Particle Filter Theory and Practice with Positioning Applications

FREDRIK GUSTAFSSON, Senior Member, IEEE
Linköping University
Sweden

The particle filter (PF) was introduced in 1993 as a numerical approximation to the nonlinear Bayesian filtering problem, and there is today a rather mature theory as well as a number of successful applications described in literature. This tutorial serves two purposes: to survey the part of the theory that is most important for applications and to survey a number of illustrative positioning applications from which conclusions relevant for the theory can be drawn.

The theory part first surveys the nonlinear filtering problem and then describes the general PF algorithm in relation to classical solutions based on the extended Kalman filter (EKF) and the point mass filter (PMF). Tuning options, design alternatives, and user guidelines are described, and potential computational bottlenecks are identified and remedies suggested. Finally, the marginalized (or Rao-Blackwellized) PF is overviewed as a general framework for applying the PF to complex systems.

The application part is more or less a stand-alone tutorial without equations that does not require any background knowledge in statistics or nonlinear filtering. It describes a number of related positioning applications where geographical information systems provide a nonlinear measurement and where it should be obvious that classical approaches based on Kalman filters (KFs) would have poor performance. All applications are based on real data and several of them come from real-time implementations. This part also provides complete code examples.

Manuscript received June 18, 2008; revised January 26 and June 17, 2009; released for publication September 8, 2009.

Refereeing of this contribution was handled by L. Kaplan.

Author's address: Dept. of Electrical Engineering, Linköping University, ISY, Linköping, SE-58183, Sweden, E-mail: (Fredrik@isy.liu.se).

I. INTRODUCTION

A dynamic system can in general terms be characterized by a state-space model with a hidden state from which partial information is obtained by observations. For the applications in mind, the state vector may include position, velocity, and acceleration of a moving platform, and the observations may come from either internal onboard sensors (the navigation problem) measuring inertial motion or absolute position relative to some landmark or from external sensors (the tracking problem) measuring for instance range and bearing to the target.

The nonlinear filtering problem is to make inference on the state from the observations. In the Bayesian framework, this is done by computing or approximating the posterior distribution for the state vector given all available observations at that time. For the applications in mind, this means that the position of the platform is represented with a conditional probability density function (pdf) given the observations.

Classical approaches to Bayesian nonlinear filtering described in literature include the following algorithms:

1) The Kalman filter (KF) [1, 2] computes the posterior distribution exactly for linear Gaussian systems by updating finite-dimensional statistics recursively.

2) For nonlinear, non-Gaussian models, the KF algorithm can be applied to a linearized model with Gaussian noise with the same first- and second-order moments. This approach is commonly referred to as the extended Kalman filter (EKF) [3, 4]. This may work well but without any guarantees for mildly nonlinear systems where the true posterior is unimodal (just one peak) and essentially symmetric.

3) The unscented Kalman filter (UKF) [5, 6] propagates a number of points in the state space from which a Gaussian distribution is fit at each time step. UKF is known to accomodate also the quadratic term in nonlinear models, and is often more accurate than EKF. The divided difference filter (DDF) [7] and the quadrature Kalman filter (QKF) [8] are two other variants of this principle. Again, the applicability of these filters is limited to unimodal posterior distributions.

4) Gaussian sum Kalman filters (GS-KFs) [9] represent the posterior with a Gaussian mixture distribution. Filters in this class can handle multimodal posteriors. The idea can be extended to KF approximations like the GS-QKF in [8].

5) The point mass filter (PMF) [10, 9] grids the state space and computes the posterior over this grid recursively. PMF applies to any nonlinear and non-Gaussian model and is able to represent any posterior distribution. The main limiting factor is the curse of dimensionality of the grid size in higher state

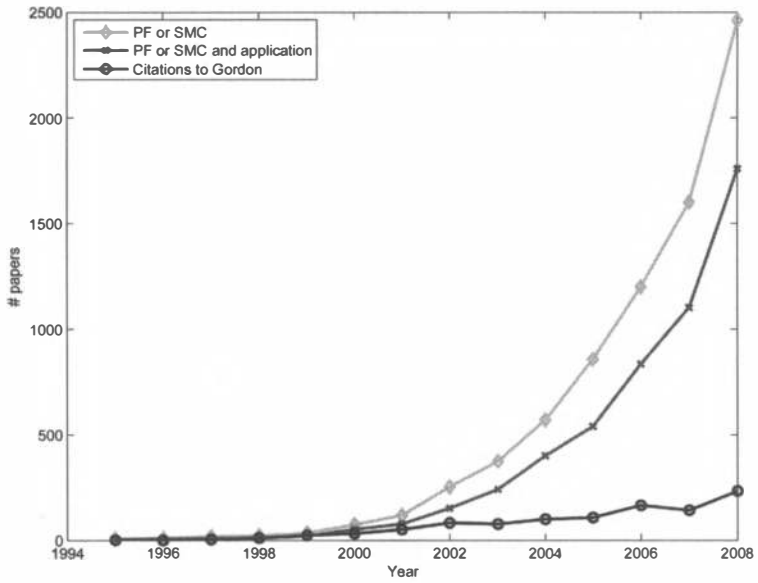

Fig. 1. Evolution over time of research on PFs. Graph shows number of papers in Thomson/ISI database that match search on "particle filter" OR "sequential Monte Carlo" (upper curve), "particle filter" OR "sequential Monte Carlo" AND "application" (middle curve), and number of citations of [15] (lower curve).

dimensions and that the algorithm itself is of quadratic complexity in the grid size.

It should be stressed that both EKF and UKF approximate the model and propagate Gaussian distributions representitive of the posterior while the PMF uses the original model and approximates the posterior over a grid. The particle filter (PF) also provides a numerical approximation to the nonlinear filtering problem similar to the PMF but uses an adaptive stochastic grid that automatically selects relevant grid points in the state space, and in contrast to the PMF, the standard PF has linear complexity in the number of grid points.

The first traces of the PF date back to the 1950s [11, 12], and the control community made some attempts in the 1970s [13, 14]. However, the PF era started with the seminal paper [15], and the independent developments in [16, 17]. Here, an important resampling step was introduced. The timing for proposing a general solution to the nonlinear filtering problem was perfect in that the computer development enabled the use of computationally complex algorithms in quite realistic problems. Since the paper [15] the research has steadily intensified; see the article collection [18], the surveys [19–22], and the monograph [23]. Fig. 1 illustrates how the number of papers increases exponentially each year, and the same appears to be true for applied papers. The PFs may be a serious alternative for real-time applications classically approached by the (E)KF. The more nonlinear model, or the more non-Gaussian noise, the more potential PFs have, especially in applications where computational power is rather cheap, and the sampling rate is moderate.

Positioning of moving platforms has been a technical driver for real-time applications of the PF in both the signal processing and the robotics communities. For this reason, we spend some time explaining several such applications in detail and summarizing the experiences of using the PF in practice. The applications concern positioning of underwater (UW) vessels, surface ships, cars, and aircraft using geographical information systems (GIS) containing a database with features of the surrounding landscape. These applications provide conclusions supporting the theoretical survey part.

In the robotics community, the PF has been developed into one of the main algorithms (fastSLAM) [24] for solving the simultaneous localization and mapping (SLAM) problem [25–27]. This can be seen as an extension of the aforementioned applications, where the features in the GIS are dynamically detected and updated on the fly. Visual tracking has turned out to be another important application for the PF. Multiple targets are here tracked from a video stream alone [28–30] or by fusion with other information, for instance, acoustic sensors [31].

The common denominator of these applications of the PF is the use of a low-dimensional state vector consisting of horizontal position and course (three-dimensional pose). The PF performs quite well in a three-dimensional state space. In higher dimensions the curse of dimensionality quite soon makes the particle representation too sparse to be a meaningful representation of the posterior distribution. That is, the PF is not practically useful when extending the models to more realistic cases

with

1) motion in three dimensions (six-dimensional pose),
2) more dynamic states (accelerations, unmeasured velocities, etc.),
3) or sensor biases and drifts.

A technical enabler for such applications is the marginalized PF (MPF), also referred to as the Rao-Blackwellized PF (RBPF). It allows for the use of high-dimensional state-space models as long as the (severe) nonlinearities only affect a small subset of the states. In this way the structure of the model is utilized so that the particle filter is used to solve the most difficult tasks, and the (E)KF is used for the (almost) linear Gaussian states. The fastSLAM algorithm is in fact a version of the MPF, where hundreds or thousands of feature points in the state vector are updated using the (E)KF. The need for the MPF in the list of applications will be motivated by examples and experience from practice.

This tutorial uses notation and terminology that should be familiar to the AES community, and it deliberately avoids excessive use of concepts from probability theory, where the main tools here are Bayes' theorem and the marginalization formula (or law of total probability). There are explicit comparisons and references to the KF, and the applications are in the area of target tracking and navigation. For instance, a particle represents a (target) state trajectory; the (target) motion dynamics and sensor observation model are assumed to be in state-space form, and the PF algorithm is split into time and measurement updates.

The PF should be the nonlinear filtering algorithm that appeals to engineers the most since it intimately addresses the system model. The filtering code is thus very similar to the simulation code that the engineer working with the application should already be quite familiar with. For that reason, one can have a code-first approach, starting with Section IX to get a complete simulation code for a concrete example. This section also provides some other examples using an object-oriented programming framework where models and signals are represented with objects, and can be used to quickly compare different filters, tunings, and models. Section X provides an overview of a number of applications of the PF, which can also be read stand-alone. Section XI extends the applications to models of high state dimensions where the MPF has been applied. The practical experiences are summarized in Section XII.

However, the natural structure is to start with an overview of the PF theory as found in Section II, and a summary of the MPF theory is provided in Section VIII, where the selection of topics is strongly influenced by the practical experiences in Section XII.

II. NONLINEAR FILTERING

A. Models and Notation

Applied nonlinear filtering is based on discrete time nonlinear state-space models relating a hidden state x_k to the observations y_k:

$$x_{k+1} = f(x_k, v_k), \qquad v_k \sim p_{v_k}, \quad x_1 \sim p_{x_1} \tag{1a}$$

$$y_k = h(x_k) + e_k, \qquad e_k \sim p_{e_k}. \tag{1b}$$

Here k denotes the sample number, v_k is a stochastic noise process specified by its known pdf p_{v_k}, which is compactly expressed as $v_k \sim p_{v_k}$. Similarly e_k is an additive measurement noise also with known pdf p_{e_k}. The first observation is denoted y_1, and thus the first unknown state is x_1 where the pdf of the initial state is denoted p_{x_1}. The model can also depend on a known (control) input u_k, so $f(x_k, u_k, v_k)$ and $h(x_k, u_k)$, but this dependence is omitted to simplify notation. The notation $s_{1:k}$ denotes the sequence $s_1, s_2, \ldots, s_k$ (s is one of the signals x, v, y, e), and n_s denotes the dimension of that signal.

In the statistical literature, a general Markov model and observation model in terms of conditional pdfs are often used

$$x_{k+1} \sim p(x_{k+1} \mid x_k) \tag{2a}$$

$$y_k \sim p(y_k \mid x_k). \tag{2b}$$

This is in a sense a more general model. For instance, (2) allows implicit measurement relations $h(y_k, x_k, e_k) = 0$ in (1b), and differential algebraic equations that add implicit state constraints to (1a).

The Bayesian approach to nonlinear filtering is to compute or approximate the posterior distribution for the state given the observations. The posterior is denoted $p(x_k \mid y_{1:k})$ for filtering, $p(x_{k+m} \mid y_{1:k})$ for prediction, and $p(x_{k-m} \mid y_{1:k})$ for smoothing where $m > 0$ denotes the prediction or smoothing lag. The theoretical derivations are based on the general model (2), while algorithms and discussions are based on (1). Note that the Markov property of the model (2) implies the formulas $p(x_{k+1} \mid x_{1:k}, y_{1:k}) = p(x_{k+1} \mid x_k)$ and $p(y_k \mid x_{1:k}, y_{1:k-1}) = p(y_k \mid x_k)$, which are used frequently.

A linearized model will turn up on several occasions and is obtained by a first-order Taylor expansion of (1) around $x_k = \bar{x}_k$ and $v_k = 0$:

$$x_{k+1} = f(\bar{x}_k, 0) + F(\bar{x}_k)(x_k - \bar{x}_k) + G(\bar{x}_k)v_k \tag{3a}$$

$$y_k = h(\bar{x}_k) + H(\bar{x}_k)(x_k - \bar{x}_k) + e_k \tag{3b}$$

where

$$F(\bar{x}_k) = \left.\frac{\partial f(x_k, v_k)}{\partial x_k}\right|_{x_k = \bar{x}_k, v_k = 0}$$

$$G(\bar{x}_k) = \left.\frac{\partial f(x_k, v_k)}{\partial v_k}\right|_{x_k = \bar{x}_k, v_k = 0} \tag{3c}$$

$$H(\bar{x}_k) = \left.\frac{\partial h(x_k)}{\partial x_k}\right|_{x_k = \bar{x}_k}$$

and the noise is represented by their second-order moments

$$\mathrm{cov}(e_k) = R_k, \qquad \mathrm{cov}(v_k) = Q_k, \qquad \mathrm{cov}(x_1) = P_0. \tag{3d}$$

For instance, the EKF recursions are obtained by linearizing around the previous estimate and applying the KF equations, which gives

$$K_k = P_{k|k-1}H^T(\hat{x}_{k|k-1}) \times (H(\hat{x}_{k|k-1})P_{k|k-1}H^T(\hat{x}_{k|k-1}) + R_k)^{-1} \tag{4a}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - h_k(\hat{x}_{k|k-1})) \tag{4b}$$

$$P_{k|k} = P_{k|k-1} - K_k H(\hat{x}_{k|k-1})P_{k|k-1} \tag{4c}$$

$$\hat{x}_{k+1|k} = f(\hat{x}_{k|k},0) \tag{4d}$$

$$P_{k+1|k} = F(\hat{x}_{k|k})P_{k|k}F^T(\hat{x}_{k|k}) + G(\hat{x}_{k|k})QG^T(\hat{x}_{k|k}). \tag{4e}$$

The recursion is initialized with $\hat{x}_{1|0} = x_0$ and $P_{1|0} = P_0$, assuming the prior $p(x_1) \sim \mathcal{N}(x_0, P_0)$. The EKF approximation of the posterior filtering distribution is then

$$\hat{p}(x_k \mid y_{1:k}) = \mathcal{N}(\hat{x}_{k|k}, P_{k|k}) \tag{5}$$

where $\mathcal{N}(m,P)$ denotes the Gaussian density function with mean m and covariance P. The special case of a linear model is covered by (3) in which case $F(\bar{x}_k) = F_k$, $G(\bar{x}_k) = G_k$, $H(\bar{x}_k) = H_k$; using these and the equalities $f(\bar{x}_k,0) = F_k\bar{x}_k$ and $h(\bar{x}_k) = H_k\bar{x}_k$ in (4) gives the standard KF recursion.

The neglected higher order terms in the Taylor expansion imply that the EKF can be biased and that it tends to underestimate the covariance of the state estimate. There is a variant of the EKF that also takes the second-order term of the Taylor expansion into account [32]. This is done by adding the expected value of the second-order term to the state updates and its covariance to the state covariance updates. The UKF [5, 6] does a similar correction by using propagation of systematically chosen state points (called sigma points) through the model. Related approaches include the DDF [7] that uses Sterling's formula to find the sigma points and the QKF [8] that uses the quadrature rule in numerical integration to select the sigma points. The common theme in EKF, UKF, DDF, and QKF is that the nonlinear model is evaluated in the current state estimate. The latter filters have some extra points in common that depend on the current state covariance.

UKF is closely related to the second-order EKF [33]. Both variants perform better than the EKF in certain problems and can work well as long as the posterior distribution is unimodal. The algorithms are prone to diverge, a problem that is hard to mitigate or foresee by analytical methods. The choice of state coordinates is therefore crucial in EKF and UKF (see [34, ch. 8.9.3] for one example) while this choice does not affect the performance of the PF (more than potential numerical problems).

B. Bayesian Filtering

The Bayesian solution to computing the posterior distribution $p(x_k \mid y_{1:k})$ of the state vector, given past observations, is given by the general Bayesian update recursion:

$$p(x_k \mid y_{1:k}) = \frac{p(y_k \mid x_k)p(x_k \mid y_{1:k-1})}{p(y_k \mid y_{1:k-1})} \tag{6a}$$

$$p(y_k \mid y_{1:k-1}) = \int_{\mathbb{R}^{n_x}} p(y_k \mid x_k)p(x_k \mid y_{1:k-1})dx_k \tag{6b}$$

$$p(x_{k+1} \mid y_{1:k}) = \int_{\mathbb{R}^{n_x}} p(x_{k+1} \mid x_k)p(x_k \mid y_{1:k})dx_k. \tag{6c}$$

This classical result [35, 36] is the cornerstone in nonlinear Bayesian filtering. The first equation follows directly from Bayes' law, and the other two follow from the law of total probability, using the model (2). The first equation corresponds to a measurement update, the second is a normalization constant, and the third corresponds to a time update.

The posterior distribution is the primary output from a nonlinear filter, from which standard measures as the minimum mean square (MMS) estimate $\hat{x}_k^{\mathrm{MMS}}$ and its covariance $P_{k|k}^{\mathrm{MMS}}$ can be extracted and compared with EKF and UKF outputs:

$$\hat{x}_{k|k}^{\mathrm{MMS}} = \int x_k p(x_k \mid y_{1:k})dx_k \tag{7a}$$

$$P_{k|k}^{\mathrm{MMS}} = \int (x_k - \hat{x}_k^{\mathrm{MMS}})(x_k - \hat{x}_k^{\mathrm{MMS}})^T p(x_k \mid y_{1:k})dx_k. \tag{7b}$$

For a linear Gaussian model, the KF recursions in (4) also provide the solution (7) to this Bayesian problem. However, for nonlinear or non-Gaussian models there is in general no finite-dimensional representation of the posterior distributions similar to $(\hat{x}_{k|k}^{\mathrm{MMS}}, P_{k|k}^{\mathrm{MMS}})$. That is why numerical approximations are needed.

