Storage Efficient Particle Filters with Multiple Out-of-Sequence Measurements

Berntorp, Karl; Årzén, Karl-Erik; Robertsson, Anders

Published in:
15th International Conference on Information Fusion (FUSION), 2012

2012

Storage Efficient Particle Filters with Multiple Out-of-Sequence Measurements

Karl Berntorp, Karl-Erik Årzén and Anders Robertsson
Lund University, Box 118, SE-221 00 Lund, Sweden (E-mail: firstname.lastname@control.lth.se)

***Abstract*—A particle filter based solution to the out-of-sequence measurement (OOSM) problem is proposed. The solution is storage efficient, while being computationally fast. The filter approaches the multi-OOSM problem by not only updating the estimate at the most recent time, but also for all times between the OOSM time and the most recent time. This is done by exploiting the complete in-sequence information approach and extending it to nonlinear systems. Simulation experiments on a challenging nonlinear tracking scenario show that the new approach outperforms recent state-of-the-art particle filter algorithms in some respects, despite demanding less storage requirements.**

I. Introduction

In multisensor target-tracking systems local sensor measurements are typically sent to a common center, where the measurements are fused to form position estimates. Some measurements can arrive such that more current measurements have already been processed; that is, they can be out of sequence, caused by different data processing times and transmission times. The ability to process out-of-sequence measurements (OOSMs) is important for several reasons. Obviously, discarding the OOSM can lead to poor performance since the delayed measurement contains useful information. Using the OOSM in the wrong way can, however, also lead to degraded performance because of mismatch between the different measurements.

We suggest an extension of a new type of storage efficient particle filter derived in [1]. The particle filter in [1] usually performs well, but it suffers from that the employed fixed-point smoother only updates the most recent estimate when a delayed measurement arrives. To remedy this we extend the optimal complete in-sequence information fixed-point smoother (CISI-FPS), described in [2] for linear systems, to nonlinear systems and use it in the context of particle filters. The CISI approach updates all states between the OOSM timestamp and the current time, thus performing better than only updating the current estimate. The modification for the CISI-FPS approach to work for nonlinear models is a Taylor expansion, analogous to those used in extended Kalman smoothers, see [3]. It will be shown, using a similar simulation example as in [1] and [4], that the new algorithm outperforms the approach proposed in [1]. In addition, only mean and covariances for the last $l_{\max}$ steps, where $l_{\max}$ is the predetermined maximum number of lags of the OOSMs, need to be stored, which is an improvement from the approach in [1] where the measurements also need storage. This comes with the price of slightly larger computational demands, but a comparison shows that the increase is moderate.

A. Related Work

Over the last decade there has been substantial research considering out-of-sequence measurements for tracking. An overview of initial work spanning to the late 1990's is found in [5]. In [6], [7], and [8], the research progressed from deriving the optimal solution when the OOSM was assumed to be delayed less than one sampling interval, to deriving suboptimal algorithms with delays that were several sampling intervals long, and finally handling sensor bias. The optimal solution, in the mean-square sense, was derived in [9] for different amounts of available information.

One of the drawbacks with the presented approaches is that only the most recent estimate is updated with the OOSM. In real-life scenarios there are typically multiple OOSMs arriving, either in succession or interleaved with in-sequence measurements (ISMs). In this case the preceding approaches will in general not be optimal. The first optimal solution to the multi-OOSM problem is due to [10], where the assumption was that the out-of-sequence measurements were not interleaved with the in-sequence measurements. The first general optimal solution with multiple out-of-sequence measurements, denoted the complete in-sequence information (CISI) approach, was presented in [2]. A number of approaches yielding the optimal solution were compared in terms of complexity. The conclusion was that the CISI-FPS approach is superior compared to the CISI fixed-interval smoothing approach, the fading information approach, and the information filter-equivalent measurement method in terms of computational demands. Furthermore, the storage requirements in [2] are only the mean and covariances for $k-l_{\max}, k-l_{\max}+1, \ldots, k-1, k$, where k is the current time index. For the scenario with several OOSMs arriving at the same time the CISI approach is applied sequentially, still giving optimality. For an overview of linear smoothing methods, see [11].

