The Particle Filter in Practice

Thomas B. Schön and Fredrik Gustafsson
Division of Automatic Control
Linköping University
SE–58183 Linköping, Sweden
{schon,fredrik}@isy.liu.se

Rickard Karlsson
NIRA Dynamics AB
Teknikringen 6
SE–58330 Linköping, Sweden
rickard.karlsson@niradynamics.se

August 21, 2009

1 Introduction

Positioning of moving platforms has been a *technical driver* for real-time applications of the particle filter (PF) in both the signal processing and the robotics communities. For this reason, we will spend some time to explain several such applications in detail, and to summarize the experiences of using the PF in practice. The applications concern positioning of underwater vessels, surface ships, cars, and aircraft using geographical information systems containing a database with features of the surrounding. In the robotics community, the PF has been developed into one of the main algorithms (FastSLAM) for solving the simultaneous localization and mapping (SLAM) problem. This can be seen as an extension to the aforementioned applications, where the features in the geographical information system are dynamically detected and updated on the fly.

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. However, the PF is *not* practically useful when extending the models to more realistic cases with

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

A *technical enabler* for such applications is the Rao-Blackwellized particle filter also referred to as the marginalized particle filter (MPF). It allows 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 Kalman filter is used for linear Gaussian subsystems of the complete model. For subsystems that are almost linear and only slightly non-Gaussian, the extended Kalman filter (EKF) or the unscented Kalman filter (UKF) can be applied to reduce the burden of the PF. This latter is supposed to be a quite general case,

since it is hard to find high-dimensional models where all states undergo complex nonlinear transformation with non-Gaussian noise disturbances.

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 Kalman filter. The need for the MPF in the list of applications will be motivated by examples and experience from practice.

The subsequent section discusses four different applications, where the PF and the MPF are primarily used to compute position estimates. Due to the importance of MPF when it comes to applications, Section 3 is devoted to the MPF. In Section 4 the positioning problem is extended to the case where there is no map available, i.e., the SLAM problem. Finally, the conclusions are given in Section 5.

2 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 particle filter can be used in all of them. In fact, successful applications of a PF are described in literature which are all based on the same state-space model and similar measurement equations.

2.1 Model Framework

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

$$x_t = (X_t, Y_t, \psi_t)^T, \tag{1a}$$

$$u_t = (v_t, \dot{\psi}_t)^T, \tag{1b}$$

$$X_{t+1} = X_t + T v_t \cos(\psi_t), \tag{1c}$$

$$Y_{t+1} = X_t + T v_t \sin(\psi_t), \tag{1d}$$

$$\psi_{t+1} = \psi_t + T\dot{\psi}_t, \tag{1e}$$

$$y_t = h(x_t) + e_t. \tag{1f}$$

Here, X_t, Y_t denote the Cartesian position, ψ_t the course or heading, T is the sampling interval, v_t is the speed and $\dot{\psi}_t$ the yaw rate. The inertial signals v_t and $\dot{\psi}_t$ 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 will be described in more detail in the sequel. The measurement relation is based on a distance measuring equipment and a geographical information system (GIS). Both the distance measurement equipment 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 or KF bank are 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 not available or to provide insufficient navigation integrity. First, the inertial inputs, the distance measurement equipment

Figure 1: *The left plot is an illustration of an underwater vessel measuring distance d_1 to sea bottom, and absolute depth d_2. The sum $d = d_1 + d_2$ is compared to a bottom map as illustrated with the contours in the plot to the right. The particle cloud illustrates a snapshot of the PF from a known validation trajectory.*

and GIS for the four applications are described. Then, some conclusions from practice are summarized. Finally, different ways to augment the state vector are described for each application, and conclusions from applying the MPF are drawn. 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.

2.2 Applications of the PF

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

2.2.1 Underwater Positioning using a Topographic Map

The speed v_t and yaw rate $\dot{\psi}_t$ 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 [12] for one simple linear model. An alternative is to use inertial measurement units for measuring and computing speed and yaw rate.

A sonar is measuring the distance d_1 to the sea floor. The depth of the platform itself d_2 can be computed from pressure sensors, or from a sonar directed up-wards. By adding these distances, the sea depth at the position X_t, Y_t 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 [20]. Figure 1 provides an illustration. Detailed bottom sea charts are so far proprietary military information, and most applications are also military. However, oil companies are starting to use unmanned underwater vessels for exploring the sea and oil platforms, and in this way building up their own maps.

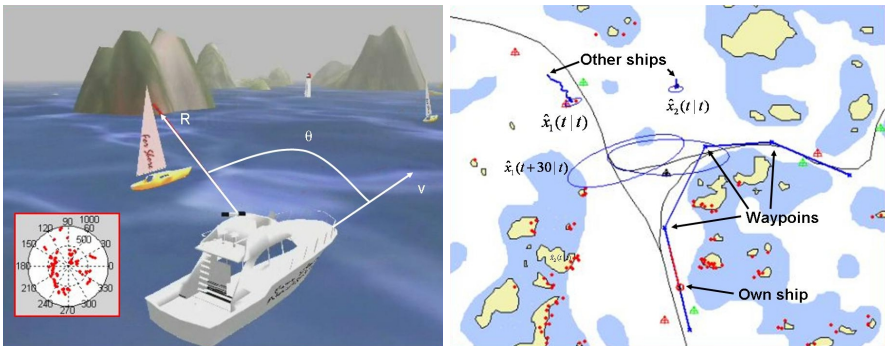


Figure 2: *The rotating radar returns detections of range R at body angle θ. The result of one radar revolution is conventionally displayed in polar coordinates as illustrated. Comparing the (R, θ) detections to a sea chart as shown to the right, the position and course are estimated by the PF. When correctly estimated, the radar overlay principle can be used for visual validation as also illustrated in the sea chart. The PF has to distinguish radar reflections from shore with clutter and other ships. The latter can be used for conventional target tracking algorithms, and collision avoidance algorithms, as also illustrated to the right.*

2.2.2 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, the standard vectorized sea charts (for instance the S-57 standard) contain a commercially available world-wide map.

The idea is to use the radar as distance measurement equipment and compare the detections with the shore profile, which is known from the sea chart conditioned on the position X_t, Y_t and course ψ_t (indeed the ship orientation, but more on this later), see [20]. The likelihood function models the radar error, but must also take clutter (false detections) and other ships into account.

The left hand part of Figure 2 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 backup and support system to satellite navigation to increase integrity. The reason is accidents and incidents caused by technical problems with the satellite navigation system, and the risk of accidental or deliberate jamming. The PF solution here is one candidate, since it is not sensitive to jamming nor does it require any extra infrastructure.