C. The Point Mass Filter

Suppose now we have a deterministic grid $\{x^i\}_{i=1}^N$ of the state space $\mathbb{R}^{n_x}$ over N points, and that at time k, based on observations $y_{1:k-1}$, we have computed the relative probabilites (assuming distinct grid points)

$$w_{k|k-1}^i \propto \mathrm{P}(x_k = x^i \mid y_{1:k-1}) \tag{8}$$

satisfying $\sum_{i=1}^N w_{k|k-1}^i = 1$ (note that this is a relative normalization with respect to the grid points). The notation x_k^i is introduced here to unify notation with the PF, and it means that the state x_k at time k visits

the grid point x^i. The prediction density and the first two moments can then be approximated by

$$\hat{p}(x_k \mid y_{1:k-1}) = \sum_{i=1}^{N} w_{k|k-1}^i \delta(x_k - x_k^i) \tag{9a}$$

$$\hat{x}_{k|k-1} = \mathrm{E}(x_k) = \sum_{i=1}^{N} w_{k|k-1}^i x_k^i \tag{9b}$$

$$P_{k|k-1} = \mathrm{cov}(x_k) = \sum_{i=1}^{N} w_{k|k-1}^i (x_k^i - \hat{x}_{k|k-1})(x_k^i - \hat{x}_{k|k-1})^T. \tag{9c}$$

Here, $\delta(x)$ denotes the Dirac impulse function. The Bayesian recursion (6) now gives

$$\hat{p}(x_k \mid y_{1:k}) = \sum_{i=1}^{N} \underbrace{\frac{1}{c_k} p(y_k \mid x_k^i) w_{k|k-1}^i}_{w_{k|k}^i} \delta(x_k - x_k^i) \tag{10a}$$

$$c_k = \sum_{i=1}^{N} p(y_k \mid x_k^i) w_{k|k-1}^i \tag{10b}$$

$$\hat{p}(x_{k+1} \mid y_{1:k}) = \sum_{i=1}^{N} w_{k|k}^i p(x_{k+1} \mid x_k^i). \tag{10c}$$

Note that the recursion starts with a discrete approximation (9a) and ends in a continuous distribution (10c). Now, to close the recursion, the standard approach is to sample (10c) at the grid points x^i, which computationally can be seen as a multidimensional convolution,

$$w_{k+1|k}^i = p(x_{k+1}^i \mid y_{1:k}) = \sum_{j=1}^{N} w_{k|k}^j p(x_{k+1}^i \mid x_k^j), \quad i = 1,2,\ldots,N. \tag{11}$$

This is the principle of the PMF [9, 10], whose advantage is its simple implementation and tuning (the engineer basically only has to consider the size and resolution of the grid). The curse of dimensionality limits the application of PMF to small models (n_x less than two or three) for two reasons: the first one is that a grid is an inefficiently sparse representation in higher dimensions, and the second one is that the multidimensional convolution becomes a real bottleneck with quadratic complexity in N. Another practically important but difficult problem is to translate and change the resolution of the grid adaptively.

III. THE PARTICLE FILTER

A. Relation to the Point Mass Filter

The PF has much in common with the PMF. Both algorithms approximate the posterior distribution with a discrete density of the form (9a), and they are both based on a direct application of (6) leading to the numerical recursion in (10). However, there are some major differences:

1) The deterministic grid x^i in the PMF is replaced with a dynamic stochastic grid x_k^i in the PF that changes over time. The stochastic grid is a much more efficient representation of the state space than a fixed or adaptive deterministic grid in most cases.

2) The PF aims at estimating the whole trajectory $x_{1:k}$ rather than the current state x_k. That is, the PF generates and evaluates a set $\{x_{1:k}^i\}_{i=1}^N$ of N different trajectories. This affects (6c) as follows:

$$p(x_{1:k+1}^i \mid y_{1:k}) = \underbrace{p(x_{k+1}^i \mid x_{1:k}^i, y_{1:k})}_{p(x_{k+1}^i \mid x_k^i)} \underbrace{p(x_{1:k}^i \mid y_{1:k})}_{w_{k|k}^i} \tag{12}$$

$$= w_{k|k}^i p(x_{k+1}^i \mid x_k^i). \tag{13}$$

Comparing this to (10c) and (11), we note that the double sum leading to a quadratic complexity is avoided by this trick. However, this quadratic complexity appears if one wants to recover the marginal distribution $p(x_k \mid y_{1:k})$ from $p(x_{1:k} \mid y_{1:k})$, more on this in Section IIIC.

3) The new grid in the PF is obtained by sampling from (10c) rather than reusing the old grid as done in the PMF. The original version of the PF [15] samples from (10c) as it stands by drawing one sample each from $p(x_{k+1} \mid x_k^i)$ for $i = 1,2,\ldots,N$. More generally, the concept of importance sampling [37] can be used. The idea is to introduce a proposal density $q(x_{k+1} \mid x_k, y_{k+1})$, which is easy to sample from, and rewrite (6c) as

$$p(x_{k+1} \mid y_{1:k}) = \int_{\mathbb{R}^{n_x}} p(x_{k+1} \mid x_k) p(x_k \mid y_{1:k}) dx_k = \int_{\mathbb{R}^{n_x}} q(x_{k+1} \mid x_k, y_{k+1}) \frac{p(x_{k+1} \mid x_k)}{q(x_{k+1} \mid x_k, y_{k+1})} \times p(x_k \mid y_{1:k}) dx_k. \tag{14}$$

The trick now is to generate a sample at random from $x_{k+1}^i \sim q(x_{k+1} \mid x_k^i, y_{k+1})$ for each particle, and then adjust the posterior probability for each particle with the importance weight

$$p(x_{1:k+1} \mid y_{1:k}) = \sum_{i=1}^{N} \underbrace{\frac{p(x_{k+1}^i \mid x_k^i)}{q(x_{k+1}^i \mid x_k^i, y_{k+1})} w_{k|k}^i}_{w_{k+1|k}^i} \delta(x_{1:k+1} - x_{1:k+1}^i). \tag{15}$$

As indicated, the proposal distribution $q(x_{k+1}^i \mid x_k^i, y_{k+1})$ depends on the last state in the particle trajectory $x_{1:k}^i$, but also the next measurement y_{k+1}. The simplest choice of proposal is to use the dynamic model itself $q(x_{k+1}^i \mid x_k^i, y_{k+1}) = p(x_{k+1}^i \mid x_k^i)$ leading to $w_{k+1|k}^i = w_{k|k}^i$. The choice of proposal and its actual form are discussed more thoroughly in Section V.

4) Resampling is a crucial step in the PF. Without resampling, the PF would break down to a set of independent simulations yielding trajectories $x^i_{1:k}$ with relative probabilities w^i_k. Since there would then be no feedback mechanism from the observations to control the simulations, they would quite soon diverge. As a result, all relative weights would tend to zero except for one that tends to one. This is called sample depletion, sample degeneracy, or sample impoverishment. Note that a relative weight of one $w^i_{k|k} \approx 1$ is not at all an indicator of how close a trajectory is to the true trajectory since this is only a relative weight. It merely says that one sequence in the set $\{x^i_{1:k}\}^N_{i=1}$ is much more likely than all of the other ones. Resampling introduces the required information feedback from the observations, so trajectories that perform well will survive the resampling. There are some degrees of freedom in the choice of resampling strategy discussed in Section IVA.

B. Algorithm

The PF algorithm is summarized in Algorithm 1. It can be seen as an algorithmic framework from which particular versions of the PF can be defined later on. It should be noted that the most common form of the algorithm combines the weight updates (16a, d) into one equation. Here, we want to stress the relations to the fundamental Bayesian recursion by keeping the structure of a measurement update (6a)–(10a)–(16a), normalization (6b)–(10b)–(16b), and time update (6c)–(10c)–(16c, d).

ALGORITHM 1 Particle Filter *Choose a proposal distribution* $q(x_{k+1} \mid x_{1:k}, y_{k+1})$, *resampling strategy, and the number of particles* N.

Initialization: Generate $x^i_1 \sim p_{x_0}$, $i = 1,\ldots,N$ *and let* $w^i_{1|0} = 1/N$.

Iteration For $k = 1,2,\ldots$.

1) *Measurement update: For* $i = 1,2,\ldots,N$,

$$w^i_{k|k} = \frac{1}{c_k} w^i_{k|k-1} p(y_k \mid x^i_k) \tag{16a}$$

where the normalization weight is given by

$$c_k = \sum_{i=1}^{N} w^i_{k|k-1} p(y_k \mid x^i_k). \tag{16b}$$

2) *Estimation: The filtering density is approximated by* $\hat{p}(x_{1:k} \mid y_{1:k}) = \sum_{i=1}^N w^i_{k|k} \delta(x_{1:k} - x^i_{1:k})$ *and the mean* (7a) *is approximated by* $\hat{x}_{1:k} \approx \sum_{i=1}^N w^i_{k|k} x^i_{1:k}$.

3) *Resampling: Optionally at each time, take* N *samples with replacement from the set* $\{x^i_{1:k}\}^N_{i=1}$ *where the probability to take sample* i *is* $w^i_{k|k}$ *and let* $w^i_{k|k} = 1/N$.

4) *Time update: Generate predictions according to the proposal distribution*

$$x^i_{k+1} \sim q(x_{k+1} \mid x^i_k, y_{k+1}) \tag{16c}$$

and compensate for the importance weight

$$w^i_{k+1|k} = w^i_{k|k} \frac{p(x^i_{k+1} \mid x^i_k)}{q(x^i_{k+1} \mid x^i_k, y_{k+1})}. \tag{16d}$$

C. Prediction, Smoothing, and Marginals

Algorithm 1 outputs an approximation of the trajectory posterior density $p(x_{1:k} \mid y_{1:k})$. For a filtering problem, the simplest engineering solution is to just extract the last state x^i_k from the trajectory $x^i_{1:k}$ and use the particle approximation

$$\hat{p}(x_k \mid y_{1:k}) = \sum_{i=1}^{N} w^i_{k|k} \delta(x_k - x^i_k). \tag{17}$$

Technically this is incorrect, and one may overlook the depletion problem by using this approximation. The problem is that in general all paths $x^j_{1:k-1}$ can lead to the state x^i_k. Note that the marginal distribution is functionally of the same form as (6c). The correct solution taking into account all paths leading to x^i_k leads (similar to (11)) to an importance weight

$$w^i_{k+1|k} = \frac{\sum_{j=1}^N w^j_{k|k} p(x^i_{k+1} \mid x^j_k)}{q(x^i_{k+1} \mid x^i_k, y_{k+1})} \tag{18}$$

that replaces the one in (16d). That is, the marginal PF can be implemented just like Algorithm 1 by replacing the time update of the weights with (18). Note that the complexity increases from $\mathcal{O}(N)$ in the PF to $\mathcal{O}(N^2)$ in the marginal PF, due to the new importance weight. A method with $\mathcal{O}(N \log(N))$ complexity is suggested in [38].

The marginal PF has found very interesting applications in system identification, where a gradient search for unknown parameters in the model is applied [39, 40]. The same parametric approach has been suggested for SLAM in [41] and optimal trajectory planning in [42].

Though the PF appears to solve the smoothing problem for free, the inherent depletion problem of the history complicates the task, since the number of surviving trajectories with a time lag will quickly be depleted. For fixed-lag smoothing $p(x_{k-m:k} \mid y_{1:k})$, one can compute the same kind of marginal distributions as for the marginal PF leading to another compensation factor of the importance weight. However, the complexity will then be $\mathcal{O}(N^{m+1})$. Similar to the KF smoothing problem, the suggested solution [43] is based on first running the PF in the usual way and then applying a backward sweep of a modified PF.

The prediction to get $p(x_{1:k+m} \mid y_{1:k})$ can be implemented by repeating the time update in Algorithm 1 m times.

D. Reading Advice

The reader may at this stage continue to Section IX to see MATLAB™ code for some illustrative examples, or to Section X to read about the results and experiences using some other applications, or proceed to the subsequent sections that discuss the following issues:

1) The tuning possibilities and different versions of the basic PF are discussed in Section IV.
2) The choice of proposal distribution is crucial for performance, just as in any classical sampling algorithm [37], and this is discussed in Section V.
3) Performance in terms of convergence of the approximation $\hat{p}(x_{1:k} \mid y_{1:k}) \to p(x_{1:k} \mid y_{1:k})$ as $N \to \infty$ and relation to fundamental performance bounds are discussed in Section VI.
4) The PF is computationally quite complex, and some potential bottlenecks and possible remedies are discussed in Section VII.

IV. TUNING

The number of particles N is the most immediate design parameter in the PF. There are a few other degrees of freedom discussed below. The overall goal is to avoid sample depletion, which means that only a few particles, or even only one, contribute to the state estimate. The choice of proposal distribution is the most intricate one, and it is discussed separately in Section V. How the resampling strategy affects sample depletion is discussed in Section IVA. The effective number of samples in Section IVB is an indicator of sample depletion in that it measures how efficiently the PF is utilizing its particles. It can be used to design proposal distributions, depletion mitigation tricks, and resampling algorithms and also to choose the number of particles. It can also be used as an online control variable for when to resample. Some dedicated tricks are discussed in Section IVC.

A. Resampling

Without the resampling step, the basic PF would suffer from sample depletion. This means that after a while all particles but a few will have negligible weights. Resampling solves this problem but creates another in that resampling inevitably destroys information and thus increases uncertainty in the random sampling. It is therefore of interest to start the resampling process only when it is really needed. The following options for when to resample are possible.

1) The standard version of Algorithm 1 is termed sampling importance resampling (SIR), or bootstrap PF, and is obtained by resampling each time.
2) The alternative is to use importance sampling, in which case resampling is performed only when needed. This is called sampling importance sampling (SIS). Usually, resampling is done when the effective number of samples, as will be defined in the next section, becomes too small.

As an alternative, the resampling step can be replaced with a sampling step from a distribution that is fitted to the particles after both the time and measurement update. The Gaussian PF (GPF) in [44] fits a Gaussian distribution to the particle cloud after which a new set of particles is generated from this distribution. The Gaussian sum PF (GSPF) in [45] uses a Gaussian sum instead of a distribution.

B. Effective Number of Samples

An indicator of the degree of depletion is the effective number of samples,[1] defined in terms of the coefficient of variation c_v [19, 46, 47] as

$$N_{\text{eff}} = \frac{N}{1 + c_v^2(w_{k|k}^i)} = \frac{N}{1 + \dfrac{\text{Var}(w_{k|k}^i)}{(\text{E}(w_{k|k}^i))^2}} = \frac{N}{1 + N^2\text{Var}(w_{k|k}^i)}. \tag{19a}$$

The effective number of samples is thus at its maximum $N_{\text{eff}} = N$ when all weights are equal $w_{k|k}^i = 1/N$, and the lowest value it can attain is $N_{\text{eff}} = 1$, which occurs when $w_{k|k}^i = 1$ with probability $1/N$ and $w_{k|k}^i = 0$ with probability $(N-1)/N$.

A logical computable approximation of N_{eff} is provided by

$$\hat{N}_{\text{eff}} = \frac{1}{\sum_i (w_{k|k}^i)^2}. \tag{19b}$$

This approximation shares the property $1 \le \hat{N}_{\text{eff}} \le N$ with the definition (19a). The upper bound $\hat{N}_{\text{eff}} = N$ is attained when all particles have the same weight and the lower bound $\hat{N}_{\text{eff}} = 1$ when all the probability mass is devoted to a single particle.

The resampling condition in the PF can now be defined as $\hat{N}_{\text{eff}} < N_{\text{th}}$. The threshold can for instance be chosen as $\hat{N}_{\text{th}} = 2N/3$.

C. Tricks to Mitigate Sample Depletion

The choice of proposal distribution and resampling strategy are the two available instruments to avoid sample depletion problems. There are also some simple and more practical ad hoc tricks that can be tried as discussed below.

[1] Note that the literature often defines the effective number of samples as $N/(1 + \text{Var}(w_{k|k}^i))$, which is incorrect.

One important trick is to modify the noise models so the state noise and/or the measurement noise appear larger in the filter than they really are in the data generating process. This technique is called "jittering" in [48], and a similar approach was introduced in [15] under the name "roughening." Increasing the noise level in the state model (1a) increases the support of the sampled particles, which partly mitigates the depletion problem. Further, increasing the noise level in the observation model (1b) implies that the likelihood decays slower for particles that do not fit the observation, and the chance to resample these increases. In [49], the depletion problem is handled by introducing an additional Markov Chain Monte Carlo (MCMC) step to separate the samples.

In [15], the so-called prior editing method is discussed. The estimation problem is delayed one time step so that the likelihood can be evaluated at the next time step. The idea is to reject particles with sufficiently small likelihood values, since they are not likely to be resampled. The update step is repeated until a feasible likelihood value is received. The roughening method could also be applied before the update step is invoked. The auxiliary PF [50] is a more formal way to sample such that only particles associated with large predictive likelihoods are considered; see Section VF.

Another technique is regularization. The basic idea to is convolve each particle with a diffusion kernel with a certain bandwidth before resampling. This will prevent multiple copies of a few particles. One may for instance use a Gaussian kernel where the variance acts as the bandwidth. One problem in theory with this approach is that this kernel will increase the variance of the posterior distribution.

V. CHOICE OF PROPOSAL DISTRIBUTION

In this section we focus on the choice of proposal distribution, which influences the depletion problem significantly, and we outline available options with some comments on when they are suitable.