All work presented above is for linear systems. The implementation-wise easiest extension for nonlinear systems is to use extended Kalman filter (EKF) type approximations. For systems with significant nonlinearities and/or non-Gaussian noise the use of EKFs can, however, lead to poor performance. Several methods for exploring OOSMs in the more general particle filter framework have been derived. Papers [12] and [13] outlined a particle filter based solution, where the measurement equation is allowed to be nonlinear. The

particle weights are updated without the OOSM and are then modified utilizing the OOSM with a Markov chain Monte Carlo smoothing step to overcome the problem of degeneracy in the particle filter. However, the state-space model has to be linear to be able to form the proposal density. Another drawback with this approach is that the storage requirements are large, since all particles have to be stored for the last $l_{\max}$ steps. A workaround for this was described in [14], where an invertible state-transition matrix is assumed. This matrix is then used to retrodict the states back to the time of the OOSM. The only storage requirements in this algorithm are the mean and covariances for the last $l_{\max}$ steps. A comparison between particle filters and Kalman filters for OOSM filtering is found in [15]. For the presented example, which is linear, the two types of estimators perform similarly.

To enable nonlinear state-space models, an extension of [14] was presented in [1], denoted storage efficient particle filter (SEPF). In that extension, only the mean, covariances, and measurements for the last $l_{\max}$ steps are stored. Since there is only one measurement vector y_k at each time instant, and since the dimension of the measurements usually is less than that of the states, the storage requirements compared to storing the particles should be considerably smaller. Different fixed-point smoothers are used to determine the likelihood of the measurement given each particle at the current time. The likelihood is then utilized to update the weight of the particle. When comparing extended Kalman smoothers (EKS), unscented Kalman smoothers, and particle smoothers on a highly nonlinear example, EKS seems to outperform the others despite demanding less computational power. As mentioned previously SEPF usually performs very well. The performance may suffer, however, when the OOSMs change the particle weights too much. This problem is partially overcome in [4] and [16] where an algorithm for detecting these OOSMs was derived, denoted SEPF with selective processing.

B. Outline

Section II presents the OOSM problem formulation and makes a distinguishment of different types of OOSMs. We give a brief presentation of particle filters in Section III. We also give a review of the storage efficient particle filter with and without selective processing, respectively. The proposed algorithm is outlined in Section IV. A validation is performed in Section V using a challenging simulation scenario. Finally, the paper is concluded in Section VI.

II. Problem Formulation

We consider the scenario of possibly nonlinear state dynamics, possibly nonlinear measurement relations, and additive process and measurement noise. Some measurements experience negligible delay, implying that they can be processed without taking the delay into account. However, a considerable portion of the measurements arrive with delays that have to be accounted for. The considered systems are of the form

$$x_{k+1} = f_{k+1,k}(x_k) + v_{k+1,k}, \tag{1}$$

$$y_k = h_k(x_k) + e_k, \tag{2}$$

where x_k is the state at time index k, $f_{k+1,k}$ is the state-transition function from time index k to $k+1$, h_k is the measurement function, $v_{k+1,k}$ is the process noise, and y_k is the measurement corrupted with measurement noise e_k. The measurement noise e_k is independent of $v_{k+1,k}$. The timestamp is referred to as t_k. Furthermore, the arrival time of a measurement is written as t_k^a. Denote the set of measurements generated in the interval $[i, j]$ available at time t_k as $\mathcal{W}_k^{i:j}$. Let $\mathcal{Z}_{k,\bar{\tau}}$ denote the set of OOSMs available at time t_k except y_τ. For clarity a definition of OOSMs and ISMs is given in Definition 1.

Definition 1. Given a measurement y_τ, if there exists another measurement y_k with $t_k^a < t_\tau^a$ and $t_k > t_\tau$, then y_τ is an OOSM. Otherwise, y_τ is an ISM. □

To distuingish between different OOSM scenarios the notion of *most recent time* is defined next:

Definition 2. Given an OOSM y_τ, if

$$t_{m(\tau)} = \max\{t_k;\ \forall t_k,\ t_k^a < t_\tau^a\},$$

then $t_{m(\tau)}$ is the most recent time (MRT) corresponding to y_τ. □

As mentioned in Section I, for linear systems it is only under special circumstances that it is enough to update the most recent mean and covariance and still have optimality. The case for when this approach is optimal is when the OOSM scenario is of *type I*, see [2]. The type I scenario is explained in Definition 3.

Definition 3. If for any two OOSMs y_{τ_1} and y_{τ_2}, where y_{τ_1} arrives before y_{τ_2} (i.e., $t_{\tau_1}^a < t_{\tau_2}^a$), we have that the most recent time corresponding to y_{τ_1} is before the timestamp of y_{τ_2} (i.e., $t_{m(\tau_1)} < t_{\tau_2}$), then the OOSM scenario is of type I. □

An example of a type I OOSM scenario is shown in Fig. 1, and a scenario that does not fulfill the assumptions in Definition 3 is shown in Fig. 2. Only the CISI approach

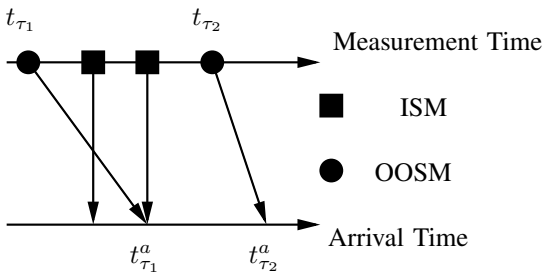


Fig. 1. An example of the type I OOSM scenario when, for linear systems, it is optimal to use the OOSMs for updating the most recent estimate only. For an explanation of OOSMs and ISMs, see Definition 1.