2.2.3 Vehicle Positioning using a Road Map

The speed v_t and yaw rate $\dot{\psi}_t$ are here computed from the angular velocities of the non-driven wheels on one axle, using rather simple geometrical relations.

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 distance measurement equipment is basically the prior that the

Figure 3: *Left: Example of multimodal posterior represented by a number of distinct particle clouds from NIRA Dynamics navigation system. This is caused by the regular road pattern and will be resolved after a sufficiently long sequence of turns. Right: PF in an embedded navigation solution run in real-time on a pocket PC.*

vehicle is located on a road, and not a conventional physical sensor. See [13, 15] for more details, and Figure 3 for an illustration. More sophisticated applications use vibrations in wheel speeds and vehicle body as a distance measurement equipment. When a rough surface is detected, this distance measurement equipment an 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 non-mapped parking area or a 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).

2.2.4 Aircraft Positioning using a Topographic Map

The principal approach here is quite similar to the underwater positioning. A high-end IMU is used in an inertial navigation system (INS) which dead-reckons the sensor data to position and orientation with quite high accuracy. Still, absolute position support is needed to prevent long-term drifts.

The distance measurement equipment is a wide-lobe down-ward looking radar that measures the distance to the ground. The absolute altitude is computed using the INS and a supporting barometric pressure sensor. For more information about this application, see e.g., [28, 25]. Figure 4 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 meters.

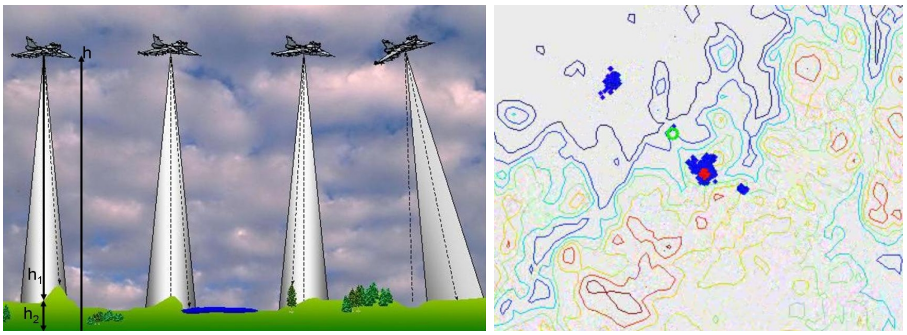


Figure 4: *The left figure is an illustration of an aircraft measuring distance h_1 to ground. The on-board baro-altitude supported INS system provides absolute altitude over sea level h, and the difference $h_2 = h - h_1$ is compared to a topographical map. The right plot shows a snapshot of the PF particle cloud, just after the aircraft has left the sea in the upper left corner. There are three distinct modes, where the one corresponding to the correct position dominates.*

2.3 Summary of Practical Experiences

2.3.1 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 to achieve real-time performance in any of the applications. Some remarkable examples:

- The vehicle positioning PF was implemented on a pocket computer using 15000 particles already in 2001, [13].
- The aircraft positioning PF was implemented in the programming language ADA used in the aircraft computer and shown to satisfy real-time performance on the on-board 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, and that a rather old computer was used.

2.3.2 Sampling Rates

The distance measurement equipment 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.

2.3.3 Implementation

Implementing and debugging the PF has not been a major issue. On the contrary, students and non-experts have faced less problems with the PF than for similar projects involving the EKF. In many cases, they obtained deep intuition for including non-trivial but *ad-hoc* modifications.

2.3.4 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 impoverishment problem [14]. 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 impoverishment 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 particle filter converges for unbounded functions, like the state x_t itself, it is sufficient to have a lower bound on the likelihood, see [19] for details.

2.3.5 Number of Particles

The number of particles is chosen quite large to achieve good transient behavior 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 substantially (typically a factor of ten). A real-time implementation should be designed for the worst case. However, adapting the sampling time T and the 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.

2.3.6 Choosing the Proposal Density

The standard sampling importance resampling (SIR) PF works fine for an initial design. However, the maps contain rather detailed information about position, and can in the limit be considered as state constraints. In such high signal-to-noise applications, the standard proposal density $p(x_{t+1}|x_t)$ 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 general has the form

$$x_{t+1}^{(i)} \sim q(x_{t+1}|x_{1:t}, y_{1:t+1}), \qquad i = 1, \ldots, N \tag{2}$$

so this is perfectly fine. Consider for instance positioning based on road maps. In standard SIR PF, the next positions are randomized around the predicted position according to the process 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 in the map 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.

2.3.7 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 particle filter. Hence, divergence monitoring has to be performed in

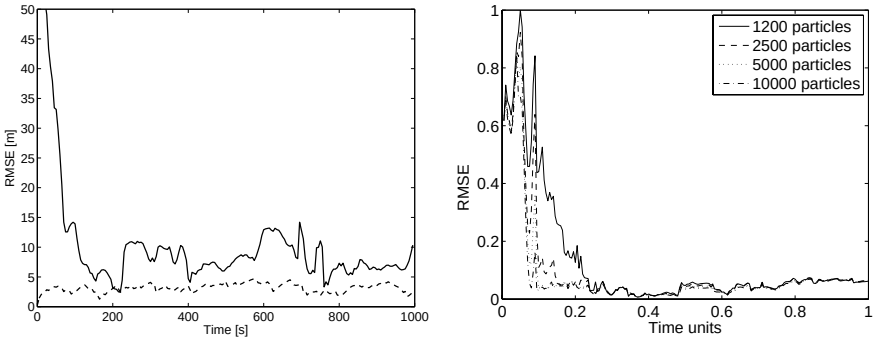


Figure 5: *Left: The position RMSE for the underwater application from the PF (solid line) using the experimental test data together with the parametric CRLB (dashed line) as the EKF solution around the true trajectory. Right: Horizontal position error from the aircraft application as a function of time for different number of particles. Note that the scale has been normalized for confidentiality reasons.*

parallel with the actual PF code, and when divergence is detected, the PF is re-initialized.

One indicator of particle impoverishment is the efficient 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 un-normalized likelihoods are a more logical choice for monitoring. Standard hypothesis tests can be applied for testing whether the particle predictions represent the likelihood distribution.

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

The re-initialization 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. In the vehicle positioning application [13], 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.