First note that the most general proposal distribution has the form $q(x_{1:k} \mid y_{1:k})$. This means that the whole trajectory should be sampled at each iteration, which is clearly not attractive in real-time applications. Now, the general proposal can be factorized as

$$q(x_{1:k} \mid y_{1:k}) = q(x_k \mid x_{1:k-1}, y_{1:k}) q(x_{1:k-1} \mid y_{1:k}). \tag{20}$$

The most common approximation in applications is to reuse the path $x_{1:k-1}$ and only sample the new state x_k, so the proposal $q(x_{1:k} \mid y_{1:k})$ is replaced by $q(x_k \mid x_{1:k-1}, y_{1:k})$. The approximate proposal suggests good values of x_k only, not of the trajectory $x_{1:k}$. For filtering problems this is not an issue, but for smoothing problems the second factor becomes important. Here, the idea of block sampling [51] is quite interesting.

Now, due to the Markov property of the model, the proposal $q(x_k \mid x_{1:k-1}, y_{1:k})$ can be written as

$$q(x_k \mid x_{1:k-1}, y_{1:k}) = q(x_k \mid x_{k-1}, y_k). \tag{21}$$

The following sections discuss various approximations of this proposal and in particular how the choice of proposal depends on the signal-to-noise ratio (SNR). For linear Gaussian models, the SNR is in loose terms defined as $\|Q\|/\|R\|$; that is, the SNR is high if the measurement noise is small compared with the signal noise. Here, we define SNR as the ratio of the maximal value of the likelihood to the prior,

$$\text{SNR} \propto \frac{\max_{x_k} p(y_k \mid x_k)}{\max_{x_k} p(x_k \mid x_{k-1})}. \tag{22}$$

For a linear Gaussian model, this gives $\text{SNR} \propto \sqrt{\det(Q)/\det(R)}$.

In this section we use the weight update

$$w^i_{k|k} \propto w^i_{k-1|k-1} \frac{p(y_k \mid x^i_k) p(x^i_k \mid x^i_{k-1})}{q(x_k \mid x^i_{k-1}, y_k)} \tag{23}$$

combining (16a) and (16b). The SNR thus indicates which factor in the numerator most likely to change the weights the most.

Besides the options below that all relate to (21), there are many more ad hoc–based options described in the literature.

A. Optimal Sampling

The conditional distribution includes all information from the previous state and the current observation and should thus be the best proposal to sample from. This conditional pdf can be written as

$$q(x_k \mid x^i_{k-1}, y_k) = p(x_k \mid x^i_{k-1}, y_k) = \frac{p(y_k \mid x_k) p(x_k \mid x^i_{k-1})}{p(y_k \mid x^i_{k-1})}. \tag{24a}$$

This choice gives the proposal weight update

$$w^i_{k|k} \propto w^i_{k-1|k-1} p(y_k \mid x^i_{k-1}). \tag{24b}$$

The point is that the weight will be the same whatever sample of x^i_k is generated. Put in another way, the variance of the weights is unaffected by the sampling. All other alternatives will add variance to the weights and thus decrease the effective number of samples according to (19a). In the interpretation of keeping the effective number of samples as large as possible, (24a) is the optimal sampling.

The drawbacks are as follows:

1) It is generally hard to sample from this proposal distribution.

2) It is generally hard to compute the weight update needed for this proposal distribution, since it would require integrating over the whole state space, $p(y_k \mid x^i_{k-1}) = \int p(y_k \mid x_k) p(x_k \mid x^i_{k-1}) dx_k$.

One important special case when these steps actually become explicit is a linear and Gaussian measurement relation, which is the subject of Section VE.

B. Prior Sampling

The standard choice in Algorithm 1 is to use the conditional prior of the state vector as proposal distribution

$$q(x_k \mid x^i_{k-1}, y_k) = p(x_k \mid x^i_{k-1}) \tag{25a}$$

where $p(x_k \mid x^i_{k-1})$ is referred to as the prior of x_k for each trajectory. This yields

$$w^i_{k|k} = w^i_{k|k-1} p(y_k \mid x^i_k) = w^i_{k-1|k-1} p(y_k \mid x^i_k). \tag{25b}$$

This leads to the most common by far version of the PF (SIR) that was originally proposed in [15]. It performs well when the SNR is small, which means that the state prediction provides more information about the next state value than the likelihood function. For medium or high SNR, it is more natural to sample from the likelihood.

C. Likelihood Sampling

Consider first the factorization

$$p(x_k \mid x^i_{k-1}, y_k) = p(y_k \mid x^i_{k-1}, x_k) \frac{p(x_k \mid x^i_{k-1})}{p(y_k \mid x^i_{k-1})} = p(y_k \mid x_k) \frac{p(x_k \mid x^i_{k-1})}{p(y_k \mid x^i_{k-1})}. \tag{26a}$$

If the likelihood $p(y_k \mid x_k)$ is much more peaky than the prior and if it is integrable in x_k [52], then

$$p(x_k \mid x^i_{k-1}, y_k) \propto p(y_k \mid x_k). \tag{26b}$$

That is, a suitable proposal for the high SNR case is based on a scaled likelihood function

$$q(x_k \mid x^i_{k-1}, y_k) \propto p(y_k \mid x_k) \tag{26c}$$

which yields

$$w^i_{k|k} = w^i_{k-1|k-1} p(x^i_k \mid x^i_{k-1}). \tag{26d}$$

Sampling from the likelihood requires that the likelihood function $p(y_k \mid x_k)$ is integrable with respect to x_k [52]. This is not the case when $n_x > n_y$. The interpretation in this case is that for each value of y_k, there is a infinite-dimensional manifold of possible x_k to sample from, each one equally likely.

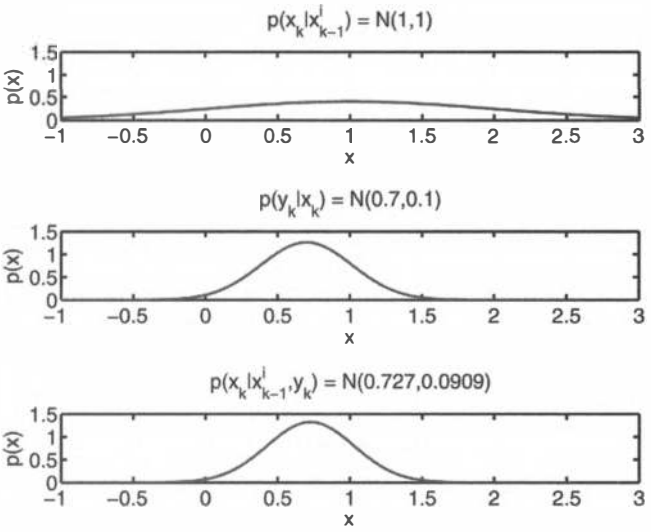

Fig. 2. Illustration of (24a) for scalar state and observation model. State dynamics moves particle to $x_k = 1$ and adds uncertainty with variance 1, after which observation $y_k = 0.7 = x_k + e_k$ is taken. Posterior in this high SNR example is here essentially equal to likelihood.

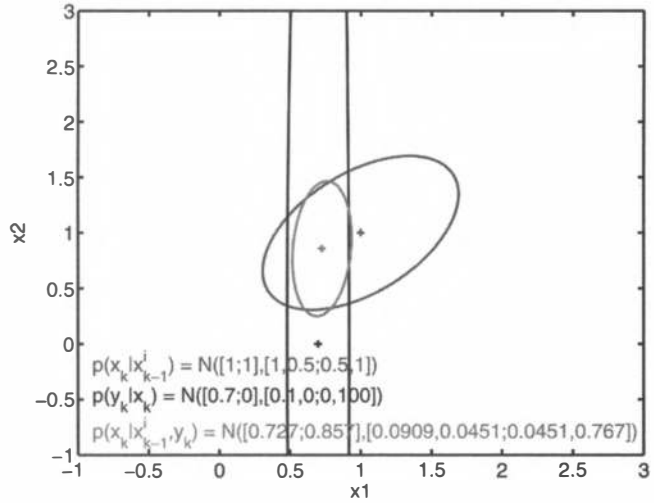

Fig. 3. Illustration of (24a) for two-dimensional state and scalar observation model. State dynamics moves particle to $x_k = (1,1)^T$ and adds correlated noise, after which an observation $y_k = 0.7 = (1,0)x_k + e_k$ is taken. Posterior in this high SNR example corresponds roughly to likelihood in one dimension (x_1) and prior in the other dimension (x_2).

D. Illustrations

A simple linear Gaussian model is used to illustrate the choice of proposal as a function of SNR. Fig. 2 illustrates a high SNR case for a scalar model, where the information in the prior is negligible compared with the peaky likelihood. This means that the optimal proposal essentially becomes a scaled version of the likelihood.

Fig. 3 illustrates a high SNR case for a two-dimensional state, where the observation dimension is smaller than the state space. The optimal

proposal can here be interpreted as the intersection of the prior and likelihood.

E. Optimal Sampling with Linearized Likelihood

The principles illustrated in Figs. 2 and 3 can be used for a linearized model [43], similar to the measurement update in the EKF (4ef). To simplify the notation somewhat, the process noise in (1a) is assumed additive $x_{k+1} = f(x_k) + v_k$. Assuming that the measurement relation (1b) is linearized as (3b) when evaluating (24a), the optimal proposal can be approximated with

$$q(x_k \mid x_{k-1}^i, y_k) = \mathcal{N}(f(x_{k-1}^i) + K_k^i(y_k - \hat{y}_k^i), (H_k^{i,T} R_k^\dagger H^i + Q_{k-1}^\dagger)^\dagger) \quad (27a)$$

where † denotes pseudoinverse. The Kalman gain, linearized measurement model, and measurement prediction, respectively, are given by

$$K_k^i = Q_{k-1} H_k^{i,T} (H_k^i Q_{k-1} H_k^{i,T} + R_k)^{-1} \quad (27b)$$

$$H_k^i = \left.\frac{\partial h(x_k)}{\partial x_k}\right|_{x_k = f(x_{k-1}^i)} \quad (27c)$$

$$\hat{y}_k^i = h(f(x_{k-1}^i)). \quad (27d)$$

The weights should thus be multiplied by the following likelihood in the measurement update:

$$p(y_k \mid x_{k-1}^i) = \mathcal{N}(y_k - \hat{y}_k^i, H_k^i Q_{k-1} H_k^{i,T} + R_k). \quad (27e)$$

The modifications of (27) can be motivated intuitively as follows. At time $k-1$, each particle corresponds to a state estimate with no uncertainty. The EKF recursions (4) using this initial value gives

$$\hat{x}_{k-1|k-1} \sim \mathcal{N}(x_{k-1}^i, 0) \Rightarrow \quad (28a)$$

$$\hat{x}_{k|k-1} = f(x_{k-1}^i) \quad (28b)$$

$$P_{k|k-1} = Q_{k-1} \quad (28c)$$

$$K_k = Q_{k-1} H_k^T (H_k Q_{k-1} H_k^T + R_k)^{-1} \quad (28d)$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - h(\hat{x}_{k|k-1})) \quad (28e)$$

$$P_{k|k} = Q_{k-1} - K_k H_k Q_{k-1}. \quad (28f)$$

We denote this sampling strategy OPT-EKF. To compare it with standard SIR algorithm, one can interpret the difference in terms of the time update. The modification in Algorithm 1 assuming a Gaussian distribution for both process and measurement noise, is to make the following substitution in the time update

$$x_{k+1}^i = f(x_k^i) + v_k^i \quad (29a)$$

$$\text{SIR}: \quad v_k^i \sim \mathcal{N}(0, Q_k) \quad (29b)$$

$$\text{OPT-EKF}: \quad v_k^i \in \mathcal{N}(K_{k+1}^i(y_{k+1} - h(f(x_k^i))), (H_{k+1}^{i,T} R_{k+1}^\dagger H_{k+1}^i + Q_k^\dagger)^\dagger). \quad (29c)$$

and measurement update

$$\text{SIR}: \quad w_{k|k}^i = w_{k-1|k-1}^i \mathcal{N}(y_k - h(x_k^i), R_k) \quad (29d)$$

$$\text{OPT-EKF}: \quad w_{k|k}^i = w_{k-1|k-1}^i \mathcal{N}(y_k - h(f(x_{k-1}^i)), H_k^i Q_{k-1} H_k^{i,T} + R_k) \quad (29e)$$

respectively. For OPT-SIR, the SNR definition can be more precisely stated as

$$\text{SNR} \propto \frac{\|H_k^i Q_{k-1} H_k^{i,T}\|}{\|R_k\|}. \quad (30)$$

We make the following observations and interpretations on some limiting cases of these algebraic expressons:

1) For small SNR, $K_k^i \approx 0$ in (27b) and $(H_k^{i,T} R_k^\dagger H_k^i + Q_{k-1}^\dagger)^\dagger \approx Q_{k-1}$ in (29c), which shows that the resampling (29c) in OPT-EKF proposal approaches (29b) in SIR as the SNR goes to zero. That is, for low SNR the approximation approaches prior sampling in Section VB.

2) Conversely, for large SNR and assuming H_k^i invertible (implicitly implying $n_y \geq n_x$), then $(H_k^{i,T} R_k^\dagger H_k^i + Q_{k-1}^\dagger)^\dagger \approx H_k^{i,-1} R_k H_k^{i,-T}$ in (29c). Here, all information about the state is taken from the measurement, and the model is not used; that is, for high SNR the approximation approaches likelihood sampling in Section VC.

3) The pseudoinverse † is used consequently in the notation for the proposal covariance $(H_k^{i,T} R_k^\dagger H_k^i + Q_{k-1}^\dagger)^\dagger$ instead of inverse to accomodate the following cases:

a) singular process noise Q_{k-1}, which is the case in most dynamic models including integrated noise,

b) singular measurement noise R_k, to allow ficticious measurements that model state constraints. For instance, a known state constraint corresponds to infinite information in a subspace of the state space, and the corresponding eigenvector of the measurement information $H_k^i R_k^\dagger H_k^{i,T}$ will overwrite the prior information $Q_{k-1}^\dagger$.

F. Auxiliary Sampling

The auxiliary sampling proposal resampling filter [50] uses an auxiliary index in the proposal distribution $q(x_k, i \mid y_{1:k})$. This leads to an algorithm

that first generates a large number M (typically $M = 10N$) of pairs $\{x_k^j, i^j\}_{j=1}^M$. From Bayes' rule, we have

$$p(x_k, i \mid y_{1:k}) \sim p(y_k \mid x_k) p(x_k, i \mid y_{1:k-1}) \tag{31a}$$

$$= p(y_k \mid x_k) p(x_k \mid i, y_{1:k-1}) p(i \mid y_{1:k-1}) \tag{31b}$$

$$= p(y_k \mid x_k) p(x_k \mid x_{k-1}^i) w_{k-1|k-1}^i. \tag{31c}$$

This density is implicit in x_k and thus not useful as an proposal density, since it requires x_k to be known. The general idea is to find an approximation of $p(y_k \mid x_{k-1}^i) = \int p(y_k \mid x_k) p(x_k \mid x_{k-1}^i) dx_k$. A simple though useful approximation is to replace x_k with its estimate and thus let $p(y_k \mid x_{k-1}^i) = p(y_k \mid \hat{x}_k^i)$ above. This leads to the proposal

$$q(x_k, i \mid y_{1:k}) = p(y_k \mid \hat{x}_k^i) p(x_k \mid x_{k-1}^i) w_{k-1|k-1}^i. \tag{31d}$$

Here, $\hat{x}_k^i = \mathrm{E}(x_k \mid x_{k-1}^i)$ can be the conditional mean or $\hat{x}_k^i \sim p(x_k \mid x_{k-1}^i)$ a sample from the prior. The new samples are drawn from the marginalized density

$$x_k^j \sim p(x_k \mid y_{1:k}) = \sum_i p(x_k, i \mid y_{1:k}). \tag{31e}$$

To evalute the proposal weight, we first take Bayes rule which gives

$$q(x_k, i \mid y_{1:k}) = q(i \mid y_{1:k}) q(x_k \mid i, y_{1:k}). \tag{31f}$$

Here, another choice must be made. The latter proposal factor should be defined as

$$q(x_k \mid i, y_{1:k}) = p(x_k \mid x_{k-1}^i). \tag{31g}$$

Then, this factor cancels out when forming

$$q(i \mid y_{1:k}) \propto p(y_k \mid \hat{x}_k^i) w_{k-1|k-1}^i. \tag{31h}$$

The new weights are thus given by

$$w_{k|k}^i = w_{k-1|k-1}^{ij} \frac{p(y_k \mid x_k^j) p(x_k^j \mid x_{k-1}^{ij})}{q(x_k^j, i^j \mid y_{1:k})}. \tag{31i}$$

Note that this proposal distribution is a product of the prior and the likelihood. The likelihood has the ability to punish samples x_k^i that give a poor match to the most current observation, unlike SIR and SIS where such samples are drawn and then immediately rejected. There is a link between the auxiliary PF and the standard SIR as pointed out in [53], which is useful for understanding its theoretical properties.

VI. THEORETICAL PERFORMANCE

The key questions here are how well the PF filtering density $\hat{p}(x_{1:k} \mid y_{1:k})$ approximates the true posterior $p(x_{1:k} \mid y_{1:k})$, and what the fundamental mean square error (MSE) bounds for the true posterior are.

A. Convergence Issues

The convergence properties of the PF are well understood on a theoretical level, see the survey [54] and the book [55]. The key question is how well a function $g(x_k)$ of the state can be approximated $\hat{g}(x_k)$ by the PF compared with the conditional expectation $\mathrm{E}(g(x_k))$, where

$$\mathrm{E}(g(x_k)) = \int g(x_k) p(x_{1:k} \mid y_{1:k}) dx_{1:k} \tag{32}$$

$$\hat{g}(x_k) = \int g(x_k) \hat{p}(x_{1:k} \mid y_{1:k}) dx_{1:k} = \sum_{i=1}^N w_{k|k}^i g(x_k^i). \tag{33}$$

In short, the following key results exist.

