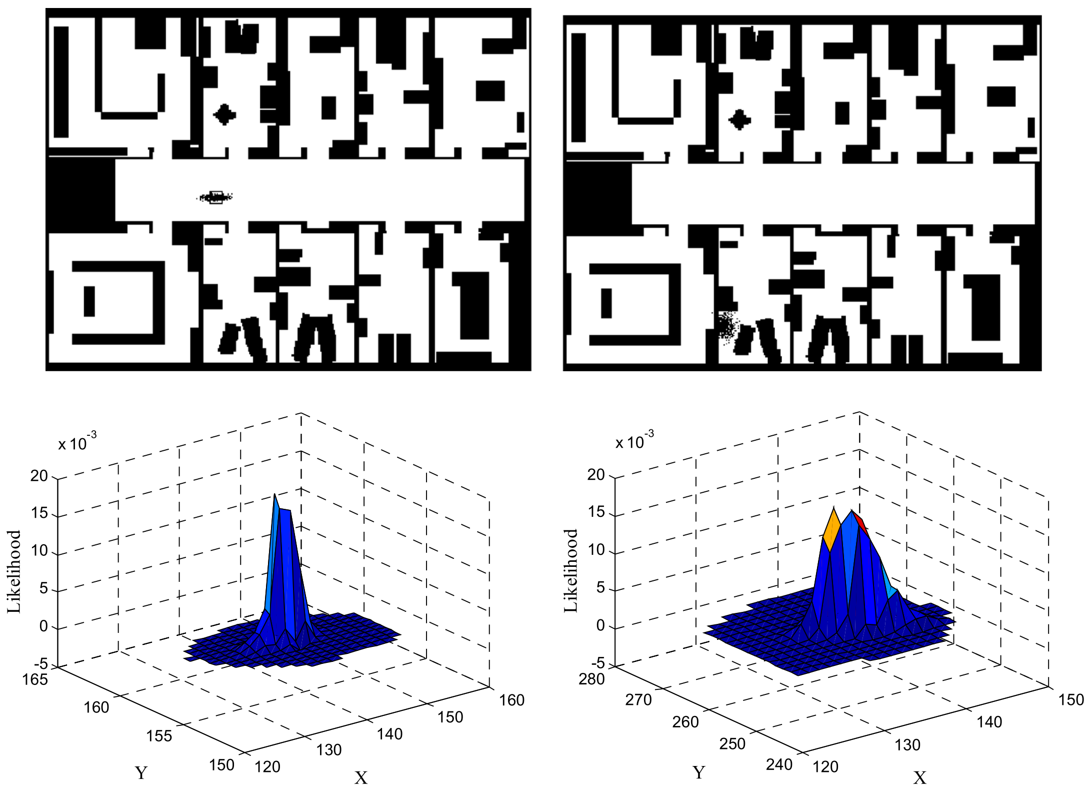

Figure 10. Distribution of particles (upper row), Li-fitting surfaces (bottom row) in two stops (N =500, M =100, n =36).

Both the number N of particles and the number M of fulcrums are critical to the performance of PFs. To capture the average performance, 100 MC trials were run. The EDs when different numbers N of particles are used are plotted by time steps in Figure 11, which indicates that the Li-fitting approach has indeed reduced the estimation accuracy somewhat as compared with the SIR PF and the GPF (the latter two perform similarly). For a large number of particles,for example, 500, a small number of fulcrums, for example, 100, can fit the likelihood efficiently. Furthermore, it can be seen that for the same number of fulcrums, applying more particles does not always generate better results. This indicates that it is not suitable to use too few fulcrums to fit the likelihood of too many particles; instead, the ratio of the number of fulcrums to the number of particles should be set in a reasonable scope. Too small a ratio (too few fulcrums) leads to worse results, while too high a ratio does not benefit the filter speed.

The mean ED from stop 3 to 24 against the number M of fulcrums is plotted in Figure 12, which shows that the larger the number of fulcrums, the more accurate is the approximation. The reason that the result is better for a larger number of particles is because of the very first steps (while in the latter steps, a greater number of particles leads to worse estimation accuracy). Note that the Li-fitting approach does not fit during the initial stage where particles are distributed broadly in the state space. The computing time of the Li-fitting based PF, SIR, and GPF against the number of particles are given in Table II and III respectively for the scanning data size n = 36 and 180. The results demonstrate again the fast processing advantage of our Li-fitting approaches, especially when informational-rich observations (n = 180) are applied.

The resampling that is dimension-free takes a large part of computation in the simple 1D model but not in this complicated multi-dimensional models. Here, the weight updating of particles is the primary computation of the PF. Therefore, the GPF is faster than the SIR but is much slower than the Li-fitting MCL. Comparably, the Li-fitting approach is qualified to significantly reduce the computation while maintaining approximation quality.