2.3.8 Performance Bounds

For all four GPS-free applications the positioning performance is in the order of ten meter root mean square error (RMSE). Further, the performance has been shown to be close to the Cramér-Rao lower bound (CRLB) for a variety of examined trajectories. In Figure 5 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 underwater application and on the right hand side the horizontal position error is provided for the aircraft application.

2.3.9 PF 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:

- Underwater 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 no alternative below sea level, where no satellites are reachable, and deploying infrastructure (sonar buoys) is quite expensive.
- 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 Figure 2.
- Vehicle positioning: The PF position was also used in a complete voice controlled navigation system with dynamic route optimization, see Figure 3.
- 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.

2.4 Applications of the MPF

2.4.1 Underwater Positioning

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

2.4.2 Surface Positioning

There are two bottlenecks in the surface positioning PF that can be mitigated using the MPF. Both relates 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_t. 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 an unknown and time-varying offset in the reported radar angle, which has to be compensated for.

2.4.3 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 re-occurrence on the road network can be predicted with high reliability.

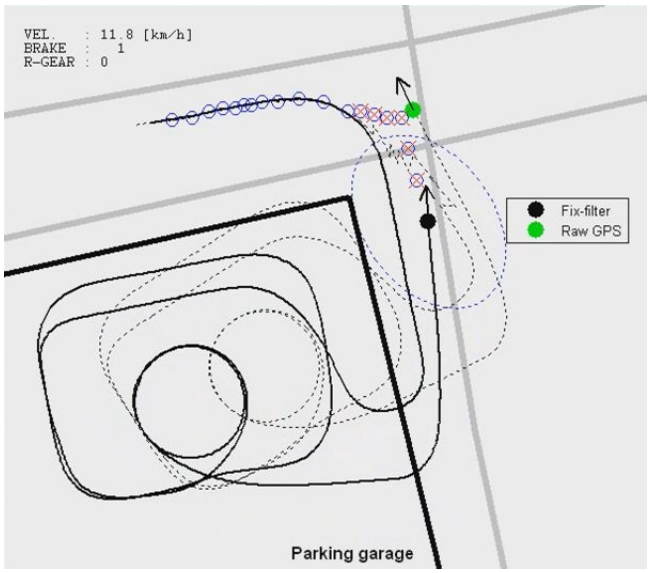


Figure 6: *Navigation of a car in a parking garage. Results for the MPF when relative wheel radii and gyro offset are added to the state vector. The two trajectories correspond to the map-aided system and an EKF with the same state vector, but where GPS is used as position sensor. Since the GPS gets several drop-outs before the parking garage, the dead-reckoning trajectory is incorrect.*

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 on 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 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, accurate 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 Figure 6.

2.4.4 Aircraft Positioning

The primary role of the terrain based navigation module is to support the inertial navigation system (INS) with absolute position information. The INS consists of an extended Kalman filter based on a state vector with over 20 motion states and sensor bias parameters. The current bottleneck is the interface between the terrain navigation module and the INS. The reason is that the terrain navigation module outputs a possibly multimodal position density, while the EKF used for INS expects a Gaussian observation. The natural idea is to integrate both terrain navigation and INS into one filter. This results in 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 re-

flected in trees. Tests have validated that a Gaussian mixture where on mode as 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 [28, 26] for the details.

3 Marginalized Particle Filter

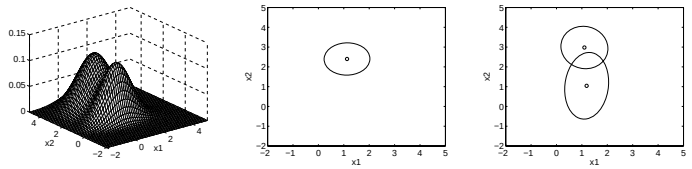
If there is a linear Gaussian substructure available in the model this can often be exploited to derive a better estimator. The basic idea is to estimate the "nonlinear" states using the particle filter and the conditionally linear Gaussian states using the Kalman filter. The resulting filter is called the MPF or the Rao-Blackwellized particle filter, and it has been known for quite some time, see e.g., [8, 5, 10, 6, 2, 28, 25].

In Section 3.1 the representation used in the MPF is illustrated by comparing it to several well-known estimators. The model structure is introduced in Section 3.2 and in Section 3.3 the MPF algorithm is given. Finally, the variance reduction property and the computational complexity of MPF are briefly discussed in Section 3.4 and 3.5, respectively.

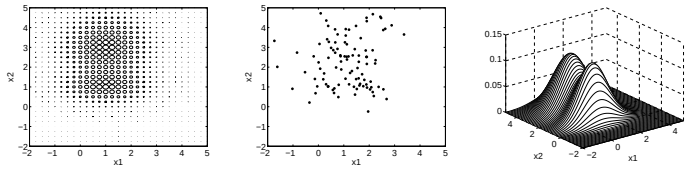
3.1 Representation

The task of nonlinear filtering can be split into two parts: representation of the filtering density function and the propagation of this density through the time and measurement update stages. Figure 7 illustrates different representations of the filtering density for a two-dimensional example (similar to the example used in [29]). The extended Kalman filter can be interpreted as using one Gaussian distribution for representation and the propagation is performed according to a linearized model. The Gaussian sum filter [1, 30] extends the EKF to be able to represent multi-modal densities, still with an approximate propagation.

Figure 7(d)–(f) illustrates numerical approaches where the exact nonlinear relations present in the model are used for propagation. The point mass filter (grid-based approximation) [4] employ a regular grid, where the grid weight is proportional to the filtering density. The PF represents the filtering density by a stochastic grid in the form of a set of samples, where all particles (samples) have the same weight. Finally, the marginalized particle filter uses a stochastic grid for some of the states, and Gaussian densities for the rest. That is, the MPF can be interpreted as a particle representation for a subspace of the state-space, where each particle has an associated Gaussian density for the remaining state dimensions, Figure 7(f). It will be demonstrated that an exact nonlinear propagation is still possible if there is a linear substructure present in the model.



(a) True PDF. (b) Gaussian approximation. (c) Gaussian sum approximation.



(d) Grid-based approximation. (e) Particle approximation. (f) MPF approximation.