1) Almost sure weak convergence

$$\lim_{N \to \infty} \hat{p}(x_{1:k} \mid y_{1:k}) = p(x_{1:k} \mid y_{1:k}) \tag{34}$$

in the sense that $\lim_{N \to \infty} \hat{g}(x_k) = \mathrm{E}(g(x_k))$.

2) MSE asymptotic convergence

$$\mathrm{E}(\hat{g}(x_k) - \mathrm{E}(g(x_k)))^2 \leq \frac{p_k \|g(x_k)\|_{\sup}}{N} \tag{35}$$

where the supremum norm of $g(x_k)$ is used. As shown in [55] using the Feynman-Kac formula, under certain regularity and mixing conditions, the constant $p_k = p < \infty$ does not increase in time. The main condition [54, 55] for this result is that the unnormalized weight function is bounded. Further, most convergence results as surveyed in [56] are restricted to bounded functions of the state $g(x)$ such that $|g(x)| < C$ for some C. The convergence result presented in [57] extends this to unbounded functions, for instance estimation of the state itself $g(x) = x$, where the proof requires the additional assumption that the likelihood function is bounded from below by a constant.

In general, the constant p_k grows polynomially in time, but does not necessarily depend on the dimension of the state space, at least not explicitly. That is, in theory we can expect the same good performance for high-order state vectors. In practice, the performance degrades quickly with the state dimension due to the curse of dimensionality. However, it scales much better with state dimension than the PMF, which is one of the key reasons for the success of the PF.

B. Nonlinear Filtering Performance Bound

Besides the performance bound of a specific algorithm as discussed in the previous section, there are more fundamental estimation bounds for nonlinear filtering that depend only on the model and not on the applied algorithm. The Cramér-Rao Lower Bound (CRLB) $P_{k|k}$ provides such a performance bound for

any unbiased estimator $\hat{x}_{k|k}$,

$$\text{cov}(\hat{x}_{k|k}) \geq P_{k|k}^{\text{CRLB}}. \tag{36}$$

The most useful version of CRLB is computed recursively by a Riccati equation which has the same functional form as the KF in (4) evaluated at the true trajectory $x_{1:k}^o$,

$$P_{k|k}^{\text{CRLB}} = P_{k|k-1}^{\text{CRLB}} - P_{k|k-1}^{\text{CRLB}} H(x_k^o)^T \times (H(x_k^o) P_{k|k-1}^{\text{CRLB}} H^T(x_k^o) + R_k)^{-1} H(x_k^o) P_{k|k-1}^{\text{CRLB}} \tag{37a}$$

$$P_{k+1|k}^{\text{CRLB}} = F(x_k^o) P_{k|k}^{\text{CRLB}} F^T(x_k^o) + G(x_k^o) Q_k G(x_k^o)^T. \tag{37b}$$

The following remarks summarize the CRLB theory with respect to the PF:

1) For a linear Gaussian model

$$x_{k+1} = F_k x_k + G_k v_k, \qquad v_k \sim \mathcal{N}(0, Q_k) \tag{38a}$$

$$y_k = H_k x_k + e_k, \qquad e_k \sim \mathcal{N}(0, R_k) \tag{38b}$$

the KF covariance $P_{k|k}$ coincides with $P_{k|k}^{\text{CRLB}}$. That is, the CRLB bound is attainable in the linear Gaussian case.

2) In the linear non-Gaussian case, the covariances Q_k, R_k, and P_0 are replaced with the inverse intrinsic accuracies $\mathcal{I}_{v_k}^{-1}$, $\mathcal{I}_{e_k}^{-1}$ and $\mathcal{I}_{x_0}^{-1}$, respectively. Intrinsic accuracy is defined as the Fisher information with respect to the location parameter, and the inverse intrinsic accuracy is always smaller than the covariance. As a consequence of this, the CRLB is always smaller for non-Gaussian noise than for Gaussian noise with the same covariance. See [58] for the details.

3) The parametric CRLB is a function of the true state trajectory $x_{1:k}^o$ and can thus be computed only in simulations or when ground truth is available from a reference system.

4) The posterior CRLB is the parametric CRLB averaged over all possible trajectories $P_{k|k}^{\text{PostCRLB}} = \text{E}(P_{k|k}^{\text{ParCRLB}})$. The expectation makes its computation quite complex in general.

5) In the linear Gaussian case, the parametric and posterior bounds coincide.

6) The covariance of the state estimate from the PF is bounded by the CRLB. The CRLB theory also says that the PF estimate attains the CRLB bound asymptotically in both the number of particles and the information in the model (basically the SNR).

Consult [59] for details on these issues.

VII. COMPLEXITY BOTTLENECKS

It is instructive and recommended to generate a profile report from an implementation of the PF. Quite often, unexpected bottlenecks are discovered that can be improved with a little extra work.

A. Resampling

One real bottleneck is the resampling step. This crucial step has to be performed at least regularly when N_{eff} becomes too small.

The resampling can be efficiently implemented using a classical algorithm for sampling N ordered independent identically distributed variables according to [60], commonly referred to as Ripley's method:

```
function [x,w]=resample(x,w)
  % Multinomial sampling with Ripley's method
  u=cumprod(rand(1,N).^(1./[N:-1:1]));
  u=fliplr(u);
  wc=cumsum(w);
  k=1;
  for i=1:N
    while(wc(k)<u(i))
      k=k+1;
    end
    ind(i)=k;
  end
  x=x(ind,:);
  w=ones(1,N)./N;
```

The complexity of this algorithm is linear in the number of particles N, which cannot be beaten if the implementation is done at a sufficiently low level. For this reason this is the most frequently suggested algorithm also in the PF literature. However, in engineering programming languages such as MATLAB™, vectorized computations are often an order of magnitude faster than code based on "for" and "while" loops.

The following code also implements the resampling needed in the PF by completely avoiding loops.

```
function [x,w]=resample(x,w)
  % Multinomial sampling with sort
  u=rand(N,1);
  wc=cumsum(w);
  wc=wc/wc(N);
  [dum,ind1]=sort([u;wc]);
  ind2=find(ind1<=N);
  ind=ind2-(0:N-1)';
  x=x(ind,:);
  w=ones(1,N)./N;
```

This implementation relies on the efficient implementation of sort. Note that sorting is of complexity $N \log_2(N)$ for low-level implementations, so in theory it should not be an alternative to Ripley's method for sufficiently large N. However, as Fig. 4 illustrates, the sort algorithm is a factor of five faster for one instance of a vector-oriented programming

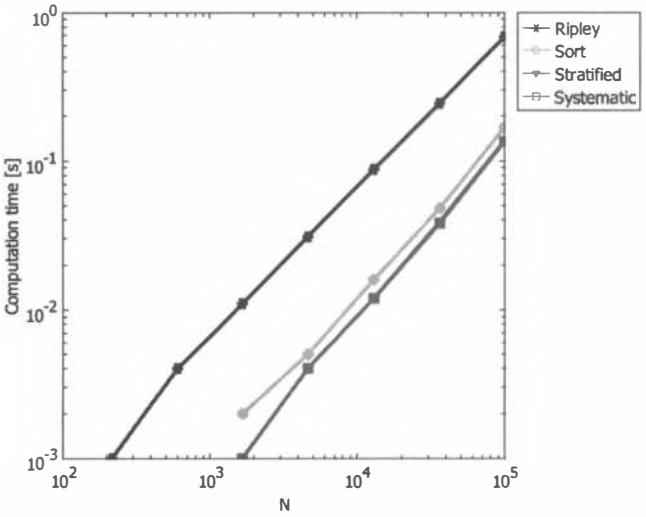

Fig. 4. Computational complexity in vectorized language of two different resampling algorithms: Ripley and sort.

language. Using interpreters with loop optimization reduces this difference, but the sort algorithm is still an alternative.

Note that this code does not use the fact that `wc` is already ordered. The sorting also gets further simplified if the sequence of uniform numbers is ordered. This is one advantage of systematic or stratified sampling [16], where the random number generation is replaced with one of the following lines:

```
% Stratified sampling
u=([0:N-1]'+(rand(N,1)))/N;
% Systematic sampling
u=([0:N-1]'+rand(1))/N;
```

Both the code based on `sort` and `for`, `while` are possible. Another advantage with these options is that the state space is more systematically covered, so there will not be any large uncovered volumes existing at random.

B. Likelihood Evaluation and Iterated Measurement Updates

The likelihood evaluation can be a real bottleneck if not properly implemented. In the case that there are several independent sensors, an iterated measurement update can be performed. Denote the M sensor observations y_k^j, for $j = 1,2,\ldots,M$. Then, independence directly gives

$$p(y_k \mid x_k) = \prod_{j=1}^{M} p(y_k^j \mid x_k). \tag{39}$$

This trick is even simpler than the corresponding iterated measurement update in the KF.

However, this iterated update is not necessarily the most efficient implementation. One example is the multivariate Gaussian distribution for independent measurements

$$y_{k,j} = h_j(x_k^i) + e_{k,j}, \qquad e_{k,j} \sim \mathcal{N}(0, R_{k,j}). \tag{40}$$

The likelihood is given by

$$p(y_k \mid x_k^i) \propto e^{-0.5\sum_{j=1}^{M}(y_{k,j} - h_j(x_k^i))^T R_{k,j}^{-1}(y_{k,j} - h_j(x_k^i))} \tag{41a}$$

$$= \prod_{j=1}^{M} e^{-0.5(y_{k,j} - h_j(x_k^i))^T R_{k,j}^{-1}(y_{k,j} - h_j(x_k^i))}. \tag{41b}$$

The former equation with a sum should be used to avoid extensive calls to the exponential function. Even here, the process for vectorizing the calculations in the sum for all particles in parallel is not trivial.

C. Time Update Sampling

Generating random numbers from nonstandard proposals may be time consuming. Then, remembering that dithering is often a necessary practical trick to tune the PF, one should investigate proposals including dithering noise that are as simple as possible to sample from.

D. Function Evaluations

When all issues above have been dealt with, the only thing that remains is to evaluate the functions $f(x,v)$ and $h(x)$. These functions are evaluated a large number of times, so it is worthwile to spend time optimizing their implementation. An interesting idea is to implement these in dedicated hardware taylored to the application. This was done using analog hardware in [61] for an arctangens function, which is common in sensor models for bearing measurements.

E. PF versus EKF

The computational steps of EKF (4) and SIR-PF (16) are compared with the KF in Table I. The EKF requires only one function evaluation of $f(x,v)$ and $h(x)$ per time step, while the PF requires N evaluations. However, if the gradients are not available analytically in the EKF, then at least another n_x evaluations of both $f(x,v)$ and $h(x)$ are needed. These numbers increase when the step size of the numeric gradients are adaptive. Further, if the process noise is not additive, even more numerical derivatives are needed. However, the PF is still roughly a factor N/n_x more complex.

The most time consuming step in the KF is the Riccati recursion of the matrix P. Here, either the matrix multiplication FP in the time update or the matrix inversion in the measurement update dominate for large enough models. Neither of these are needed in the PF. The time update of the state is the same.

The complexity of a matrix inversion using state-of-the-art algorithms [62] is $\mathcal{O}(n_y^{2.376})$. The matrix inversion in the measurement update can be

TABLE I
Comparison of EKF in (4) and SIR-PF in (16): Main Computational Steps

Algorithm	Extended Kalman filter	Particle filter
Time update	$F = \frac{\partial f(x,v)}{\partial x}$, $G = \frac{\partial f(x,v)}{\partial v}$	$v^i \sim p_v$
	$x := f(x,0)$	$x^i := f(x^i, v^i)$
	$P := FPF^T + GQG^T$	
Measurement update	$H = \frac{\partial h(x)}{\partial x}$	$w^i := w^i p_e(y - h(x^i))$
	$K = PH^T(HPH^T + R)^{-1}$	
	$x := x + K(y - h(x))$	
	$P := P - KHP$	
Estimation	$\hat{x} = x$	$\hat{x} = \sum_{i=1}^{N} w^i x^i$
Resampling	—	$x^i \sim \sum_{j=1}^{N} w^j \delta(x - x^j)$

avoided using the iterated measurement update. The condition is that the covariance matrix R_k is (block-) diagonal.

As a first-order approximation for large n_x, the KF is $\mathcal{O}(n_x^3)$ from the matrix multiplication FP, while the PF is $\mathcal{O}(Nn_x^2)$ for a typical dynamic model where all elements of $f(x,v)$ depend on all states, for instance the linear model $f(x,v) = Fx + v$. Also from this perspective, the PF is a factor N/n_x computationally more demanding than the EKF.

VIII. MARGINALIZED PARTICLE FILTER THEORY

The main purpose of the marginalized PF (MPF) is to keep the state dimension small enough for the PF to be feasible. The resulting filter is called the MPF or the Rao-Blackwellized PF (RBPF), and it has been known for quite some time under different names, see, e.g., [49, 63–68].

The MPF utilizes possible linear Gaussian substructures in the model (1). The state vector is assumed partitioned as $x_k = ((x_k^n)^T, (x_k^l)^T)^T$ where x_k^l enters both the dynamic model and the observation model linearly. We refer a bit informally to x_k^l as the linear state and x_k^n as the nonlinear state. MPF essentially represents x_k^n with particles and applies one KF per particle. The KF provides the conditional distribution for x_k^l conditioned on the trajectory $x_{1:k}^n$ of nonlinear states and the past observations.

A. Model Structure

A rather general model, containing a conditionally linear Gaussian substructure is given by

$$x_{k+1}^n = f_k^n(x_k^n) + F_k^n(x_k^n)x_k^l + G_k^n(x_k^n)v_k^n \quad (42a)$$

$$x_{k+1}^l = f_k^l(x_k^n) + F_k^l(x_k^n)x_k^l + G_k^l(x_k^n)v_k^l \quad (42b)$$

$$y_k = h_k(x_k^n) + H_k(x_k^n)x_k^l + e_k. \quad (42c)$$

The state vector and Gaussian state noise are partitioned as

$$x_k = \begin{pmatrix} x_k^n \\ x_k^l \end{pmatrix}$$

$$v_k = \begin{pmatrix} v_k^n \\ v_k^l \end{pmatrix} \sim \mathcal{N}(0, Q_k) \quad (42d)$$

$$Q_k = \begin{pmatrix} Q_k^n & Q_k^{ln} \\ (Q_k^{ln})^k & Q_k^l \end{pmatrix}.$$

Furthermore, x_0^l is assumed Gaussian, $x_0^l \sim \mathcal{N}(\bar{x}_0, \bar{P}_0)$. The density of x_0^n can be arbitrary, but it is assumed known. The underlying purpose with this model structure is that conditioned on the sequence $x_{1:k}^n$, (42) is linear in x_k^l with Gaussian prior, process noise, and measurement noise, respectively, so the KF theory applies.

B. Algorithm Overview

The MPF relies on the following key factorization:

$$p(x_k^l, x_{1:k}^n \mid y_{1:k}) = p(x_k^l \mid x_{1:k}^n, y_{1:k}) p(x_{1:k}^n \mid y_{1:k}). \quad (43)$$

These two factors decompose the nonlinear filtering task into two subproblems:

1) A KF operating on the conditionally linear, Gaussian model (42) provides the exact conditional posterior $p(x_k^l \mid x_{1:k}^n, y_{1:k}) = \mathcal{N}(x_k^l; \hat{x}_{k|k}^l(x_{1:k}^{n,i}), P_{k|k}^l(x_{1:k}^{n,i}))$. Here, (42a) becomes an extra measurement for the KF with $x_{k+1}^n - f_k^n(x_k^n)$ acting as the observation.

2) A PF estimates the filtering density of the nonlinear states. This involves a nontrivial marginalization step by integrating over the state space of all x_k^l using the law of total probability

$$\begin{aligned} p(x_{1:k+1}^n \mid y_{1:k}) &= p(x_{1:k}^n \mid y_{1:k}) p(x_{k+1}^n \mid x_{1:k}^n, y_{1:k}) \\ &= p(x_{1:k}^n \mid y_{1:k}) \int p(x_{k+1}^n \mid x_k^l, x_{1:k}^n, y_{1:k}) \\ &\quad \times p(x_k^l \mid x_{1:k}^n, y_{1:k}) dx_k^l \\ &= p(x_{1:k}^n \mid y_{1:k}) \int p(x_{k+1}^n \mid x_k^l, x_{1:k}^n, y_{1:k}) \\ &\quad \times \mathcal{N}(x_k^l; \hat{x}_{k|k}^l(x_{1:k}^{n,i}), P_{k|k}^l(x_{1:k}^{n,i})) dx_k^l. \end{aligned} \quad (44)$$

The intuitive interpretation of this result is that the linear state estimate acts as an extra state noise in (42a) when performing the PF time update.

The time and measurement updates of KF and PF are interleaved, so the timing is important. The information structure in the recursion is described in Algorithm 2. Table II summarizes the information steps in Algorithm 2. Note that the time index appears

TABLE II
Summary of the Information Steps in Algorithm 2 for the Marginalized PF Utilizing a Linear Gaussian Substructure

Prior	$p(x_k^l, x_{1:k}^p \mid y_{1:k}) = p(x_k^l \mid x_{1:k}^p, y_{1:k}) p(x_{1:k}^p \mid y_{1:k})$
PF TU	$p(x_{1:k}^p \mid y_{1:k}) \Rightarrow p(x_{1:k+1}^p \mid y_{1:k})$
KF TU	$p(x_k^l \mid x_{1:k}^p, y_{1:k}) \Rightarrow p(x_{k+1}^l \mid x_{1:k}^p, y_{1:k})$
KF dyn MU	$p(x_{k+1}^l \mid x_{1:k}^p, y_{1:k}) \Rightarrow p(x_{k+1}^l \mid x_{1:k+1}^p, y_{1:k})$
PF MU	$p(x_{1:k+1}^p \mid y_{1:k}) \Rightarrow p(x_{1:k+1}^p \mid y_{1:k+1})$
KF obs MU	$p(x_{k+1}^l \mid x_{1:k+1}^p, y_{1:k}) \Rightarrow p(x_{k+1}^l \mid x_{1:k+1}^p, y_{1:k+1})$
Posterior	$p(x_{k+1}^l, x_{1:k+1}^p \mid y_{1:k+1}) = p(x_{k+1}^l \mid x_{1:k+1}^p, y_{1:k+1}) p(x_{1:k+1}^p \mid y_{1:k+1})$

five times in the right hand side expansion of the prior. The five steps increase each k one at the time to finally form the posterior at time $k+1$.