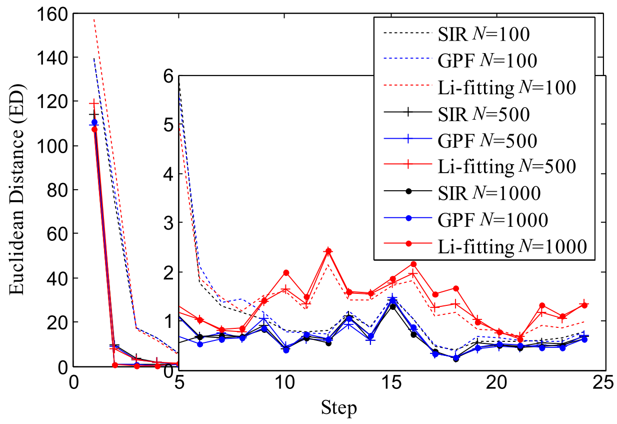

Figure 11. Estimation error by steps when different numbers of particles are used (n =36, M =100 in Li-fitting particle filters).

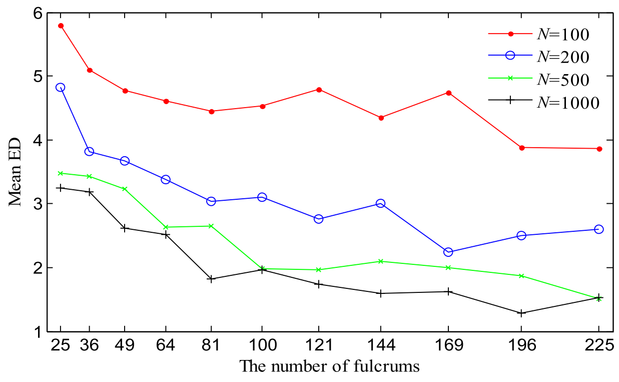

Figure 12. Mean Euclidean distance against the number of fulcrums used (n =36).

Table II. Real-time performance of PFs (*second*) when the sensor data size n =36.

Number of particles	100	500	1000
Basic SIR PF	0.593	3.668	10.078
Gaussian PF	0.594	3.654	9.5165
Li-fitting PF	0.787	2.140	4.1419

PF, particle filter; SIR PF, sampling importance resampling particle filter.

There is a trade-off between increasing the processing speed by reducing likelihood calculation and improving the estimation accuracy by maintaining accurate likelihood calculation. The choice depends on the practitioner's preference between the estimation accuracy and the processing speed. The Li-fitting approach provides a choice for applications in which a fast processing speed is much preferred.

5. DISCUSSIONS

The simulations have demonstrated the validity of the Li-fitting approach but also exhibited its extra limitations for the likelihood calculation for which counter measures are needed.

Table III. Real-time performance of PFs (*second*) when the sensor data size $n = 180$.

Number of particles	100	500	1000
Basic SIR PF	3.417	20.937	50.768
Gaussian PF	3.566	15.707	47.321
Li-fitting PF	3.806	6.547	11.871

PF, particle filter; SIR PF, sampling importance resampling particle filter.

(1) Straightforward numerical fitting may give negative likelihoods, which is not reasonable to update the particle weight that must be positive. To correct this, the negative likelihood can be set to zero.
(2) The implicit Li-fitting approach in the form of smoothing does not need to obtain the likelihood function directly and only needs two parameters: the number of fulcrums M and the boundary margin r. The latter can be set as a relatively small positive constant or determined by the distribution of particles as given in (11).
(3) When a fixed number of fulcrums are used, more particles do not always lead to better estimation. To obtain the optimal approximation, the ratio of the number of fulcrums used, M, to the number of particles N should be set within a reasonable scope, for which we suggest the scope of [1/5, 1/2].
(4) The Li-fitting approach is more suitable for "relatively stable tracking" stage in which the state transition is stable and the variance of the particle distribution is relatively small. Finding an ingenious solution to apply the numerical fitting approach to complicated state dynamics models still requires further research.
(5) Current Li-fitting applications use a fixed number of fulcrums and a constant form of fitting/smoothing function, which are simple but may not always be satisfactory because the complexity of the likelihood function often vary over time. Therefore, advanced/adaptive Li-fitting approaches that are able to adjust the number of fulcrums and the fitting/smoothing function according to the system requirement would be valuable.