Figure 7: *True probability density function and different approximate representations, in order of appearance, Gaussian, Gaussian sum, point masses (grid-based approximation), particle samples and MPF representation.*

3.2 Model Structure

A general nonlinear state-space model is given by

$$x_{t+1} = f_t(x_t, u_t) + w_t, \tag{3a}$$

$$y_t = h_t(x_t) + e_t. \tag{3b}$$

Note that the model (1) used in the positioning applications of the PF constitutes a special case of this structure. If there is a linear Gaussian substructure available in the model it is often rewarding to partitioned the state vector x_t according to

$$x_t = \begin{pmatrix} x_t^n \\ x_t^l \end{pmatrix}, \tag{4}$$

where the subvector x_t^l enters the dynamic model linearly with additive Gaussian noise, conditioned on x_t^n. Furthermore, x_t^n denotes the part of the state vector that does not fulfil the conditions for x_t^l. We will, for simplicity, informally refer to x_t^l as the linear states and x_t^n as the nonlinear states. A rather general model, containing a conditionally linear Gaussian substructure is given by

$$x_{t+1}^n = f_t^n(x_t^n) + A_t^n(x_t^n)x_t^l + G_t^n(x_t^n)w_t^n, \tag{5a}$$

$$x_{t+1}^l = f_t^l(x_t^n) + A_t^l(x_t^n)x_t^l + G_t^l(x_t^n)w_t^l, \tag{5b}$$

$$y_t = h_t(x_t^n) + C_t(x_t^n)x_t^l + e_t, \tag{5c}$$

where the process noise is assumed white and Gaussian distributed with

$$w_t = \begin{pmatrix} w_t^n \\ w_t^l \end{pmatrix} \sim \mathcal{N}(0, Q_t), \qquad Q_t = \begin{pmatrix} Q_t^n & Q_t^{ln} \\ (Q_t^{ln})^T & Q_t^l \end{pmatrix}. \tag{5d}$$

The measurement noise e_t is assumed white and Gaussian distributed according to

$$e_t \sim \mathcal{N}(0, R_t). \tag{5e}$$

Furthermore, x_0^l is Gaussian,

$$x_0^l \sim \mathcal{N}(\bar{x}_0, \bar{P}_0). \tag{5f}$$

Finally, the density of x_0^n can be arbitrary, but it is assumed known. More specifically, conditioned on the nonlinear state variables there is a linear substructure, subject to Gaussian noise available in (5).

Bayesian estimation methods, such as the particle filter, provide estimates of the filtering density function $p(x_t|y_{1:t})$. By employing the fact

$$p(x_t^l, x_{1:t}^n|y_{1:t}) = p(x_t^l|x_{1:t}^n, y_{1:t})p(x_{1:t}^n|y_{1:t}), \tag{6}$$

the overall problem is decomposed into two sub-problems,

- A Kalman filter operating on the conditionally linear, Gaussian model (5) provides an estimate of $p(x_t^l|x_{1:t}^n, y_{1:t})$.
- A particle filter for estimating the filtering density for the nonlinear states.

It is very important to note that the two sub-problems mentioned above are coupled. This coupling is given in the subsequent section. However, a complete derivation is out of the scope for the present work, see e.g., [28] for such a derivation. Here, it is worth noting that the model (5) can be further generalized by introducing an additional discrete mode parameter, giving a larger family of marginalized filters, see [29]. This has recently been applied in [26]. An important special case of (5) is a model with linear state equations and nonlinear measurement equations. For this case the computational complexity is significantly reduced, since the same covariance matrix can be used for all the Kalman filters. This will be clearer from the discussion below.

3.3 Algorithm

Using (6) the problem can be put in a form that is suitable for the MPF framework, i.e., to analytically marginalize out the linear state variables from $p(x_t|y_{1:t})$. Note that $p(x_t^l|x_{1:t}^n, y_{1:t})$ is analytically tractable, since $x_{1:t}^n$ is given by the particle filter. Hence, the underlying model is conditionally linear Gaussian, and the density function is given by the Kalman filter. Furthermore, since the estimate of $p(x_t^n|y_{1:t})$ is provided by the particle filter, it is given by

$$\hat{p}^N(x_t^n|y_{1:t}) = \sum_{i=1}^{N} \tilde{\omega}_t^{(i)} \delta(x_t^n - x_t^{n,(i)}). \tag{7}$$

Hence, the resulting estimate of the filtering density is given by

$$\hat{p}^N(x_t|y_{1:t}) = \sum_{i=1}^{N} \tilde{\omega}_t^{(i)} \delta(x_t^n - x_t^{n,(i)}) \mathcal{N}(x_t^l; \hat{x}_{t|t}^{l,(i)}, P_{t|t}^{(i)}), \tag{8}$$

motivating the fact that the MPF provides an estimate of the filtering density that is a mix of a parametric and an nonparametric estimate. That is a mix of a parametric distribution from the Gaussian family and a nonparametric distribution represented by samples. If the same number of particles is used in the standard PF and the MPF, the latter will provide estimates of better or at least the same quality. Intuitively this makes sense, since the dimension of $p(x_t^n|y_{1:t})$ is smaller than the dimension of $p(x_t|y_{1:t})$, implying that the particles occupy a lower dimensional space. Furthermore, the optimal algorithm (KF) is used to estimate the linear state variables. In Section 3.4 a more detailed discussion regarding the improved accuracy in the estimates is given. The MPF for estimating the states in a dynamic model in the form (5) is provided in Algorithm 1. For the sake of brevity, the particle index i and the dependence on the nonlinear state x_t^n are not explicitly stated in the algorithm.

Algorithm 1 (Marginalized particle filter)

1. *Initialization: For* $i = 1, \ldots, N$, *initialize the particles,* $x_{0|-1}^{n,(i)} \sim p_{x_0^n}(x_0^n)$ *and set* $\{x_{0|-1}^{l,(i)}, P_{0|-1}^{(i)}\} = \{\bar{x}_0^l, \bar{P}_0\}$. *Set* $t := 0$.

2. *PF measurement update: For* $i = 1, \ldots, N$, *evaluate the importance weights*

$$\omega_t^{(i)} = \frac{p(y_t|x_t^{(i)})p(x_t^{n,(i)}|x_{t-1}^{n,(i)})}{q(x_t^{n,(i)}|x_{1:t-1}^{n,(i)}, y_{1:t})}, \tag{9}$$

   and normalize $\tilde{\omega}_t^{(i)} = \omega_t^{(i)} / \sum_{j=1}^N \omega_t^{(j)}$.