ALGORITHM 2 Marginalized Particle Filter *With reference to the standard PF in Algorithm* 1 *and the KF; iterate the following steps for each time step:*

1) *PF measurement update and resampling using* (42c) *where* x_k^l *is interpreted as measurement noise.*
2) *KF measurement update using* (42c) *for each particle* $x_{1:k}^{n,i}$.
3) *PF time update using* (42a) *where* x_k^l *is interpreted as process noise.*
4) *KF time update using* (42b) *for each particle* $x_{1:k}^{n,i}$.
5) *KF extra measurement update using* (42a) *for each particle* $x_{1:k}^{n,i}$.

The posterior distribution for the nonlinear states is given by a discrete particle distribution as usual, while the posterior for the linear states is given by a Gaussian mixture:

$$p(x_{1:k}^n \mid y_{1:k}) \approx \sum_{i=1}^{N} w_{k|k}^i \delta(x_{1:k}^n - x_{1:k}^{n,i}) \tag{45a}$$

$$p(x_k^l \mid y_{1:k}) \approx \sum_{i=1}^{N} w_{k|k}^i \mathcal{N}(x_k^l; \hat{x}_{k|k}^l(x_{1:k}^{n,i}), P_{k|k}^l(x_{1:k}^{n,i})). \tag{45b}$$

For a complete derivation, see [67]. As demonstrated in [69], standard KF and particle filtering code can be reused when implementing the MPF. The model (42) can be further generalized by introducing an additional discrete mode parameter, giving a larger family of marginalized filters; see [68].

C. Complexity Issues

In general, each KF comes with its own Riccati equation. However, the Riccati equation is the same if the following three conditions are satisfied:

$$G_k^n(x_k^n) = G_k^n \quad \text{or} \quad F_k^n(x_k^n) = 0 \tag{46a}$$

$$G_k^l(x_k^n) = G_k^l \tag{46b}$$

$$H_k(x_k^n) = H_k. \tag{46c}$$

It is easy to verify that the Ricatti equations in this case only involve matrices that are the same for all trajectories $x_{1:k}^{n,i}$. This implies a significant complexity reduction.

One important special case of (42) in practice is a model with linear state equations with a nonlinear observation which is a function of a (small) part of the state vector

$$x_{k+1}^n = F_k^{nn} x_k^n + F_k^{nl} x_k^l + G_k^n v_k^n \tag{47a}$$

$$x_{k+1}^l = F_k^{ln} x_k^n + F_k^{ll} x_k^l + G_k^l v_k^l \tag{47b}$$

$$y_k = h_k(x_k^n) + e_k. \tag{47c}$$

For instance, all applications in Section X fall into this category. In this case, step 3 in Algorithm 2 disappears.

The MPF appears to add quite a lot of overhead computations. It turns out, however, that the MPF is often more efficient. It may seem impossible to give any general conclusions, so application-dependent simulation studies have to be performed. Nevertheless, quite realistic predictions of the computational complexity can be done with rather simple calculations, as pointed out in [70]. The result is that for the case when (46) is satisfied, MPF should always be more efficient, otherwise the complexities are comparable.

D. Variance Reduction

The MPF reduces the variance of the linear states which is demonstrated below. The law of total variance says that

$$\text{cov}(U) = \text{cov}(\text{E}(U \mid V)) + \text{E}(\text{cov}(U \mid V)). \tag{48}$$

Letting $U = x_k^l$ and $V = x_{1:k}^n$ gives the following decomposition of the variance of the PF:

$$\underbrace{\text{cov}(x_k^l)}_{\text{PF}} = \text{cov}(\text{E}(x_k^l \mid x_{1:k}^n)) + \text{E}(\text{cov}(x_k^l \mid x_{1:k}^n)) \tag{49a}$$

$$= \underbrace{\text{cov}(\hat{x}_{k|k}^l(x_{1:k}^{n,i}))}_{\text{MPF}} + \sum_{i=1}^{N} w_k^i \underbrace{P_{k|k}(x_{1:k}^{n,i})}_{\text{KF}}. \tag{49b}$$

Here, we recognize $(x_k^l \mid x_{1:k}^{n,i})$ as the Gaussian distribution, delivered by the KF, conditioned on the trajectory $x_{1:k}^{n,i}$. Now, the MPF computes the mean of each trajectory as $\hat{x}_{k|k}^l(x_{1:k}^{n,i})$, and the unconditional mean estimator is simply the mean of these,

$$\hat{x}_{k|k}^l = \sum_{i=1}^{N} w_k^i \hat{x}_{k|k}^l(x_{1:k}^{n,i}) \tag{50}$$

and its covariance follows from the first term in (49b). The first term in (49b) corresponds to the spread of the mean contribution from the Gaussian mixture, and this is the only uncertainty in the MPF.

The variance decomposition shows that the covariance for the MPF is strictly smaller than the corresponding covariance for the PF. This can also be seen as a result of Rao-Blackwell's lemma, see, e.g., [37], and the marginalization is commonly referred to as Rao-Blackwellization. This result says that the improvement in the quality of the estimate is given by the term $\mathrm{E}(\mathrm{cov}(x_k^l \mid x_{1:k}^n))$. Note that when (46) is satisfied, then $P_{k|k}^i = P_{k|k}$ and thus $\sum_{i=1}^N w_k^i P_{k|k}^i = P_{k|k}$. That is, the KF covariance $P_{k|k}$ is a good indicator of how much that has been gained in using the MPF instead of the PF. As a practical rule of thumb, the gain in MPF increases as the uncertainty in the linear state increases in the model. Further discussions regarding the variance reduction property of the MPF are provided for instance in [49].

The variance reduction in the MPF can be used in two different ways:

1) With the same number of particles, the variance in the estimates of the linear states can be decreased.
2) With the same performance in terms of variance for the linear states, the number of particles can be decreased.

This is schematically illustrated in Fig. 5, for the case when (46) is satisfied, implying that the same covariance matrix can be used for all particles. The two alternatives above are illustrated for the case when a PF with 10,000 particles is first applied and then replaced by the MPF.

E. MPF Synonyms

The following names have been suggested for the filter in this section:

1) MPF as is motivated by the nontrivial marginalization step (44).
2) "Rao-Blackwellized particle filter," as motivated by the variance reduction in (49).
3) "Mixture Kalman filter," as motivated by the various mixture distributions that appear, for instance in (45b).
4) Another logical name would be "separable particle filter" in parallel to the well-established separable nonlinear least squares problem. In fact, the special case of a static problem where only (42c) exists falls into this class of problems. Here, the weighted least squares estimate of x_k^l is first computed as a function of $x_{1:k}^n$, which is then backsubstituted into the model with its estimation covariance to form a nonlinear least squares problem in $x_{1:k}^n$ only.

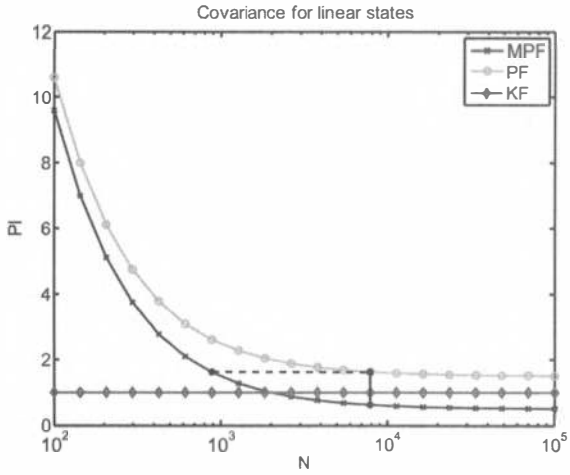

Fig. 5. Schematic view of how covariance of linear part of state vector depends on number of particles for PF and MPF, respectively. Gain in MPF is given by KF covariance.

F. Illustrating Example

The aim here is to illustrate how the MPF works using the following nonlinear stochastic system

$$x_{k+1}^n = x_k^l x_k^n + v_k^n \tag{51a}$$

$$x_{k+1}^l = x_k^l + v_k^l \tag{51b}$$

$$y_k = 0.2(x_k^n)^2 + e_k \tag{51c}$$

where the noise is assumed white and Gaussian according to

$$v_k = \begin{pmatrix} v_k^n \\ v_k^l \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0.25 & 0 \\ 0 & 10^{-4} \end{pmatrix}\right) \tag{51d}$$

$$e_k \sim \mathcal{N}(0,1). \tag{51e}$$

The initial state x_0 is given by

$$x_0 \sim \mathcal{N}\left(\begin{pmatrix} 0.1 \\ 0.99 \end{pmatrix}, \begin{pmatrix} 16 & 0 \\ 0 & 10^{-3} \end{pmatrix}\right). \tag{51f}$$

This particular model was used in [71], where it illustrated grid-based (point-mass) filters. Obviously, the states can be estimated by applying the standard PF to the entire state vector. However, a better solution is to exploit the conditionally linear, Gaussian substructure that is present in (51). The nonlinear process x_k^n is a first-order auto regressive (AR) process, where the linear process x_k^l is the time-varying parameter. The linear, Gaussian substructure is used by the MPF and the resulting

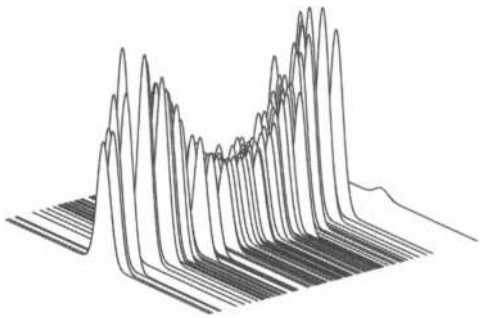

Fig. 6. Estimated filter pdf for system (51) at time 10, $p(x_{10} \mid y_{1:10})$ using MPF. It is instructive to see that linear state x_{10}^l is estimated by Gaussian densities (from the KF), and position along the nonlinear state x_{10}^n is given by a particle (from the PF).

filtering density function at time 10, $p(x_{10} \mid y_{1:10})$ before the resampling step is shown in Fig. 6 (for a particular realization). In this example 2000 particles were used, but only 100 of them are plotted in Fig. 6 in order to obtain a clearer illustration of the result. The figure illustrates the fact that the MPF is a combination of the KF and the PF. The density functions for the linear states are provided by the KFs, which is evident from the fact that the marginals $p(x_k^{l,i} \mid y_{1:k})$ are given by Gaussian densities. Furthermore, the nonlinear state estimates are provided by the PF. Hence, the linear states are given by a parametric estimator (the KF), whereas the nonlinear states are given by a nonparametric estimator (the PF). In this context the MPF can be viewed as a combination of a parametric and a nonparametric estimator.

IX. PARTICLE FILTER CODE EXAMPLES

This section gives concrete MATLAB™-like code for a general SIR-PF, and applies it to a fully annotated simulation example. Further, object-oriented implementations of nonlinear filters are illustrated on target tracking applications. The classes and examples are available in the Signal and Systems Lab; URL: www.control.isy.liu.se/~fredrik/sigsyslab.

A. Terrain-Based Positioning

The following scalar state example suits three purposes. First, it enables intuitive graphical illustrations. Second, it introduces the positioning applications in the next section. Third, it should be easy to implement for interested readers for reproducing the example and extending the code to other applications.

Consider the model

$$x_{k+1} = x_k + u_k + v_k \tag{52a}$$

$$y_k = h(x_k) + e_k \tag{52b}$$

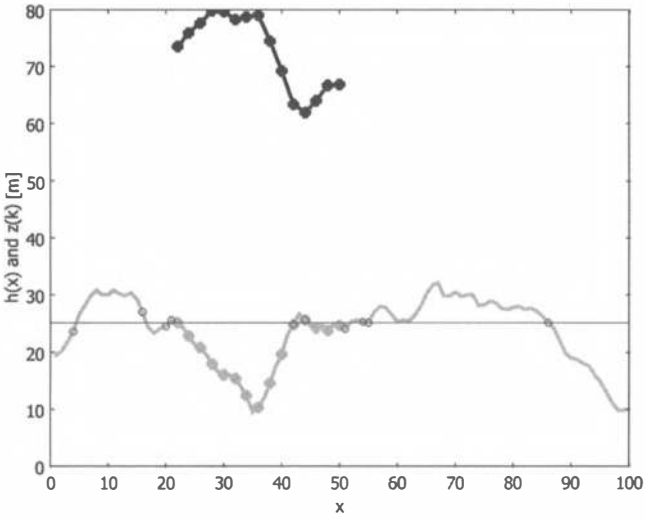

Fig. 7. Aircraft altitude $z(x_k)$ (upper dark line) as a function of position x_k (dots on upper dark line) and nonlinear measurement relation $h(x)$ (lower gray line) for the model in (52). Computed terrain altitude $h(x_1)$ is also marked, and circle is put in all grid points that give best match to this altitude.

where both the state and the measurement are scalar valued. This model mimics a navigation problem in one-dimension, where u_k is a measurable velocity, v_k unmeasurable velocity disturbance, and the observation y_k measures the terrain altitude, which is known in the database $h(x)$. An illustration from a real application is found in Fig. 6. Note that the terrain altitude as a measurement relation is not one to one, since a given terrain altitude is found at many different positions. However, the observed terrain profile will after a short time be unique for the flown trajectory.

Fig. 7 shows a trajectory, and one realization of the nonlinear function terrain profile $h(x)$, generated by the code below.

```
x=1:100; % Map grid
h=20+filter(1,[1 -1.8 0.81],randn(1,100));
   % Terrain altitude
N=15;
z=100+filter(1,[1 -1.8 0.81],randn(N,1));
   % Measurement input
u=2*ones(N,1); % State input
x0=20+cumsum(u); % True position
y=z-interp1(x,h,x0); % Noisefree measurement
yn=y+1*randn(N,1); % Noisy measurement
plot(x0,y,'o-b',x,h,'g',x0,z-y,'go',
   'linewidth',3)
```

The horizontal line indicates where the first measurement is taken. There are ten different intersections between the terrain profile and this observation, where the grid point just before each intersection is marked in the figure. This is clearly a problem where the posterior is multimodal after the first measurement update.

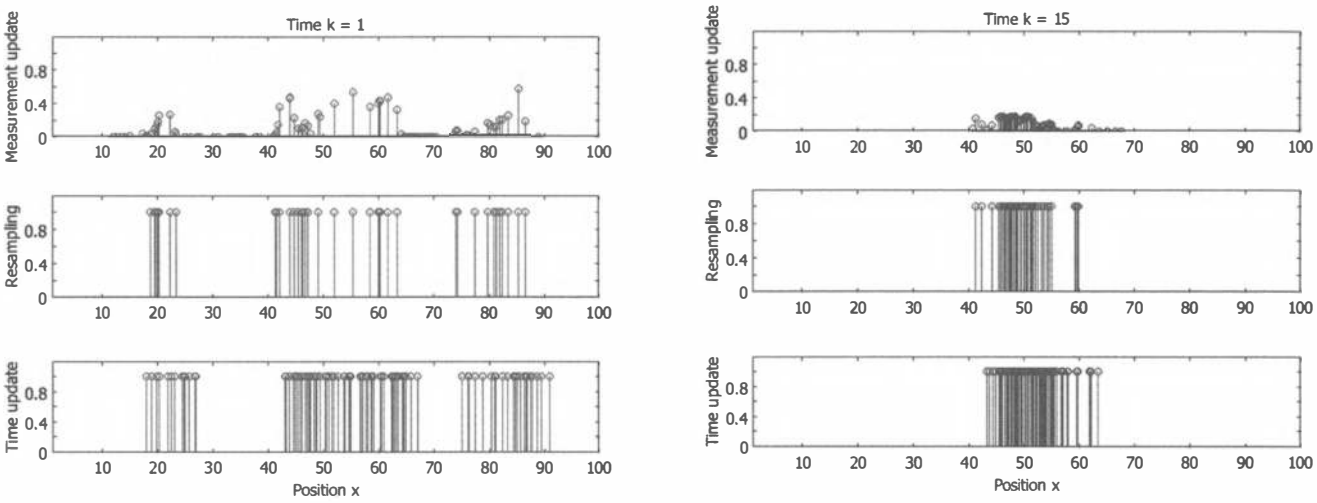

Fig. 8. First two subplots: approximations of $p(x_k \mid y_{1:k})$ before and after resampling, respectively. Last subplot: approximations of $p(x_{k+1} \mid y_{1:k})$.