6. CONCLUSIONS

This paper has proposed a numerical fitting approach for calculating the particle likelihood to speed up the particle filter, termed the Li-fitting approach. It uses a relatively small number of fulcrums to infer the likelihood of particles and is particularly efficient when the observation/likelihood function is complicated and computationally intensive. The Li-fitting approach alleviates the proportional dependence of the computational demand of the PF on the number of particles used, exhibiting a significant mollification of the contradiction between the computational cost and the approximation accuracy. It has been detailed how the numerical fitting method can be implemented in explicit fitting and implicit smoothing forms. Meanwhile, an awareness of the limitations concerning the Li-fitting approach was also discussed with potential solutions provided. Simulation results have demonstrated that the processing speed of the PF has been highly accelerated by the Li-fitting approach without significantly losing the estimation accuracy.

The future work will be twofold: further development of advanced adaptive Li-fitting approaches that are capable of adjusting the number of fulcrums and the fitting function for more challenging environments; and the employment of the numerical fitting tool for parameter estimation within the context of statistical signal processing.

APPENDIX: LINEARIZATION ERROR OF LI-FITTING

For engineering convenience, one may directly assume the likelihood function having linear dependence on the parameters as in (19). However, the assumption of the linearity dependence does not hold for most practical systems. This appendix gives the linearization error of converting a nonlinear fitting model to be one linear function. The analysis is based on the Taylor series expansion.

A Taylor series is a series expansion of a function about a point. A one-dimensional Taylor series of a real function $f(x)$ about a point $x = x_0$ is given by

$$f\,(x) = f\,(x_0) + f'\,(x_0)\,(x - x_0) + \cdots + \frac{f^{(n)}(x_0)}{n!}\,(x - x_0)^n + R_n \tag{A.1}$$

where R_n is a remainder term known as the Lagrange remainder (or truncation error), which is given by

$$R_n = \frac{f^{(n+1)}\,(x^*)}{(n+1)!}\,\left(x - x_0\right)^{n+1} \tag{A.2}$$

where $x^* \in [x_0, x]$ lies somewhere in the interval from x_0 to x.

Thus, if we use the engineering-friendly monomials function as in (19), there is at least an error of R_n occurring. The expression R_n in (A.2) indicates that the closer the prediction data x is with x_0 (the smaller (x-x_0) is), the more feasible it is to express the function in that interval in a lower order and with a smaller R_n. That is to say, the closer the particle is to the fulcrums (i.e., the smaller the piecewise interval), the more accurate and reliable the Li-fitting approach will be. This explains why the piecewise fitting is suggested in our approach to deal with the trade-off between better estimation accuracy and faster computing speed.

ACKNOWLEDGMENTS

This work is partly supported by National Natural Science Foundation of China (ref. 51475383) and European Commission: FP7-PEOPLE-2012-IRSES (ref. 318878) and MSCA-RISE-2014 (ref. 641794).

REFERENCES

1. Šimandl M, Královec J, Söderström T. Anticipative grid design in point-mass approach to nonlinear state estimation. *IEEE Transactions on Automatic Control* 2002; **47**(4):699–702.
2. Šimandl M, Královec J, Söderström T. Advanced point-mass method for nonlinear state estimation. *Automatica* 2006; **42**:1133–1145.
3. Lefebvre T, Bruyninckx H, Schutter J. Comment on a new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control* 2002; **47**(8):1406–1408.
4. Mallick M, Sindhu B. Critical analysis of the particle flow filter. *Proceedings of 2015 International Conference on Control, Automation and Information Sciences (ICCAIS)*, Changshu, China, 2015; 512–517.
5. Andrieu C, Doucet A, Holenstein R. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society. Series B* 2010; **72**:269–302.
6. Cappé O, Godsill SJ, Moulines E. An overview of existing methods and recent advances in sequential Monte Carlo. *Proceedings of the IEEE* 2007; **95**(5):899–924.
7. Doucet A, Johansen AM. A tutorial on particle filtering and smoothing: fifteen years later. In *Handbook of Nonlinear Filtering*, Crisan D, Rozovsky B (eds). Oxford University Press: Oxford, 2009; 656–704.
8. Elvira V, Míguez J, Djurić PM. *Adapting the number of particles in sequential Monte Carlo methods through an online scheme for convergence assessment*, 2015. arXiv:1509.04879.
9. Kwok C, Fox D, Meilă M. Real-time particle filters. *Proceedings of the IEEE* 2004; **92**(3):469–484.
10. Li T, Sun S, Sattar TP. Adapting sample size in particle filters through KLD-resampling. *Electronics Letters* 2013; **46**(12):740–742.
11. Van Leeuwen PJ. Nonlinear data assimilation in geosciences: an extremely efficient particle filter. *Quarterly Journal of the Royal Meteorological Society* 2010; **136**(653):1991–1999.
12. Thrun S, Burgard W, Fox D. Probabilistic Robotics. MIT Press: London, 2005; 191–280.
13. Yu M, Liu C, Chen WH, Chambers J. A Bayesian framework with an auxiliary particle filter for GMTI-based ground vehicle tracking aided by domain knowledge. *Proceedings of SPIE - The International Society for Optical Engineering*, 2014; 90911I-1–90911I-10.
14. Gning A, Ristic B, Mihaylova L, Abdallah F. Introduction to box particle Filtering. *IEEE Signal Processing Magazine* 2013; **30**(4):166–171.
15. Lymperopoulos I, Lygeros J. Sequential Monte Carlo methods for multi-aircraft trajectory prediction in air traffic management. *International Journal of Adaptive Control and Signal Processing* 2010; **24**:830–849.