3. *Draw* N *new particles, with replacement (resampling), for each* $i = 1, \ldots, N$,

$$Pr\left(x_{t|t}^{n,(i)} = x_{t|t-1}^{n,(j)}\right) = \tilde{\omega}_t^{(j)}, \qquad j=1,\ldots,\ N.$$

4. *PF time update and KF:*

   (a) *Kalman filter measurement update:*

$$\hat{x}_{t|t}^l = \hat{x}_{t|t-1}^l + K_t\left(y_t - h_t - C_t\hat{x}_{t|t-1}^l\right), \tag{10a}$$
$$P_{t|t} = P_{t|t-1} - K_t M_t K_t^T, \tag{10b}$$
$$M_t = C_t P_{t|t-1} C_t^T + R_t, \tag{10c}$$
$$K_t = P_{t|t-1} C_t^T M_t^{-1}. \tag{10d}$$

   (b) *PF time update (prediction): For* $i = 1, \ldots, N$, *predict new particles,*

$$x_{t+1|t}^{n,(i)} \sim q\left(x_{t+1|t}^n | x_{1:t}^{n,(i)}, y_{1:t+1}\right).$$

   (c) *Kalman filter time update:*

$$\begin{aligned}\hat{x}_{t+1|t}^l &= \bar{A}_t^l \hat{x}_{t|t}^l + G_t^l (Q_t^{ln})^T (G_t^n Q_t^n)^{-1} z_t + f_t^l \\ &\quad + L_t\left(z_t - A_t^n \hat{x}_{t|t}^l\right),\end{aligned} \tag{11a}$$
$$P_{t+1|t} = \bar{A}_t^l P_{t|t} (\bar{A}_t^l)^T + G_t^l \bar{Q}_t^l (G_t^l)^T - L_t N_t L_t^T, \tag{11b}$$
$$N_t = A_t^n P_{t|t} (A_t^n)^T + G_t^n Q_t^n (G_t^n)^T, \tag{11c}$$
$$L_t = \bar{A}_t^l P_{t|t} (A_t^n)^T N_t^{-1}, \tag{11d}$$

   where

$$z_t = x_{t+1}^n - f_t^n, \tag{12a}$$
$$\bar{A}_t^l = A_t^l - G_t^l (Q_t^{ln})^T (G_t^n Q_t^n)^{-1} A_t^n, \tag{12b}$$
$$\bar{Q}_t^l = Q_t^l - (Q_t^{ln})^T (Q_t^n)^{-1} Q_t^{ln}. \tag{12c}$$

5. *Set* $t := t + 1$ *and repeat from step* 2.

From Algorithm 1, it should be clear that the only difference from the standard PF is that the time update (prediction) stage has been changed. In the standard PF, the prediction stage is given solely by step $4(b)$ in Algorithm 1. In order to help intuition, step 4 in Algorithm 1 will now be briefly discussed. Step $4(a)$ is a standard Kalman filter measurement update using the information available in the measurement y_t. Once this has been performed the new estimates of the linear states can be used to obtain a prediction of the nonlinear state $x^n_{t+1|t}$. This is performed in Step $4(b)$. Now, consider model (5) conditioned on the nonlinear state variable. The conditioning implies that (5a) can be thought of as a measurement equation. This is used in step $4(c)$ together with a time update of the linear state estimates.

An alternative way is to describe the algorithm in terms of multiple models and a mixing of PF and KF time updates and measurement updates, see [18] for details on this approach. An elegant interpretation and derivation of the MPF in terms of a filter bank is provided in [17].

The estimates, as expected means, of the state variables and their covariances are given below.

$$\hat{x}^n_{t|t} = \sum_{i=1}^{N} \tilde{\omega}^{(i)}_t \hat{x}^{n,(i)}_{t|t}, \tag{13a}$$

$$\hat{P}^n_{t|t} = \sum_{i=1}^{N} \tilde{\omega}^{(i)}_t \left(\left(\hat{x}^{n,(i)}_{t|t} - \hat{x}^n_{t|t} \right) \left(\hat{x}^{n,(i)}_{t|t} - \hat{x}^n_{t|t} \right)^T \right), \tag{13b}$$

$$\hat{x}^l_{t|t} = \sum_{i=1}^{N} \tilde{\omega}^{(i)}_t \hat{x}^{l,(i)}_{t|t}, \tag{13c}$$

$$\hat{P}^l_{t|t} = \sum_{i=1}^{N} \tilde{\omega}^{(i)}_t \left(P^{(i)}_{t|t} + \left(\hat{x}^{l,(i)}_{t|t} - \hat{x}^l_{t|t} \right) \left(\hat{x}^{l,(i)}_{t|t} - \hat{x}^l_{t|t} \right)^T \right), \tag{13d}$$

where $\{\tilde{\omega}^{(i)}_t\}_{i=1}^N$ are the normalized importance weights, provided by step 2 in Algorithm 1.

3.4 Variance Reduction

The *law of total variance* says that

$$\mathsf{Cov}\left(U\right) = \mathsf{Cov}\left(\mathrm{E}(U|V)\right) + \mathrm{E}\left(\mathsf{Cov}(U|V)\right), \tag{14}$$

where U and V denotes stochastic variables. Letting $U = x^l_t$ and $V = x^n_{1:t}$ results in the following decomposition of the variance of the PF

$$\underbrace{\mathsf{Cov}(x^l_t)}_{PF} = \mathsf{Cov}\left(\mathrm{E}(x^l_t|x^n_{1:t})\right) + \mathrm{E}\left(\mathsf{Cov}(x^l_t|x^n_{1:t})\right) \tag{15a}$$

$$= \underbrace{\mathsf{Cov}\left(\hat{x}^l_{t|t}\right)}_{MPF} + \sum_{i=1}^{N} w^{(i)}_t \underbrace{P^{(i)}_{t|t}}_{KF}. \tag{15b}$$

Here, we recognize $p(x^l_t|x^n_{1:t})$ as the Gaussian distribution, delivered by the KF, conditioned on the trajectory $x^n_{1:t}$. Now, the MPF computes the mean of

each trajectory as $\hat{x}_{t|t}^{l,(i)}$ and the unconditional mean estimator is simply the mean of these, $\sum_{i=1}^{N} \tilde{w}_t^{(i)} \hat{x}_{t|t}^{l,(i)}$, and its covariance follows from the first term in (15b). The second term in (15b) corresponds to the contribution due to the fact that each Gaussian distribution is represented by one sample, as is done in the PF. This principle, which leads to the variance reduction property is sometimes referred to as Rao-Blackwellization, see, e.g., [27] and it is the basic part that improves performance using the marginalization idea. Note that for the variance reduction to be significant, the left hand side in (15) has to be significantly smaller than the right hand side. In other words, the term

$$\mathrm{E}\left(\mathsf{Cov}(x_t^l | x_{1:t}^n)\right) \tag{16}$$

has to be large. That is, the expectation of the conditional variance of the corresponding Kalman filter has to be large. In order to make this a bit clearer, we have

$$\mathsf{Cov}\left(\mathrm{E}(x_t^l | x_{1:t}^n)\right) \leq \mathsf{Cov}(x_t^l), \tag{17}$$

since $\mathrm{E}\left(\mathsf{Cov}(x_t^l | x_{1:t}^n)\right) \geq 0$. This shows that the variance of the linear part is always smaller for the MPF than for the PF. The difference is the expected covariance term,

$$\mathrm{E}\left(\mathsf{Cov}(x_t^l | x_{1:t}^n)\right) = \sum_{i=1}^{N} \tilde{\omega}_t^{(i)} P_{t|t}^{(i)}. \tag{18}$$