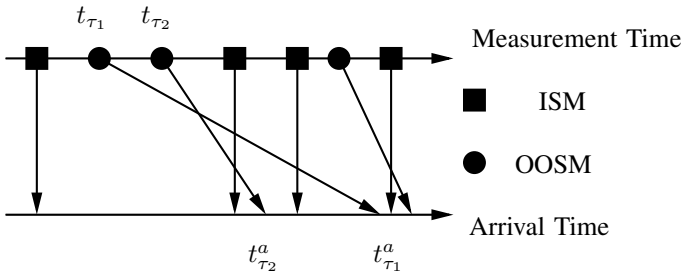


Fig. 2. An example of an OOSM scenario which is not of type I, when one must update all states between the OOSM measurement time and the current time to guarantee optimality.

guarantees optimality when the OOSM scenario is not of type I, which is often the case when having more than one sensor producing out-of-sequence measurements.

Suppose that we have processed measurements up to time index k and that one or several delayed measurements arrive. The problem discussed in the sequel is that of using y_τ to approximate the posterior distribution $p(x_k|y_{0:k,\tau})$. When an approximation of the distribution is formed, the position estimates and associated covariances can be generated.

III. Particle Filters with OOSM

The particle filter that the OOSM algorithm is based on is stated next, which is followed by a summary of the storage efficient particle filter algorithm. This is the starting point of the algorithm developed in Section IV.

A. Particle Filters

Particle methods, or sequential Monte Carlo methods, see [17] for a detailed introduction, are methods that try to represent the probability density function by a set of particles. This is done by approximating the density function as

$$p(x_{0:k}|y_{1:k}) \approx \sum_{i=1}^{N} w_{0:k}^{(i)} \delta(x_{0:k} - \hat{x}_{t|0:k}^{(i)}), \tag{3}$$

where $\delta(.)$ is the delta function, $\hat{x}_{t|0:k}^{(i)}$ is the ith particle at time index t given measurements from time index 0 to k, and $w_t^{(i)}$ is the normalized (scalar) weight of particle i. The Bayesian recursion formula

$$p(x_{0:k}|y_{1:k}) = \frac{p(y_k|x_{0:k}, y_{1:k-1})p(x_{0:k}|y_{1:k-1})}{p(y_k|y_{1:k-1})} \times p(x_{0:k-1}|y_{1:k-1}) \tag{4}$$

is used to get a sequential relationship. The complexity of the distribution (3) increases with time. Therefore the sampling is performed by using the proposal density $q(x_{0:k}|y_{1:k})$, which has the property

$$q(x_{0:k}|y_{1:k}) = q(x_k|x_{0:k-1}, y_{1:k})q(x_{0:k-1}|y_{1:k-1}). \tag{5}$$

Because of (5), it is enough to sample the last state components from the distribution $\hat{x}_k^{(i)} \sim q(x_k|x_{0:k-1}, y_{1:k})$. A simplification of (5) by assuming $q(x_k|x_{0:k-1}, y_{1:k}) = p(x_k|x_{k-1})$ and using (4) leads to that the weights are updated as

$$w_k^{(i)} \propto p(y_k|\hat{x}_k^{(i)}) w_{k-1}^{(i)}. \tag{6}$$

The position and covariance estimates at time index k can now be formed as

$$\hat{x}_{k|k} = \sum_{i=1}^{N} w_k^{(i)} \hat{x}_k^{(i)}, \tag{7}$$

$$P_{k|k} = \sum_{i=1}^{N} w_k^{(i)} (\hat{x}_k^{(i)} - \hat{x}_{k|k})(\hat{x}_k^{(i)} - \hat{x}_{k|k})^{\mathrm{T}}, \tag{8}$$

where $\hat{x}_k^{(i)}$ is the ith sample from $p(x_k|x_{k-1})$. Since all particles but a few typically will have negligible weights after a while, it is common that new particles are drawn with replacement according to

$$\Pr(x_k^{(i)} = \hat{x}_k^{(j)}) = w_k^j, \quad j = 1, \ldots, N. \tag{9}$$

This is one form of the well known sampling importance resampling (SIR) particle filter. The algorithm, as used in this work, is outlined in Algorithm 1.