The following code lines define the model (52) as an object structure:

```
m.f=inline('x+u','x','u');
m.h=inline('z-interp1(x,h,xp)','xp','h',
   'x','z');
m.pv=ndist(0,5); m.pe=ndist(0,1);
m.p0=udist(10,90);
```

The pdf classes `ndist` and `udist` with the methods `rand` and `pdf` are assumed to be available. A script that both implements a version of the PF and also animates all the partial results is given below:

```
Np=100; w=ones(Np,1)/Np;
xp=rand(m.p0,Np); % Initialization
for k=1:N;
   yp=m.h(xp,h,x,z(k)); % Measurement pred.
   w=w.*pdf(m.pe,repmat(yn(k,:),Np,1)-yp);
      % Likelihood
   w=w/sum(w); % Normalization
   subplot(3,1,1), stem(xp,Np*w/10)
   xhat(k,:)=w(:)'*xp; % Estimation
   [xp,w]=resample(xp,w); % Resampling
   subplot(3,1,2), stem(xp,Np*w)
   v=rand(m.pv,Np); % Random process noise
   xp=m.f(xp,u(k,:)')+v; % State prediction
   subplot(3,1,3), stem(xp,Np*w)
end
```

Code examples of the function `resample` are given in Section VIIA. Fig. 8 shows the posterior density approximation at two time instants. Fig. 8(a) shows first the unnormalized weights after the measurement update, which with this uniform prior is just the likelihood function $p(y_1 \mid x_0) = p(y_1)$, and then follows the particle distribution after resampling (where $w^i = 1/N$) and finally the particles after time update (which is just a translation with u_1).

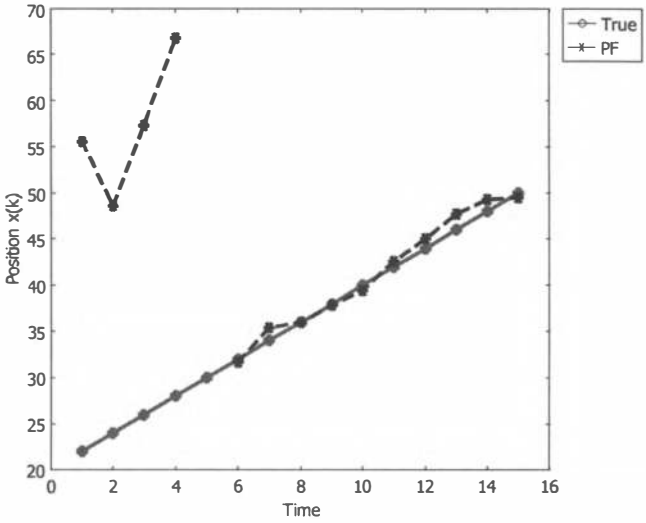

Fig. 9. True and estimated state as function of time.

Fig. 8(b) illustrates the same thing after the 15th measurement. The posterior is now more clustered to a unimodal distribution. Fig. 9 shows the position error as a function of time. The break point in performance indicates when the multimodal posterior distribution becomes unimodal.

B. Target Tracking

In an object-oriented implementation, simulation studies can be performed quite efficiently. The following example compares different filters for a simple target tracking model:

$$x_{k+1} = \begin{pmatrix} I_2 & T_s I_2 \\ 0 & I_2 \end{pmatrix} x_k + \begin{pmatrix} \frac{T_s^2}{2} I_2 \\ T_s I_2 \end{pmatrix} v_k,$$

$$v_k \sim \mathcal{N}(0, 1I_2), \quad x_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{53a}$$

$$y_k = (I_2 \quad 0)x_k + e_k, \qquad e_k \sim \mathcal{N}(0, 0.01I_2). \tag{53b}$$

The observation model is first linear to be comparable to the KF that provides the optimal estimate. The example makes use of two different objects:

1) Signal object where the state $x_{1:k}$ and observation $y_{1:k}$ sequences are stored with their associated uncertainty (covariances P_k^x, P_k^y or particle representation). Plot methods in this class can then automatically provide confidence bounds.
2) Model objects for linear and nonlinear models, with methods implementing simulation and filtering algorithms.

The purpose of the following example is to illustrate how little coding is required with this object-oriented approach. First, the model is loaded from an extensive example database as a linear state-space model. It is then converted to the general nonlinear model structure, which does not make use of the fact that the underlying model is linear.

```
mss=exlti('cv2d');
mnl=nl(mss);
```

Now, the following state trajectories are compared:

1) the true state from the simulation.
2) the CRLB computed from the nonlinear model.
3) the KF estimate using the linear model.
4) the EKF using the nonlinear model.
5) the UKF using the nonlinear model.
6) the PF using the nonlinear model.

For all except the first one, a confidence ellipsoid indicates the position estimation uncertainty.

```
y=simulate(mss,10);
xhat1=kalman(mss,y);
xhat2=ekf(mnl,y);
xhat3=ukf(mnl,y);
xhat4=pf(mnl,y,'Np',1000);
xcrlb=crlb(mnl,y);
xplot2(xcrlb,xhat4,xhat3,xhat2,xhat1,
    'conf',90)
```

Fig. 10 validates that all algorithms provide comparable estimates in accordance with the CRLB.

Now, consider the case of a radar sensor that provides good angle resolution but poor range. The measurement relation in model (53b) is changed to

$$y_k = \begin{pmatrix} \arctan\left(\dfrac{x_k^{(2)} - \theta^{(2)}}{x_k^{(1)} - \theta^{(1)}}\right) \\ \sqrt{(x_k^{(1)} - \theta^{(1)})^2 + (x_k^{(2)} - \theta^{(2)})^2} \end{pmatrix} + e_k$$

$$e_k \sim \mathcal{N}(0, \text{diag}(0.0001, 0.3)). \qquad (54)$$

Fig. 11 compares EKF and PF with respect to the CRLB. The PF performs well, where the covariances fitted to the particles are very similar to the CRLB. The EKF is slightly biased and too optimistic about the uncertainty, which is a typical behavior when neglecting higher order terms in the nonlinearities.

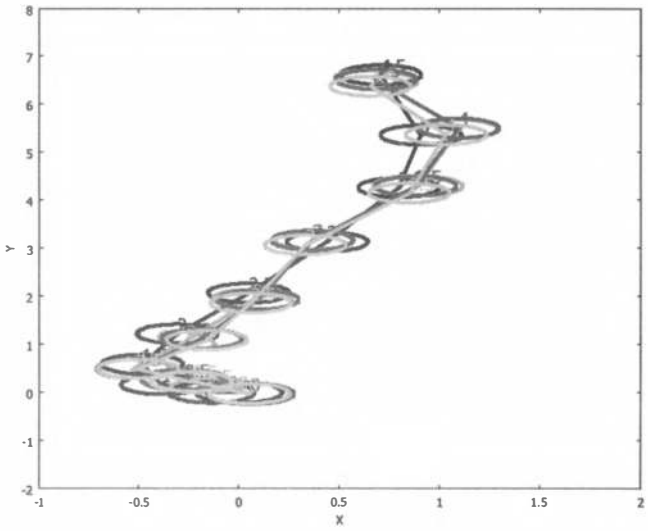

Fig. 10. Simulated trajectory using constant velocity two-dimensional motion model with position sensor, where plots show CRLB (darkest) and estimates from KF, EKF, UKF, and PF, respectively.

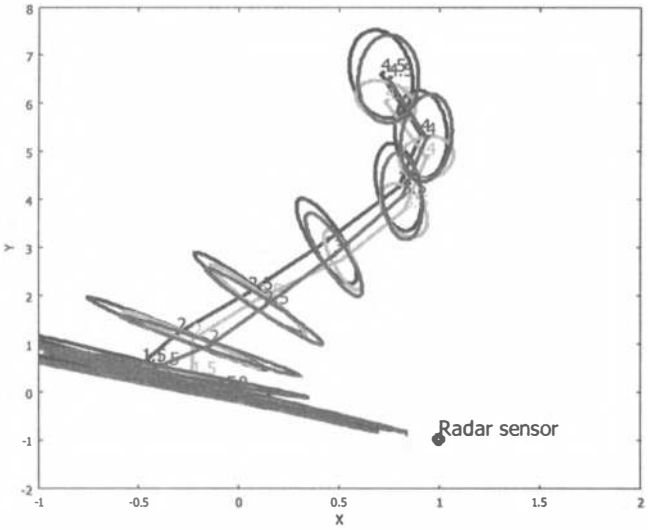

Fig. 11. Simulated trajectory using constant velocity two-dimensional motion model with radar sensor, where plots show CRLB (darkest) and estimates from EKF (small ellipsoids) and PF, respectively.

However, the performance of all filters is comparable, and the nonlinear measurement relation does not in itself motivate computer-intensive algorithms in this case.

C. Growth Model

The following toy example was used in the original paper [15]:

$$x_{k+1} = \frac{x_k}{2} + 25\frac{x_k}{1 + x_k^2} + 8\cos(k) + v_k,$$

$$v_k \sim \mathcal{N}(0, 10), \quad x_0 \sim \mathcal{N}(5, 5) \qquad (55a)$$

$$y_k = \frac{x_k^2}{20} + e_k, \qquad e_k \sim \mathcal{N}(0, 1). \qquad (55b)$$

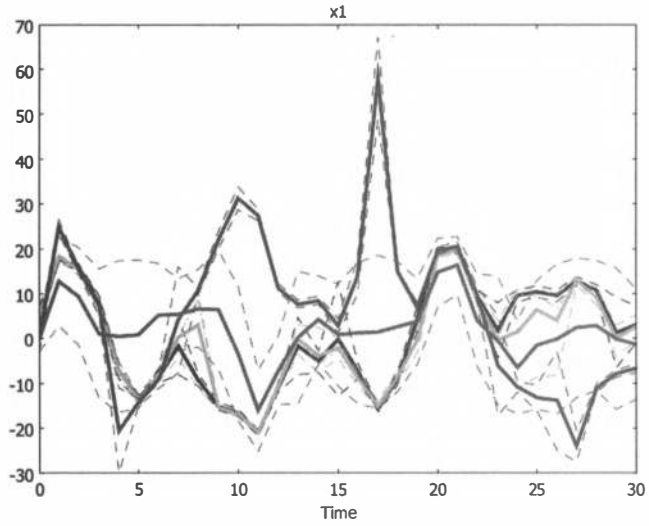

Fig. 12. Simulated trajectory using model (55), where plots show CRLB (darkest) and estimates from EKF, PF, and UKF, respectively. Table III summarizes performance.

It has since then been used many times in the particle filter literature, and it is often claimed to be a growth model. It is included here just because it has turned into a benchmark problem. The simulation code is

```
m=exnl('pfex');
z=simulate(m,30);
zcrlb=crlb(m,z);
zekf=ekf(m,z);
zukf=ukf(m,z);
zpf=pf(m,z);
xplot(zcrlb,zpf,zekf,zukf,'conf',90,'view',
   'cont','conftype',2)
[mean(zcrlb.Px) norm(z.x-zpf.x)
norm(z.x-zekf.x) norm(z.x-zukf.x)];
```

The last two lines produce the result in Fig. 12 and Table III, respectively. The conclusion from this example is that PF performs much better than the UKF which in turn performs much better than the EKF. Thus, this example illustrates quite nicely the ranking of the different filters.

X. PARTICLE FILTER POSITIONING APPLICATIONS

This section is concerned with four positioning applications of underwater vessels, surface ships, wheeled vehicles (cars), and aircraft, respectively. Though these applications are at first glance quite different, almost the same PF can be used in all of them. In fact, successful applications of the PF are described in literature which are all based on the same state-space model and similar measurement equations.

A. Model Framework

The positioning applications, as well as existing applications of fastSLAM, are all based on the

TABLE III
MSE Performance of the Estimates in Fig. 12 for the Benchmark Problem in (55)

CRLB	PF	UKF	EKF
8	18	54	132

model [72]

$$x_k = (X_k, Y_k, \psi_k)^T \tag{56a}$$

$$u_k = (V_k, \dot{\psi}_k)^T \tag{56b}$$

$$X_{k+1} = X_k + TV_k \cos(\psi_k) \tag{56c}$$

$$Y_{k+1} = X_k + TV_k \sin(\psi_k) \tag{56d}$$

$$\psi_{k+1} = \psi_k + T\dot{\psi}_k \tag{56e}$$

$$y_k = h(x_k) + e_k. \tag{56f}$$

Here, X_k, Y_k denote the Cartesian position, ψ_k the course or heading, T is the sampling interval, V_k is the speed, and $\dot{\psi}_k$ the yaw rate. The inertial signals V_k and $\dot{\psi}_k$ are considered as inputs to the dynamic model, and are given by onboard sensors. These are different in each of the four applications, and they are described in more detail in the subsequent sections. The measurement relation is based on a distance measuring equipment (DME) and a GIS. Both the DME and the GIS are different in the four applications, but the measurement principle is the same. By comparing the measured distance to objects in the GIS, a likelihood for each particle can be computed. It should here be noted that neither an EKF, UKF, nor KF bank is suited for such problems. The reason is that it is typically not possible to linearize the database other than in a very small neighborhood.

In common for the applications is that they do not rely on satellite navigation systems, which are assumed unavailable or provide insufficient navigation integrity. First, the inertial inputs, DME and GIS, for the four applications are described. Conclusions concering the PF from these applications are summarized in Section XII. Different ways to augment the state vector are described for each application in Section XI. The point is that the dimension of the state vector has to be increased in order to account for model errors and more complicated dynamics. This implies that the PF is simply not applicable, due to the high dimensional state vector.

The outline follows a bottom-up approach, starting with underwater vessels below sea level and ending with fighter aircraft in the air.

B. Underwater Positioning using a Topographic Map

The goal is to compute the position of a UW vessel. A sonar is measuring the distance d_1 to the sea floor. The depth of the platform itself d_2 can be

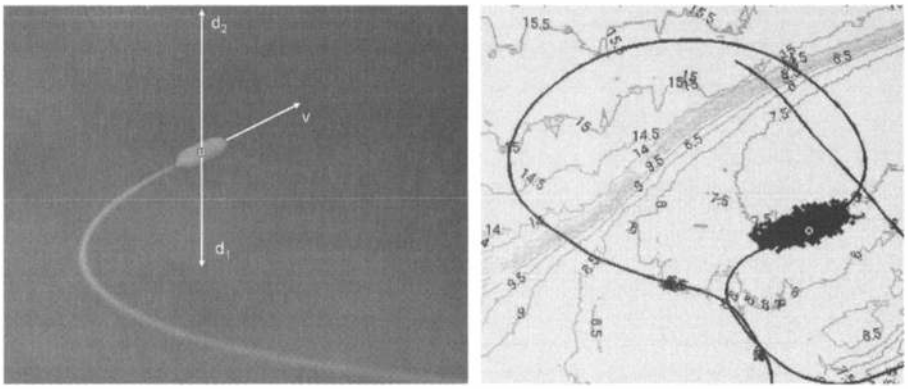

Fig. 13. Left plot is an illustration of UW vessel measuring distance d_1 to sea bottom, and absolute depth d_2. Sum $d = d_1 + d_2$ is compared with a bottom map as illustrated with contours in plot to right. Particle cloud illustrates snapshot of PF from known validation trajectory in field trial, see [75].

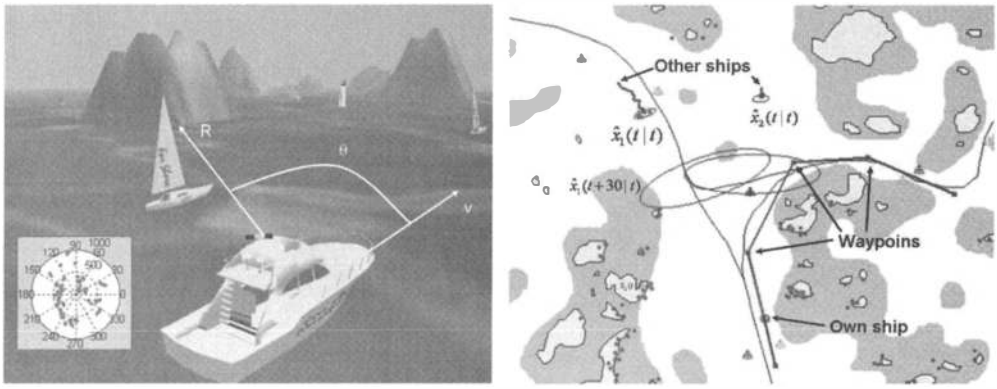

Fig. 14. Rotating radar returns detections of range R at body angle θ. Result of one radar revolution is conventionally displayed in polar coordinates as illustrated. Comparing the (R,θ) detections to sea chart as shown to (right), position and course are estimated by PF. When correctly estimated, radar overlay principle can be used for visual validation as also illustrated in sea chart. PF has to distinguish radar reflections from shore with clutter and other ships, see [76]. The latter can be used for conventional target tracking algorithms, and collision avoidance algorithms, as also illustrated to (right), see [77].

computed from pressure sensors or from a sonar directed upwards. By adding these distances, the sea depth at the position X_k, Y_k is measured. This can be compared to the depth in a dedicated sea chart with detailed topographical information, and the likelihood takes the combined effect of errors in the two sensors and the map into account, see [73]. Fig. 13 provides an illustration.

The speed V_k and yaw rate $\dot{\psi}_k$ in (56) are computed using simplified dynamic motion models based on the propeller speed and the rudder angle. It is important to note that since the PF does not rely on pure dead reckoning, such models do not have to be very accurate, see [74] for one simple linear model. An alternative is to use inertial measurement units (IMU) for measuring and computing speed and yaw rate.

Detailed seabed charts are so far proprietary military information, and most applications are also military. As an example of civilian use, oil companies are starting to use unmanned UW vessels for exploring the sea and oil platforms, and in this way they are building up their own maps.

C. Surface Positioning using a Sea Chart

The same principle as above can of course be used also for surface ships, which are constrained to be on the sea level ($d_2 = 0$). However, vectorized sea charts (for instance the S-57 standard) contain a commercially available worldwide map.

The idea is to use the radar as DME and compare the detections with the shore profile, which is known from the sea chart conditioned on the position X_k, Y_k and course ψ_k (most importantly, the ship orientation, but more on this later); see [73]. The likelihood function models the radar error, but must also take clutter (false detections) and other ships into account.

The left hand part of Fig. 14 illustrates the measurements provided by the radar, while the right hand part of the same figure shows the radar detections from one complete revolution overlayed on the sea chart. The inertial data can be computed from propeller speed and rudder angle using simplified dynamical models as above.