This states that the improvement in the quality of the estimate is given by the term $\mathrm{E}\left(\mathsf{Cov}(x_t^l | x_{1:t}^n)\right)$. That is, the Kalman filter covariance $P_{t|t}$ is a good indicator of how much has been gained in using the MPF instead of the PF. Further discussions regarding the variance reduction property of the MPF are provided in e.g., [9, 10].

3.5 Computational Complexity

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

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

This is schematically illustrated in Figure 8, for the case where $C = 0$, implying that the same covariance matrix can be used for all particles. The two alternatives above are illustrated with the thin lines, in case a PF with 10000 particles is first applied, and then replaced by the MPF.

Another related question is how the computational complexity relates to the number of particles. The MPF appears to add quite a lot of overhead computations. It turns out, however, that the MPF is often more efficient also here. It may seem impossible to give any general conclusions, so application dependent simulation studies have to be performed. Nevertheless, quite realistic predictions about the computational complexity can be done with rather simple calculations, an example of this is given below.

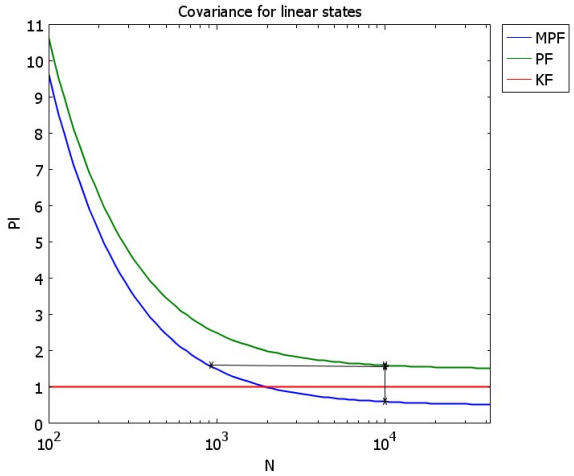


Figure 8: *Schematic view of how the covariance of the linear part of the state vector depends on the number of particles for the PF and MPF, respectively. The gain in MPF is given by the Kalman filter covariance.*

3.6 Radar Tracking Case Study

This section will serve as an illustration of the computational complexity of the marginalized particle filter. A special case of the general model (5) will be used, where the state dynamics are linear and Gaussian and the measurements are nonlinear functions in some of the states:

$$x_{t+1} = \begin{pmatrix} 1 & 0 & T & 0 & T^2/2 & 0 \\ 0 & 1 & 0 & T & 0 & T^2/2 \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} x_t + w_t, \tag{19a}$$

$$y_t = \begin{pmatrix} \sqrt{X_t^2 + Y_t^2} \\ \arctan\left(Y_t/X_t\right) \end{pmatrix} + e_t, \tag{19b}$$

where the state vector is $x_t = (X\ Y\ \dot{X}\ \dot{Y}\ \ddot{X}\ \ddot{Y})^T$, i.e., position, velocity and acceleration in two dimensions.

This is an archetypal model for many target tracking and some localization problems, where the state dynamics is modeled using a constant velocity or a constant acceleration assumption. The observation can in most applications be written as a nonlinear function of position. This is the case for bearings-only, range-only and radar sensors. More specifically, the measurement equation in (19b) models measurements of the range and the azimuth from a radar system.

In [21] a general method for analyzing the computational complexity of the MPF was presented. Here, the highlights of this analysis will be given, based on model (19).

The model has two nonlinear state variables and four linear state variables. Two cases are now studied, the full PF, where all states are estimated using the PF and the completely marginalized PF, where all linear states are marginalized

out and estimated using the KF. If we want to compare the two approaches under the assumption that they use the same computational resources, we obtain

$$N_{\text{PF}} = \underbrace{(1-c)}_{<1} N_{\text{MPF}}. \tag{20}$$

where N_{PF} and N_{MPF} denote the number of particles used in the PF and the MPF, respectively. Furthermore, c is a positive constant depending of the computational complexity of the various parts of the algorithm, see [21] for the complete details. From (20) it is clear that for a given computational complexity more particles can be used in the MPF than in the standard PF, in this example.

Using a constant computational complexity the number of particles that can be used is computed. The study is performed by first running the full PF and measure the time consumed by the algorithm. A Monte Carlo simulation, using $N = 2000$ particles, is performed in order to obtain a stable estimate of the time consumed by the algorithm. In the left hand side of Table 1 the number of particles (N), the total RMSE from 100 Monte Carlo simulations, and the simulation times are shown both for the PF and the MPF.

Table 1: *Results from the simulation. Left: Using a constant computational complexity, i.e., the algorithms can consume the same amount of time. Right: Using a constant velocity RMSE.*

	PF	**MPF**
N	2000	2574
RMSE pos	7.10	5.60
RMSE vel	3.62	3.21
RMSE acc	0.52	0.44
Time	0.59	0.60

	PF	**MPF**
N	2393	264
RMSE pos	7.07	7.27
RMSE vel	3.58	3.61
RMSE acc	0.50	0.48
Time	0.73	0.10

From Table 1 it is clear that for this example, the MPF can use more particles for a given time, which is in perfect correspondence with the theoretical result summarized in (20).

Let us now discuss what happens if a constant velocity RMSE is used. First the velocity RMSE for the full PF is found using a Monte Carlo simulation. This value is then used as a target function in the search for the number of particles needed by the MPF. The right hand side of Table 1 clearly indicates that the MPF can obtain the same RMSE using fewer particles. The result is that using full marginalization only requires 14% of the computational resources as compared to the standard PF in this example.

Note that for this example $C_t = 0$, implying that the same covariance matrix can be used for all Kalman filters, which significantly reduce the computational complexity.