16. Li T, Sun S, Sattar TP. High-speed sigma-gating SMC-PHD filter. *Signal Processing* 2013; **93**(9):2586–2593.
17. Liu H, Sun F. Efficient visual tracking using particle filter with incremental likelihood calculation. *Information Sciences* 2012; **195**:141–153.
18. Hendeby G, Karlsson R, Gustafsson F. Particle filtering: the need for speed. *EURASIP Journal on Advances in Signal Processing* 2010; **2010**:1–9.
19. Mihaylova L, Hegyi A, Gning A, Boel R. *IEEE Transactions on Intelligent Transportation Systems* 2012; **13**(1): 36–48.
20. Li T, Bolić M, Djuric P. Resampling methods for particle filtering: classification, implementation, and strategies. *IEEE Signal Processing Magazine* 2015; **32**(3):70–86.
21. Li T, Villarrubia G, Sun S, Corchado JM, Bajo J. Resampling methods for particle filtering: identical distribution, a new method and comparable study. *Frontiers of Information Technology & Electronic Engineering* 2015; **16**(11):969–984.
22. Li T, Sun S, Bolić M, Corchado JM. Algorithm design for parallel implementation of the SMC-PHD filter. *Signal Processing* 2016; **119**:115–127.
23. Doucet A, Godsill S, Andrieu C. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing* 2000; **10**(3):197–208.
24. Chen T, Schön TB, Ohlsson H, Ljung L. Decentralized particle filter with arbitrary state decomposition. *IEEE Transactions on Signal Processing* 2011; **59**(2):465–478.
25. Djuric PM, Lu T, Bugallo MF. Multiple particle filtering. *Proceedings of IEEE 32nd ICASSP*, Honululu, Hawaii, United States, 2007; III-1181–III-1184.
26. Vaswan N. Particle Filtering for large-dimensional state space with multimodal observation likelihood. *IEEE Transactions on Signal Processing* 2008; **56**(10):4583–4597.
27. Givon D, Stinis P, Weare J. Variance reduction for particle filters of systems with time scale separation. *IEEE Transactions on Signal Processing* 2009; **57**(2):424–435.
28. Li T, Sun S, Sattar TP, Corchado JM. Fight sample degeneracy and impoverishment in particle filters: a review of intelligent approaches. *Expert Systems With Applications* 2014; **41**(8):3944–3954.
29. Marin JM, Pudlo P, Robert RC, Reder R. Approximate Bayesian computational methods. *Statistics and Computing* 2012; **22**:1167–1180.
30. Pitt MK, Shephard N. Filtering via simulation: auxiliary particle filters. *Journal of the American Statistical Association* 1999; **94**(446):590–591.
31. Kotecha JH, Djuric PM. Gaussian particle filtering. *IEEE Transaction on Signal Processing* 2003; **51**(10):2592–2601.
32. Musso C, Oudjane N, LeGland F. Improving regularised particle filters. In *Sequential Monte Carlo Methods in Practice*, Doucet A, de Freitas JFG, Gordon NJ (eds). Springer-Verlag: New York, 2001; 247–271.
33. Chang C, Ansari R. Kernel particle filter for visual tracking. *IEEE Signal Processing Letters* 2005; **12**(3):242–245.
34. Campillo F, Rossi V. Convolution particle filter for parameter estimation in general state-space models. *IEEE Transactions on Aerospace and Electronic Systems* 2009; **45**(3):1063–1072.
35. Yang T, Mehta PG, Meyn SP. Feedback particle filters. *IEEE Transactions on Automatic Control* 2013; **58**(10): 2465–2480.
36. Han B, Zhu Y, Comaniciu D, Davis LS. Visual tracking by continuous density propagation in sequential Bayesian filtering framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2009; **31**(5):919–930.
37. Hayashi F. *Econometrics*. Princeton University Press: Princeton, United States, 2000.
38. Kramer SC, Sorenson HW. Recursive Bayesian estimation using piece-wise constant approximations. *Automatica* 1988; **24**(6):789–801.
39. Julier SJ, Ulhmann JK, Durrant-Whyte HF. A new method for nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control* 2000; **45**(3):472–482.