American and European maritime authorities have recently published reports highlighting the need for a

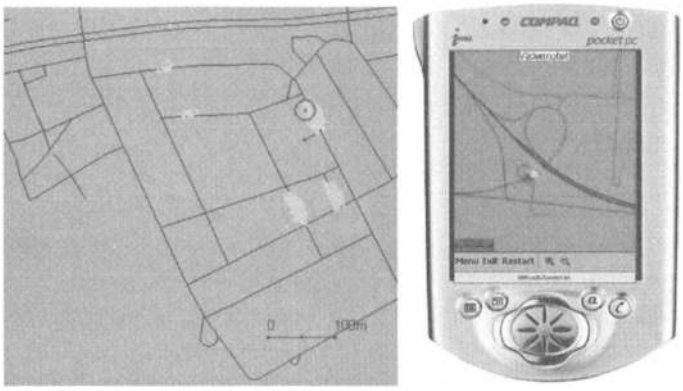

Fig. 15. Left: Example of multimodal posterior represented by number of distinct particle clouds from NIRA Dynamics navigation system. This is caused by regular road pattern and will be resolved after sufficiently long sequence of turns. Right: PF in embedded navigation solution runs in real time on pocket PC with serial interface to vehicle CAN data bus, see [80].

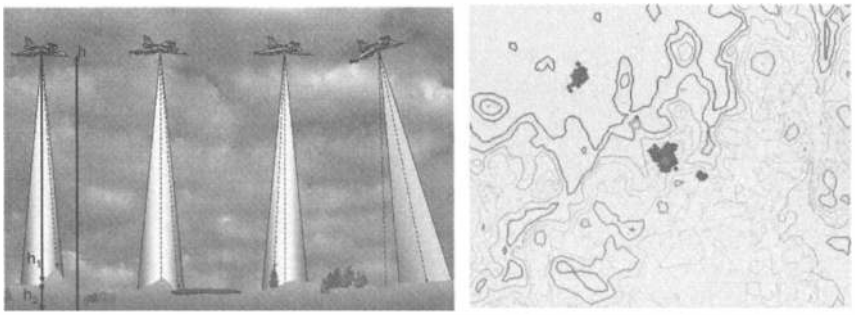

Fig. 16. Left figure is an illustration of an aircraft measuring distance h_1 to ground. Onboard baro-altitude supported INS system provides absolute altitude over sea level h, and difference $h_2 = h - h_1$ is compared to a topographical map. Right plot shows a snapshot of PF particle cloud just after aircraft has left sea in upper left corner. There are three distinct modes, where the one corresponding to the correct position dominates.

backup and support system to satellite navigation to increase integrity. The reason for this need is accidents and incidents caused by technical problems with the satellite navigation system and the risk of accidental or deliberate jamming. The LORAN standard offers one such supporting technique based on triangulation to radio beacons, see [78]. The PF solution here is a promising candidate, since it is, in contrast to LORAN, not sensitive to jamming nor does it require any infrastructure.

D. Vehicle Positioning using a Road Map

The goal here is to position a car relative to a road map by comparing the driven trajectory to the road network. The speed V_k and yaw rate $\dot{\psi}_k$ in (56) are computed from the angular velocities of the nondriven wheels on one axle using rather simple geometrical relations. Dead reckoning (56) provides a profile that fits to the road network.

The measurement relation is in its simplest form a binary likelihood which is zero for all positions outside the roads and a non-zero constant otherwise. In this case, the DME is basically the prior that the vehicle is located on a road, and not a conventional physical sensor. See [72], [79] for more details and Fig. 15 for an illustration. More sophisticated applications use vibrations in wheel speeds and vehicle body as a DME. When a rough surface is detected, this DME can increase the likelihood for being outside the road. Likewise, if a forward-looking camera is present in the vehicle, this can be used to compute the likelihood that the front view resembles a road or if it is rather a nonmapped parking area or smaller private road.

The system is suitable as a support to satellite navigation in urban environments, in parking garages or tunnels or whenever satellite signals are likely to be obstructed. It is also a stand-alone solution to the navigation problem. Road databases covering complete continents are available from two main vendors (NavTech and TeleAtlas).

E. Aircraft Positioning using a Topographic Map

The principal approach here is quite similar to the UW positioning application and extends the one-dimensional example in Section IX to two dimensions.

A high-end IMU is used in an inertial navigation system (INS) which dead reckons the sensor data to speed V_k and yaw rate $\dot{\psi}_k$ in (56) with quite high accuracy. Still, absolute position support is needed to prevent long-term drifts.

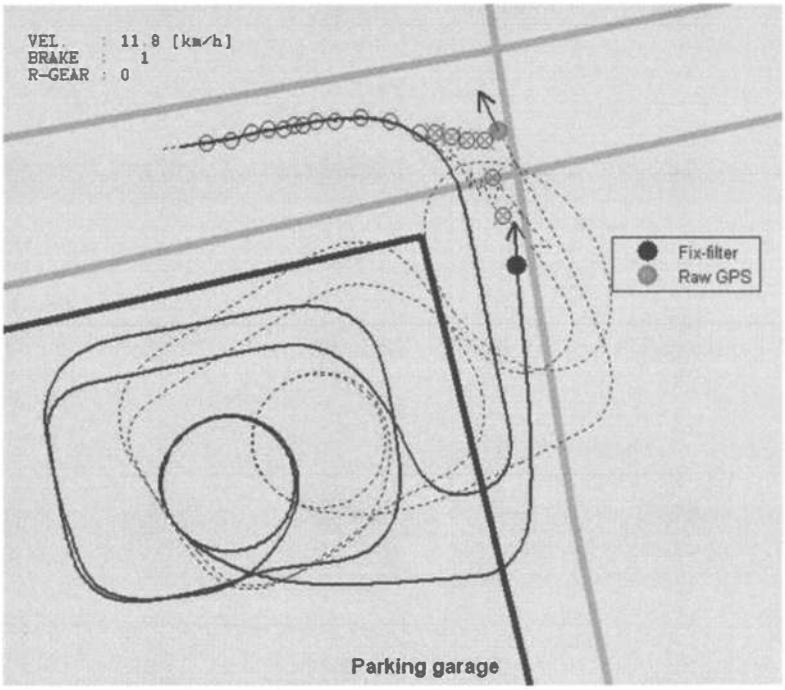

Fig. 17. Navigation of car in parking garage. Results for MPF when relative wheel radii and gyro offset are added to state vector. Two trajectories correspond to map-aided system and EKF with same state vector, but where GPS is used as position sensor. Since GPS gets several drop-outs before parking garage, dead-reckoning trajectory is incorrect; see [81].

The DME is a wide-lobe, downward looking radar that measures the distance to the ground. The absolute altitude is computed using the INS and a supporting barometric pressure sensor. Fig. 16 shows one example just before convergence to a unimodal filtering density.

Commercial databases of topographic information are available on land (but not below sea level), with a resolution of 50–200 m.

XI. MARGINALIZED PARTICLE FILTER APPLICATIONS

This section continues the applications in Section X with extended motion models where the MPF has been applied.

A. Underwater Positioning

Navigating an unmanned or manned UW vessel requires knowledge of the full three-dimensional position and orientation, not only the projection in a horizontal plane. That is, at least six states are needed. For control, also the velocity and angular velocities are needed, which directly implies at least a twelve-dimensional state vector. The PF cannot be assumed to perform well in such cases, and MPF is a promising approach [73].

B. Surface Positioning

There are two bottlenecks in the surface positioning PF that can be mitigated using the MPF. Both relate to the inertial measurements. First, the speed sensed by the log is the speed in water, not the speed over ground. Hence, the local water current is a parameter to include in the state vector. Second, the radar is strap down and measures relative to body orientation, which is not the same as the course ψ_k. The difference is the so called crab angle, which depends on currents and wind. This can also be included in the state vector. Further, there is in our demonstrator system [76] an unknown and time-varying offset in the reported radar angle, which has to be compensated for.

C. Vehicle Positioning

The bottleneck of the first generation of vehicle positioning PF is the assumption that the vehicle must be located on a road. As previously hinted one could use a small probability in the likelihood function for being off-road, but there is no real benefit for this without an accurate dead-reckoning ability, so reoccurrence on the road network can be predicted with high reliability.

The speed and yaw rate computed from the wheel angular velocity are limited by the insufficient knowledge of wheel radii. However, the deviation between actual and real wheel radii of the two wheels on one axle can be included in the state vector. Similarly, with a yaw rate sensor available (standard component in electronic stability programs (ESP) and navigation systems), the yaw rate drift can be included in the state vector. The point is that these parameters are accurately estimated when the vehicle

is on the road, and in the off-road mode, improved dead reckoning can be achieved. Tests in demonstrator vehicles have shown that the exit point from parking garages and parking areas are well estimated, and that shorter unmapped roads are not a problem; see Fig. 17.

D. Aircraft Positioning

The primary role of the terrain based navigation (TERNAV) module is to support the INS with absolute position information. The INS consists of an EKF based on a state vector with over 20 motion states and sensor bias parameters. The current bottleneck is the interface between TERNAV and INS. The reason is that TERNAV outputs a possibly multimodal position density, while the INS EKF expects a Gaussian observation. The natural idea is to integrate both TERNAV and INS into one filter. This gives a high-dimensional state vector, where one measurement (radar altitude) is very nonlinear. The MPF handles this elegantly, by essentially keeping the EKF from the existing INS and using the PF only for the radar altitude measurement.

The altitude radar gives a measurement outlier when the radar pulse is reflected in trees. Tests have validated that a Gaussian mixture where one mode has a positive mean models the real measurement error quite well. This Gaussian mixture distribution can be used in the likelihood computation, but such a distribution is in this case logically modeled by a binary Markov parameter, which is one in positions over forest and zero otherwise. In this way, the positive correlation between outliers is modeled, and a prior from ground-type information in the GIS can be incorporated. This example motivates the inclusion of discrete states in the model framework. See [67], [68] for the details.

XII. SUMMARY

This section summarizes practical experience from the applications in Sections X and XI with respect to the theorectical survey in Sections II and VIII.

A. Real-Time Issues

The PF has been applied to real data and implemented on hardware targeted for the application platforms. The sampling rate has been chosen in the order 1–2 Hz, and there is no problem in achieving real-time performance in any of the applications. Some remarkable cases follow.

1) The vehicle positioning PF was already implemented on a PDA using 15,000 particles in 2001; see [79].

2) The aircraft positioning PF was implemented in ADA and shown to satisfy real-time performance on the onboard computer in the Swedish fighter Gripen in the year 2000. Real-time performance was reached, despite the facts that a very large number of particles were used on a rather old computer.

B. Sampling Rates

The DME can in all cases deliver measurements much faster than the chosen sampling rate. However, faster sampling will introduce an unwanted correlation in the observations. This is due to the fact that the databases are quantized, so the platform should make a significant move between two measurement updates.

C. Implementation

Implementing and debugging the PF has not been a major issue. On the contrary, students and nonexperts have faced fewer problems with the PF than for similar projects involving the EKF. In many cases, they obtained deep intuition for including nontrivial but ad hoc modifications. There are today several hardware solutions reported in literature, where the parallel structure of the PF algorithms can be utilized efficiently. For instance, an FPGA implementation is reported in [82], and on a general purpose graphics processing unit (GPGPU) in [83]. Analog hardware can further be used to speed up function evaluations [61].

D. Dithering

Both the process noise and measurement noise distributions need some dithering (increased covariance). Dithering the process noise is a well-known method to mitigate the sample depletion problem [15]. Dithering the measurement noise is a good way to mitigate the effects of outliers and to robustify the PF in general. One simple and still very effective method to mitigate sample depletion is to introduce a lower bound on the likelihood. This lower bound was first introduced more or less ad hoc. However, recently this algorithm modification has been justified more rigorously. In proving that the PF converges for unbounded functions, like the state x_k itself, it is sufficient to have a lower bound on the likelihood; see [57] for details.

E. Number of Particles

The number of particles is chosen to be the quite large to achieve good transient behaviour in the start-up phase and to increase robustness. However, it has been concluded that in the normal operational mode the number of particles can be decreased

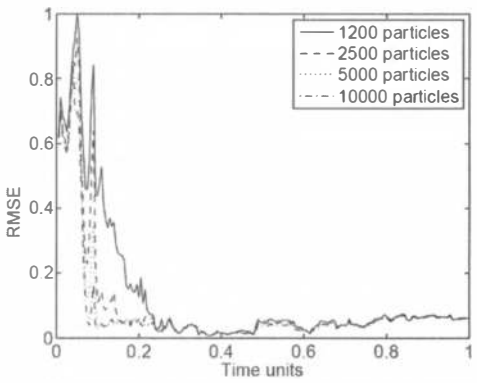

Fig. 18. RMSE performance for aircraft terrain navigation as function of number of particles.

substantially (typically a factor of ten). Fig. 18 shows experimental results for the terrain navigation application. The transient improves when going from $N = 1200$ to $N = 2500$, but using more particles give no noticable improvement after convergence.

A real-time implementation should be designed for the worst case. However, using an adaptive sampling interval T and number of particles N is one option. The idea is to use a longer sampling interval and more particles initially, and when the PF has converged to a few distinct modes, T and N can be decreased in such a way that the complexity N/T is constant.

F. Choosing the Proposal Density

The standard SIR-PF works fine for an initial design. However, the maps contain rather detailed information about position and can be considered as state constraints in the limit. In such high signal-to-noise applications, the standard proposal density used in the SIR-PF is not particularly efficient. An alternative, that typically improves the performance, is to use the information available in the next measurement already in the state prediction step. Note that the proposal in its most general form includes the next observation. Consider for instance positioning based on road maps. In standard SIR-PF, the next positions are randomized around the predicted position according to the state noise, which is required to obtain diversity. Almost all of these new particles are outside the road network, and will not survive the resampling step. Obviously this is a waste of particles. By looking at how the roads are located locally around the predicted position, a much more clever process noise can be computed, and the particles explore the road network much more efficiently.

G. Divergence Monitoring

Divergence monitoring is fundamental for real-time implementations to achieve the required level of integrity. After divergence, the particles do not reflect the true state distribution and there is no mechanism that automatically stabilizes the PF. Hence, divergence monitoring has to be performed in parallel with the actual PF code, and when divergence is detected, the PF is reinitialized.

One indicator of particle depletion is the effective number of samples N_{eff}, used in the PF. This number monitors the amount of particles that significantly contribute to the posterior, and it is computed from the normalized weights. However, the unnormalized likelihoods are a more logical choice for monitoring. Standard hypothesis tests can be applied for testing if the particle predictions represent the likelihood distribution.

Another approach is to use parallel PFs interleaved in time. The requirement is that the sensors are faster than the chosen sampling rate in the PF. The PFs then use different time delays in the sensor observations.

The reinitialization procedure issued when divergence is detected is quite application dependent. The general idea is to use a very diffuse prior, or to infer external information. For the vehicle positioning application in [79], a cellular phone operator took part in the demonstrator, and cell information was used as a new prior for the PF in case of occasional divergence.

H. Performance Bounds

For all four GPS-free applications, the positioning performance is in the order of 10 m root mean square error (RMSE), which is comparable to GPS performance. Further, the performance of the PF has been shown to be close to the CRLB for a variety of examined trajectories. In Fig. 19 two examples of performance evaluations in terms of the RMSE are depicted. On the left hand side the position RMSE and CRLB are shown for the UW application, and on the right hand side the horizontal position error is provided for the aircraft application.

I. Particle Filter in Embedded Systems

The primary application is to output position information to the operator. However, in all cases there have been decision and control applications built on the position information, which indicates that the PF is a powerful software component in embedded systems as follows.

1) UW positioning: Here, the entire mission relies on the position, so path planning and trajectory control are based on the output from the PF. Note that there is hardly any alternative below sea level, where no satellites are reachable, and deploying infrastructure (sonar buoys) is quite expensive.

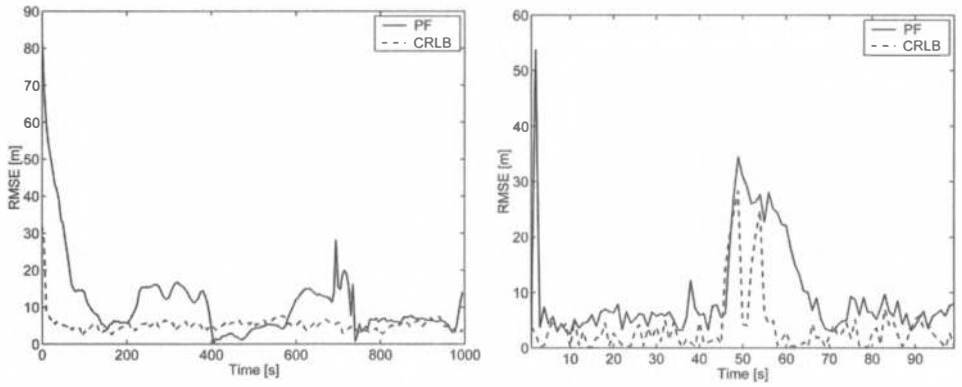

Fig. 19. Position RMSE for UW (left) and surface (right) applications compared to CRLB.

2) Surface positioning: Differentiating radar detections from shore, clutter, and other ships is an essential association task in the PF. It is a natural extension to integrate a collision avoidance system in such an application, as illustrated in a sea chart snapshot in Fig. 14.

3) Vehicle positioning: The PF position was also used in a complete voice-controlled navigation system with dynamic route optimization; see Fig. 15.

4) Aircraft navigation: The position from the PF is primarily used as a supporting sensor in the INS, whose position is a refined version of the PF output.