4 Simultaneous Localization and Mapping

Simultaneous localization and mapping (SLAM) is an extension of the positioning problem previously discussed to the case where the environment is unknown.

In SLAM this is handled by estimating a map on-line together with the platform states. The SLAM problem has been studied for a long time within many different settings and an introduction to the problem can be found in the survey papers [3, 11] and the book [31]. Here, the focus will be on the filtering part of the SLAM problem. The SLAM problem also contains very interesting issues when it comes to feature extraction, data association and matching features in consecutive images.

4.1 Problem Formulation

Many different estimation algorithms, including for example the extended Kalman filter, the particle filter and the extended information filter, have been used in solving the SLAM problem. A key observation here is that the most promising algorithms so far utilize the structure inherent in the SLAM problem in one way or the other. When it comes to the algorithms based on the particle filter it is perhaps the FastSLAM algorithm [24, 23] that has received most attention. FastSLAM can be seen as a special case of the MPF. Here, the state vector contains information about the platform x_t and the position of the features m_t, which consists of the entire map at time t, i.e.,

$$m_t = \begin{pmatrix} m_{1,t}^T & \dots & m_{M_t,t}^T \end{pmatrix}^T, \tag{21}$$

where $m_{j,t}$ denotes the position of the j^{th} map entry at time t and M_t denotes the number of entries in the map at time t. The underlying idea is to use the particle filter to estimate the platform states and to model the landmarks as linear Gaussian states, which are estimated using Kalman filters. We are interested in estimating the filtering density function $p(x_t, m_t|y_{1:t})$, which using Bayes' theorem can be expressed as

$$p(x_{1:t}, m_t|y_{1:t}) = p(m_t|x_{1:t}, y_{1:t})p(x_{1:t}|y_{1:t}), \tag{22}$$

which is exactly the factorization previously employed in (6). The FastSLAM algorithm was originally devised to solve the SLAM problem for mobile robots, where the dimension of the state vector is small, typically consisting of only three states (2D position and a course (heading) angle) [31]. This implies that all platform states can be estimated by the PF. In discussing the FastSLAM algorithm it is worth mentioning that when $p(x_{t+1}|x_t)$ is used as proposal density the algorithm is referred to as FastSLAM1.0 and when $p(x_{t+1}|x_{1:t}, y_{1:t+1})$ is used it is referred to as FastSLAM2.0. The choice of importance density makes a rather big difference in the SLAM application.

4.2 Marginalized FastSLAM

In considering different platforms, for example an unmanned aerial vehicle, the dimension of the state vector describing the platform will typically be quite high. The platform dynamics often has a linear, Gaussian substructure available in it [15] and we can of course exploit this structure as well. This will give us an algorithm capable of dealing with high dimensional platform state vectors as well. Let us for this case partition the state vector according to

$$x_t = \begin{pmatrix} (x_t^p)^T & (x_t^k)^T & m_t^T \end{pmatrix}^T, \tag{23}$$

where x_t^p denotes the states of the platform that are estimated by the particle filter, and x_t^k denotes the states of the platform that are linear and Gaussian, given information about x_t^p. These states together with the map (landmarks) m_t are estimated using Kalman filters. The dynamic model is a special case of (5),

$$x_{t+1}^p = f_t^p(x_t^p) + A_t^p(x_t^p)x_t^k + G_t^p(x_t^p)w_t^p, \tag{24a}$$
$$x_{t+1}^k = f_t^k(x_t^p) + A_t^k(x_t^p)x_t^k + G_t^k(x_t^p)w_t^k, \tag{24b}$$
$$m_{j,t+1} = m_{j,t}, \tag{24c}$$
$$y_{1,t} = h_{1,t}(x_t^p) + C_t(x_t^p)x_t^k + e_{1,t}, \tag{24d}$$
$$y_{2,t}^{(j)} = h_{2,t}(x_t^p) + H_{j,t}(x_t^p)m_{j,t} + e_{2,t}^{(j)}, \tag{24e}$$

where $x_t^n = x_t^p$ and $x_t^l = \left((x_t^k)^T \ m_t^T\right)^T$. In order to compute an estimate of the filtering density function $p(x_t^p, x_t^k, m_t|y_{1:t})$, the key is the following factorization

$$p(x_{1:t}^p, x_t^k, m_t|y_{1:t}) = p(m_t|x_{1:t}^p, x_t^k, y_{1:t})p(x_t^k|x_{1:t}^p, y_{1:t})p(x_{1:t}^p|y_{1:t}) \tag{25a}$$
$$= \prod_{j=1}^{M_t} \underbrace{p(m_{j,t}|x_{1:t}^p, x_t^k, y_{1:t})p(x_t^k|x_{1:t}^p, y_{1:t})}_{\text{(extended) Kalman filter}} \underbrace{p(x_{1:t}^p|y_{1:t})}_{\text{particle filter}}, \tag{25b}$$

where the second expression follows if all features are assumed independent. Using this factorization together with model (24) and Algorithm 1 we can derive a rather general algorithm which is applicable to many different platforms (aircraft, helicopters, cars, etc.). The i^{th} particle at time t, $x_t^{(i)}$ will contain the following information

$$x_t^{(i)} = \left(x_t^{p,(i)}, \hat{x}_t^{k,(i)}, (\hat{m}_{1,t}, \Sigma_{1,t}), \ldots, (\hat{m}_{M_t,t}, \Sigma_{M_t,t})\right), \tag{26}$$

where $\hat{m}_{j,t}$ and $\Sigma_{j,t}$ provide the position estimate and the associated covariance for the j^{th} map entry, respectively. The details of the derivation are thoroughly discussed in [22, 32], where we also give an application example in terms of an unmanned aerial vehicle. In order to illustrate the algorithm we will give some insight to this example here.

4.3 Unmanned Aerial Vehicle Application

The information we use to solve the SLAM problem originates from the following sensors:

- camera
- inertial measurement unit (IMU)
- pressure sensor

Hence, solving the SLAM problem for this particular case is a sensor fusion problem where the information from the above mentioned sensors is fused. Note that we have to make use of a linearized measurement model (24e) for the camera in order to fit the current framework, again we refer to [22] for the details. We have employed a rather simple solution when it comes to the map and the

computer vision part. We simply make use of the Harris detector [16] in order to find points of interest. Then we cut out an 11×11 patch around this interest point. These patches will then constitute our map. Similar approaches have previously been used for similar problems, see e.g., [7]. The performance is illustrated in Figure 9.