J. Marginalized Particle Filtering

Finally, the MPF offers a scalable extension of the PF in all applications surveyed here and many others. MPF is applicable for instance in the following localization, navigation, and tracking problems:

1) three-dimensional position spaces,
2) motion models with velocity and acceleration states,
3) augmenting the state vector with unknown nuisance parameters as sensor offsets and drifts.

The FastSLAM algorithm is state of the art; see [24]. This algorithm applies MPF to the SLAM problem. FastSLAM has been applied to applications where thousands of two-dimensional landmark features are marginalized out from a three dimensional motion state. Further, in [84] a double marginalization process was employed to handle hundreds of landmark features and a 24-dimensional state vector for three-dimensional navigation of an unmanned aerial vehicle in an unknown environment.

ACKNOWLEDGMENT

This survey is the result of various research projects over the last ten years, and the author is greatly indebted to the following persons who have completed a Ph.D. with a focus on particle filtering: Niclas Bergman, Rickard Karlsson, Thomas Schön, Gustaf Hendeby, David Törnqvist, and Per-Johan Nordlund. There are also numerous current graduate students and post-docs, and more than 50 master students who have contributed indirectly. This survey is very much influenced by their work.

REFERENCES

[1] Kalman, R.
A new approach to linear filtering and prediction problems.
Transactions of Journal Basic Engineering, ASME Series D, **82** (1960), 35–45.

[2] Kailath, T., Sayed, A., and Hassibi, B.
Linear Estimation (Information and System Sciences Series).
Upper Saddle River, NJ: Prentice-Hall, 2000.

[3] Smith, L. A. M. G. L., and Schmidt, S. F.
Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle.
NASA, Technical Report TR R-135, 1962.

[4] Schmidt, S.
Application of state-space methods to navigation problems.
Advances in Control Systems, (1966), 293–340.

[5] Julier, S. J., Uhlmann, J. K., and Durrant-Whyte, H. F.
A new approach for filtering nonlinear systems.
In *Proceedings of the American Control Conference*, vol. 3, 1995, 1628–1632.

[6] Julier, S. J., and Uhlmann, J. K.
Unscented filtering and nonlinear estimation.
Proceedings of IEEE, **92**, 3 (Mar. 2004), 401–422.

[7] Norgaard, M., Poulsen, N., and Ravn, O.
New developments in state estimation of nonlinear systems.
Automatica, **36** (2000), 1627–1638.

[8] Arasaratnam, I., Haykin, S., and Elliot, R.
Discrete-time nonlinear filtering algorithms using Gauss-Hermite quadrature.
Proceedings of IEEE, **95** (2007), 953.

[9] Alspach, D., and Sorenson, H.
Nonlinear Bayesian estimation using Gaussian sum approximation.
IEEE Transactions on Automatic Control, **17** (1972), 439–448.

[10] Kramer, S., and Sorenson, H.
Recursive Bayesian estimation using piece-wise constant approximations.
Automatica, **24** (1988), 789–801.

[11] Hammersley, J., and Morton, K.
Poor man's Monte Carlo.
Journal of the Royal Statistical Society, Series B, **16** (1954), 23.

[12] Rosenbluth, M., and Rosenbluth, A.
Monte Carlo calculation of the average extension of molecular chains.
Journal of Chemical Physics, **23** (1956), 590.

[13] Akashi, H., and Kumamoto, H.
Random sampling approach to state estimation in switching environment.
Automation, **13**, 1977, 429.

[14] Handshin, J. Monte Carlo techniques for prediction and filtering of nonlinear stochastic processes. Automatica, **6**, 1970, 555.

[15] Gordon, N., Salmond, D., and Smith, A.
A novel approach to nonlinear/non-Gaussian Bayesian state estimation.
In *IEE Proceedings on Radar and Signal Processing*, vol. 140, 1993, 107–113.

[16] Kitagawa, G.
Monte Carlo filter and smoother for non-Gaussian nonlinear state space models.
Journal of Computational and Graphical Statistics, **5**, 1 (1996), 1–25.

[17] Isard, M., and Blake, A.
Condensation—Conditional density propagation for visual tracking.
International Journal of Computer Vision, **29**, 1 (1998), 5–28.

[18] Doucet, A., de Freitas, N., and Gordon, N., (Eds.)
Sequential Monte Carlo Methods in Practice.
New York: Springer-Verlag, 2001.

[19] Liu, J., and Chen, R.
Sequential Monte Carlo methods for dynamic systems.
Journal of the American Statistical Association, **93** (1998).

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

[21] Djuric, P., Kotecha, J., Zhang, J., Huang, Y., Ghirmai, T., Bugallo, M., and Miguez, J.
Particle filtering.
IEEE Signal Processing Magazine, **20** (2003), 19.

[22] Cappé, O., Godsill, S., and Moulines, E.
An overview of existing methods and recent advances in sequential Monte Carlo.
IEEE Proceedings, **95** (2007), 899.

[23] Ristic, B., Arulampalam, S., and Gordon, N.
Beyond the Kalman filter: Particle filters for tracking applications.
London: Artech House, 2004.

[24] Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B.
FastSLAM a factored solution to the simultaneous localization and mapping problem.
Presented at the AAAI National Conference on Artificial Intelligence, Edmonton, Canada, 2002.

[25] Durrant-Whyte, H., and Bailey, T.
Simultaneous localization and mapping (SLAM): Part I.
IEEE Robotics & Automation Magazine, **13**, 2 (June 2006), 99–110.

[26] Bailey, T., and Durrant-Whyte, H.
Simultaneous localization and mapping (SLAM): Part II.
IEEE Robotics & Automation Magazine, **13**, 3 (Sept. 2006), 108–117.

[27] Thrun, S., Burgard, W., and Fox, D.
Probabilistic Robotics.
Cambridge, MA: MIT Press, 2005.

[28] Rathi, Y., Vaswani, N., Tannenbaum, A., and Yezzi, A.
Tracking deforming objects using particle filtering for geometric active contours.
IEEE Transactions on Pattern Analysis and Machine Intelligence, **29**, 8 (2007), 1470–1475.

[29] Rathi, Y., Vaswani, N., and Tannenbaum, A.
A generic framework for tracking using particle filter with dynamic shape prior.
IEEE Transactions on Image Processing, **16**, 5 (2007), 1370–1382.

[30] Lu, W-L., Okuma, K., and Little, J.
Tracking and recognizing actions of multiple hockey players using the boosted particle filter.
Image and Vision Computing, **27** (2009), 189–205.

[31] Cevher, V., Sankaranarayanan, A., McClellan, J., and Chellappa, R.
Target tracking using a joint acoustic video system.
IEEE Transactions on Multimedia, **9**, 4 (2007), 715–727.

[32] Bar-Shalom, Y., and Fortmann, T.
Tracking and Data Association, vol. 179 (Mathematics in Science and Engineering Series).
New York: Academic Press, 1988.

[33] Gustafsson, F., and Hendeby, G.
On nonlinear transformations of stochastic variables and its application to nonlinear filtering.
Presented at the IEEE International Conference on Acoustics, Speech, and Signal Processing, Las Vegas, NV, 2008.

[34] Gustafsson, F.
Adaptive Filtering and Change Detection.
New York: Wiley, 2001.

[35] Jazwinsky, A.
Stochastic Process and Filtering Theory, vol. 64 (Mathematics in Science and Engineering Series).
New York: Academic Press, 1970.

[36] Van Trees, H.
Detection, Estimation and Modulation Theory.
New York: Wiley, 1971.

[37] Robert, C. P., and Casella, G.
Monte Carlo Statistical Methods, (Springer Texts in Statistics Series).
New York: Springer, 1999.

[38] Klaas, M.
Toward practical n2 Monte Carlo: The marginal particle filter.
Uncertainty in Artificial Intelligence, (2005).

[39] Poyiadjis, G., Doucet, A., and Singh, S.
Maximum likelihood parameter estimation in general state-space models using particle methods.
Presented at the Joint Statistical Meeting, Minneapolis, MN, 2005.

[40] Poyiadjis, G., Doucet, A., and Singh, S.
Maximum likelihood parameter estimation using particle methods.
Presented at the IEEE Conference on Acoustics, Speech and Signal Processing, 2006.

[41] Martinez-Cantin, R., de Freitas, N., and Castellanos, J.
Analysis of particle methods for simultaneous robot localization and mapping and a new algorithm: Marginal-slam.
Presented at the IEEE International Conference on Robotics and Automation, Rome, Italy, 2007.

[42] Sing, S., Kantas, N., Vo, B., Doucet, A., and Evans, R.
Simulation-based optimal sensor scheduling with application to observer trajectory planning.
Automatica, **43** (2007), 817–830.

[43] Doucet, A., Godsill, S., and Andrieu, C.
On sequential simulation-based methods for Bayesian filtering.
Statistics and Computing, **10**, 3 (2000), 197–208.

[44] Kotecha, J., and Djuric, P.
Gaussian particle filtering.
IEEE Transactions on Signal Processing, **51** (2003), 2592.

[45] Kotecha, J., and Djuric, P.
Gaussian sum particle filtering.
IEEE Transactions on Signal Processing, **51** (2003), 2602.

[46] Kong, A., Liu, J. S., and Wong, W. H.
Sequential imputations and Bayesian missing data problems.
Journal of American Statistical Association, **89**, 425 (1994), 278–288.

[47] Liu, J.
Metropolized independent sampling with comparison to rejection sampling and importance sampling.
Statistics and Computing, **6** (1996), 113–119.

[48] Fearnhead, P.
Sequential Monte Carlo methods in filter theory.
Ph.D. dissertation, University of Oxford, UK, 1998.

[49] Doucet, A., Gordon, N., and Krishnamurthy, V.
Particle filters for state estimation of jump Markov linear systems.
IEEE Transactions on Signal Processing, **49**, 3 (2001), 613–624.

[50] Pitt, M., and Shephard, N.
Filtering via simulation: Auxiliary particle filters.
Journal of the American Statistical Association, **94**, 446 (June 1999), 590–599

[51] Doucet, A., Briers, M., and Sénécal, S.
Efficient block sampling strategies for sequential Monte Carlo methods.
Journal of Computational and Graphical Statistics, **15**, 3 (2006), 1–19.

[52] Thrun, S., Fox, D., Dellaert, F., and Burgard, W.
Particle filters for mobile robot localization.
In A. Doucet, N. de Freitas, and N. Gordon, (Eds.), *Sequential Monte Carlo Methods in Practice*, New York: Springer-Verlag, 2001.

[53] Johansen, A., and Doucet, A.
A note on auxiliary particle filters.
Statistics & Probability Letters, **78**, 12 (2008), 1498–1504.

[54] Crisan, D., and Doucet, A.
Convergence of sequential Monte Carlo methods.
Signal Processing Group, Department of Engineering, University of Cambridge, Technical Report CUED/F-INFENG/TR381, 2000.

[55] Moral, P. D.
Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications.
New York: Springer, 2004.

[56] Crisan, D., and Doucet, A.
A survey of convergence results on particle filtering methods for practitioners.
IEEE Transactions on Signal Processing, **50**, 3 (2002), 736–746.

[57] Hu, X., Schön, T., and Ljung, L.
A basic convergence result for particle filtering.
IEEE Transactions on Signal Processing, **56**, 4 (Apr. 2008), 1337–1348.

[58] Hendeby, G.
Performance and implementation aspects of nonlinear filtering.
Dissertation No. 1161, Linköping University, Sweden, 2008.

[59] Bergman, A. D. N., and Gordon, N.
Optimal estimation and Cramer-Rao bounds for partial non-Gaussian state-space model.
Annals of the Institute of Statistical Mathematics, **52**, 1 (2001), 97–112.

[60] Ripley, B.
Stochastic Simulation.
Hoboken, NJ: Wiley, 1988.

[61] Velmurugan, R., Subramanian, S., Cevher, V., Abramson, D., Odame, K., Gray, J., Lo, H-J., McClellan, M., and Anderson, D.
On low-power analog implementation of particle filters for target tracking.
In *Proceedings of the European Signal Processing Conference* (EUSIPCO), 2006.

[62] Coppersmith, D., and Winograd, S.
Matrix multiplication via arithmetic progressions.
Journal of Symbolic Computation, **9** (1990), 251–280.

[63] Doucet, A., Godsill, S. J., and Andrieu, C.
On sequential Monte Carlo sampling methods for Bayesian filtering.
Statistics and Computing, **10**, 3 (2000), 197–208.

[64] Casella, G., and Robert, C. P.
Rao-Blackwellisation of sampling schemes.
Biometrika, **83**, 1 (1996), 81–94.

[65] Chen, R., and Liu, J. S.
Mixture Kalman filters.
Journal of the Royal Statistical Society, **62**, 3 (2000), 493–508.

[66] Andrieu, C., and Doucet, A.
Particle filtering for partially observed Gaussian state space models.
Journal of the Royal Statistical Society, **64**, 4 (2002), 827–836.

[67] Schön, T., Gustafsson, F., and Nordlund, P.
Marginalized particle filters for nonlinear state-space models.
IEEE Transactions on Signal Processing, **53** (2005), 2279–2289.

[68] Nordlund, P-J., and Gustafsson, F.
Marginalized particle filter for accurate and reliable terrain-aided navigation.
IEEE Transactions on Aerospace and Electronic Systems, **35**, 3 (2008).

[69] Hendeby, G., Karlsson, R., and Gustafsson, F.
A new formulation of the Rao-Blackwellized particle filter.
In *Proceedings of IEEE Workshop on Statistical Signal Processing*, Madison, WI, Aug. 2007.

[70] Karlsson, R., Schön, T., and Gustafsson, F.
Complexity analysis of the marginalized particle filter.
IEEE Transactions on Signal Processing, **53** (2005), 4408–4411.

[71] Šimandl, M., Královec, J., and Söderström, T.
Advanced point-mass method for nonlinear state estimation.
Automatica, **42**, 7 (July 2006), 1133–1145.

[72] Gustafsson, F., Gunnarsson, F., Bergman, N., Forssell, U., Jansson, J., Karlsson, R., and Nordlund, P-J.
Particle filters for positioning, navigation and tracking.
IEEE Transactions on Signal Processing, **50**, 2 (Feb. 2002), 425–437.

[73] Karlsson, R., and Gustafsson, F.
Bayesian surface and underwater navigation.
IEEE Transactions on Signal Processing, **54**, 11 (2006), 4204–4213.

[74] Fauske, K., Gustafsson, F., and Herenaes, O.
Estimation of AUV dynamics for sensor fusion.
In *Proceedings of Fusion 2007*, Quebec, Canada, July, 2007.

[75] Karlsson, T.
Terrain aided underwater navigation using Bayesian statistics.
Department of Electrical Engineering, Linköping University, S-581 83, Linköping, Sweden, Master's Thesis LiTH-ISY-EX-3292, 2002.

[76] Dahlin, M., and Mahl, S.
Radar distance positioning system—With a particle filter approach.
Department of Electrical Engineering, Linköping University, Master's Thesis LiTH-ISY EX-3998.

[77] Rönnebjerg, A.
A tracking and collision warning system for maritime applications.
Department of Electrical Engineering, Linköping University, S-58183 Linköping, Sweden, Master's Thesis LiTH-ISY-EX-3709, 2005, in Swedish.

[78] Lo, S., Peterson, B., and Enge, P.
Loran data modulation: A primer (AESS Tutorial IV).
IEEE Aerospace and Electronic Systems Magazine, **22** (2007), 31–51.

[79] Forssell, U., Hall, P., Ahlqvist, S., and Gustafsson, F.
Novel map-aided positioning system.
In *Proceedings of FISITA*, no. F02-1131, Helsinki, 2002.

[80] Hall, P.
A Bayesian approach to map-aided vehicle positioning.
Department of Electric Engineering, Linköping University, S-581 83 Linköping, Sweden, Master's Thesis LiTH-ISY-EX-3104, 2001, in Swedish.

[81] Kronander, J.
Robust vehicle positioning: Integration of GPS and motion sensors.
Department of Electrical Engineering, Linköping University, S-58183 Linköping, Sweden, Master's Thesis LiTH-ISY-EX-3578, 2003.

[82] Athalye, A.
Design and implementation of reconfigurable hardware for real-time particle filtering.
Ph.D. dissertation, Stody Brook University, 2007.

[83] Hendeby, G., Hol, J. D., Karlsson, R., and Gustafsson, F.
A graphics processing unit implementation of the particle filter.
In *Proceedings of the European Signal Processing Conference* (EUSIPCO), Pozna'n, Poland, Sept. 2007.

[84] Karlsson, R., Schön, T., Törnqvist, D., Conte, G., and Gustafsson, F.
Utilizing model structure for efficient simultaneous localization and mapping for a UAV application.
In *Proceedings of IEEE Aerospace Conference*, Big Sky, MT, 2008.

Fredrik Gustafsson received the M.Sc. degree in electrical engineering 1988 and the Ph.D. degree in automatic control, 1992, both from Linkoping University.

During 1992–1999 he held various positions in automatic control, and 1999–2005 he had a professorship in communication systems. He has been a professor in the Sensor Informatics at Department of Electrical Engineering, Linköping University, since 2005. His research interests are in stochastic signal processing, adaptive filtering, and change detection, with applications to communication, vehicular, airborne, and audio systems. His work in the sensor fusion area involves design and implementation of nonlinear filtering algorithms for localization, navigation and tracking of all kind of platforms, including cars, aircraft, spacecraft, UAVs, surface and underwater vessels, cell phones, and film cameras for augmented reality. He is a cofounder of the companies NIRA Dynamics and Softube, developing signal processing software solutions for automotive and music industry, respectively.

He was an associate editor for *IEEE Transactions of Signal Processing* 2000–2006 and is currently associate editor for EURASIP *Journal on Applied Signal Processing* and *International Journal of Navigation and Observation.* In 2004, he was awarded the Arnberg prize by the Royal Swedish Academy of Science (KVA) and in 2007 he was elected member of the Royal Academy of Engineering Sciences (IVA).