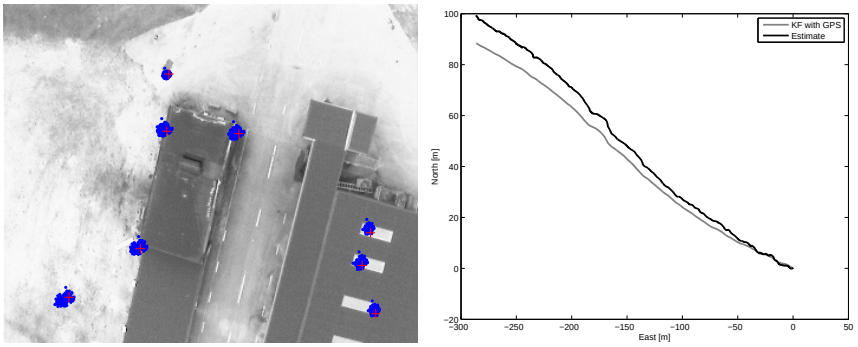


Figure 9: *Left: The scenario seen from the on-board camera, together with particle clouds representing the landmarks/map features. The crosses show the measurements from the computer vision algorithm. Right: Horizontal position estimate from the SLAM algorithm, compared to the GPS, which is only used as reference here. The unmanned aerial vehicle starts in the origin at time* $t = 0$.

5 Conclusions

The particle filter (PF) has been successfully used in quite a few different applications, at least when the state dimension is moderately low. Design issues, such as the choices of number of particles, resampling strategies and proposal density are becoming better understood both in theory and practice. There are many suggested ad-hoc strategies for practical aspects as divergence monitoring and dithering. We have in this survey used four localization applications to illustrate these issues. We have also pointed out the similarity with the FastSLAM algorithm for simultaneous localization and mapping used in robotics.

However, for complex high-dimensional models, the number of applications in literature is quite limited. The surveyed localization applications all work well using the PF for their simplest problem formulations leading to low-dimensional models. However, we have also, for each application, motivated that more complex models would give additional performance, if only the PF can be applied. Now, the curse of dimensionality prevents the use of the PF in its original form. However, all these applications reveal a linear Gaussian substructure that enables the use of the MPF. The MPF strategy can be generalized to models with substructures that are almost linear and Gaussian, using the same arguments as when applying the EKF to nonlinear models. In fact, it is hard to find high-dimension models useful in practice where all states are severely nonlinear or non-Gaussian. To illustrate the concept, the MPF was applied to a kinematic model with a high dimensional state vector for navigating an unmanned aerial vehicle using vision and inertial sensors. Here, the PF only involved the

two-dimensional horizontal position, while all other states were sufficiently linear and Gaussian for the EKF. To understand the principles of the MPF, we have reviewed a general model structure, the MPF algorithm and discussed the interplay of variance reduction and computational complexity when the PF is replaced with a MPF.

References

[1] B. D. O. Anderson and J. B. Moore. *Optimal Filtering*. Information and system science series. Prentice Hall, Englewood Cliffs, NJ, USA, 1979.

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

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

[4] N. Bergman. *Recursive Bayesian Estimation: Navigation and Tracking Applications*. Phd thesis No 579, Linköping Studies in Science and Technology, SE-581 83 Linköping, Sweden, May 1999.

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

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

[7] A. J. Davison, I. Reid, N. Molton, and O. Strasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Patterns Analysis and Machine Intelligence*, 29(6):1052–1067, June 2007.

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

[9] A. Doucet, N. Gordon, and V. Krishnamurthy. Particle filters for state estimation of jump Markov linear systems. Technical Report CUED/F-INFENG/TR 359, Signal Processing Group, Department of Engineering, University of Cambridge, Trupington street, CB2 1PZ Cambridge, 1999.

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

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

[12] K. M. Fauske, F. Gustafsson, and O. Herenaes. Estimation of AUV dynamics for sensor fusion. In *Proceedings of the 10th international conference on information fusion*, Québec, Canada, July 2007.

[13] U. Forssell, P. Hall, S. Ahlqvist, and F. Gustafsson. Novel map-aided positioning system. In *Proceedings of FISITA*, Helsinki, Finland, 2002.

[14] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings on Radar and Signal Processing*, volume 140, pages 107–113, 1993.

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

[16] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, Manchester, UK, 1988.

[17] G. Hendeby. *Performance and Implementation Aspects of Nonlinear Filtering*. Phd thesis No 1161, Linköping Studies in Science and Technology, Department of Electrical Engineering, Linköping University, Sweden, February 2008.

[18] G. Hendeby, R. Karlsson, and F. Gustafsson. A new formulation of the Rao-Blackwellized particle filter. In *IEEE Statistical Signal Processing Workshop*, Madison, Wisconsin, USA, 2007.

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

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

[21] R. Karlsson, T. Schön, and F. Gustafsson. Complexity analysis of the marginalized particle filter. *IEEE Transactions on Signal Processing*, 53(11):4408–4411, November 2005.

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

[23] M. Montemerlo and S. Thrun. *FastSLAM A scalable method for the simultaneous localization and mapping problem in robotics*. Star – Springer tracts in advanced robotics. Springer, Berlin, Germany, 2007.

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

[25] P.-J. Nordlund. *Sequential Monte Carlo Filters and Integrated Navigation*. Licentiate Thesis No 945, Department of Electrical Engineering, Linköping University, Sweden, 2002.

[26] P.-J. Nordlund and F. Gustafsson. Marginalized particle filter for accurate and reliable terrain-aided navigation. *Provisionally accepted for IEEE Transactions on Aerospace and Electronic Systems*, 2009.

[27] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer texts in statistics. Springer, New York, 1999.

[28] T. Schön, F. Gustafsson, and P.-J. Nordlund. Marginalized particle filters for mixed linear/nonlinear state-space models. *IEEE Transactions on Signal Processing*, 53(7):2279–2289, July 2005.

[29] T. B. Schön, R. Karlsson, and F. Gustafsson. The marginalized particle filter in practice. In *Proceedings of IEEE Aerospace Conference*, Big Sky, MT, USA, March 2006.

[30] H. W. Sorenson and D. L. Alspach. Recursive Bayesian estimation using Gaussian sum. *Automatica*, 7:465–479, 1971.

[31] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents. The MIT Press, Cambridge, MA, USA, 2005.

[32] D. Törnqvist, T. B. Schön, R. Karlsson, and F. Gustafsson. Particle filter SLAM with high dimensional vehicle model. *Journal of Intelligent and Robotic Systems*, 55(4–5):249–266, August 2009.