Algorithm 1. *SIR Particle Filter Algorithm*

1: *Initialize particles* $\{\hat{x}_{0|0}^{(i)}\}_{i=1}^N \sim p_{x_0}(x_0)$ *and weights* $w_0^{(i)} = 1/N$, $i = 1, \ldots, N$

2: *Time Update: Generate new particles from the importance density*

$$\hat{x}_t^{(i)} \sim p(x_t|\hat{x}_{t-1}^{(i)}),\ i = 1, \ldots, N$$

3: *Measurement Update: Calculate weights as in (6) and normalize them. Form the estimates from (7) and (8).*

4: *Resample according to (9). Go back to step 2.*

B. Storage Efficient Particle Filters

The starting point for the storage efficient particle filters (SEPFs) developed in [1] is that measurements until time t_k have been processed using a particle filter. After the kth measurement the $(k+1)$th measurement arrives delayed, with timestamp t_τ, bounded as $t_\tau \in [t_{k-l}, t_{k-l+1})$ for a positive integer l. To avoid saving the particles while enabling nonlinear state-space models, the storage efficient particle filter in [1] utilizes Bayes' rule as

$$p(x_k|y_{0:k,\tau}) = \frac{p(y_\tau|x_k, y_{0:k})}{p(y_\tau|y_{0:k})} p(x_k|y_{0:k}), \tag{10}$$

a marginalization of (4). Substitution of (3) into (10) leads to

$$\begin{aligned} p(x_k|y_{0:k,\tau}) &= \sum_{i=1}^{N} \frac{p(y_\tau|\hat{x}_k^{(i)}, y_{0:k})}{p(y_\tau|y_{0:k})} w_{0:k}^{(i)} \delta(x_k - \hat{x}_k^{(i)}) \\ &= \sum_{i=1}^{N} w_{0:k,\tau}^{(i)} \delta(x_k - \hat{x}_k^{(i)}), \end{aligned} \tag{11}$$

where the weights are generated by

$$w_{0:k,\tau}^{(i)} \propto p(y_\tau|\hat{x}_k^{(i)}, y_{0:k}) w_{0:k}^{(i)}. \tag{12}$$

Now, by using Bayes' rule, the total probability theorem, and considering the particle state $\hat{x}_k^{(i)}$ as a measurement of the state x_{k-1} using (1) as the measurement relation, [1] ends up with the Gaussian approximation

$$p(y_\tau|x_k^{(i)}, y_{0:k}) \approx \mathcal{N}(y_\tau|\hat{y}_{\tau|0:k-1,k^{(i)}}, P^y_{\tau|0:k-1,k^{(i)}}), \quad (13)$$

where

$$\hat{y}_{\tau|0:k-1,k^{(i)}} = h_\tau(\hat{x}_{\tau|0:k-1,k^{(i)}}), \quad (14)$$

$$P^y_{\tau|0:k-1,k^{(i)}} = H_\tau P^y_{\tau|0:k-1,k^{(i)}} H_\tau^{\mathrm{T}} + R_\tau. \quad (15)$$

In these equations, $P^y_{\tau|0:k-1,k^{(i)}}$ is the covariance matrix at time τ conditioned on measurements up to time index $k-1$ and the particle estimates at time index k, $H_\tau = \frac{\partial h_\tau(x)}{\partial x}\Big|_{x=\hat{x}_{\tau|0:k-1,k^{(i)}}}$, and R_τ is the covariance of e_τ. To find $\hat{x}_{\tau|0:k-1,k^{(i)}}$ is a fixed-point smoothing problem, and according to the evaluation performed in [1] the best smoother is the state-augmented extended Kalman smoother. It was pointed out in [1] that sometimes the OOSMs can lead to severe mismatch between earlier particle weights and the updated weights, which may result in poor performance. The remedy for this was to calculate the approximate effective sample number, $\frac{1}{\sum_i w_i^2}$, before and after the OOSM update, $N_{\mathrm{eff}}^{\mathrm{pri}}$ and $N_{\mathrm{eff}}^{\mathrm{post}}$, respectively. If $N_{\mathrm{eff}}^{\mathrm{post}}/N_{\mathrm{eff}}^{\mathrm{pri}} < \gamma_2$, where $\gamma_2 \ll 1$, the OOSM is simply discarded.

Remark 1. Since it is difficult to write an efficient batch-form solution, the smoothing and corresponding weight update is applied sequentially when several OOSMs arrive simultaneously.

1) Selective Processing: Another approach to handle mismatch between OOSMs and the particles was derived in [4] and [16]. The core of that approach is to use the mutual information metric or the Kullback-Leibler divergence metric to find out how informative the delayed measurement is in relation to the state. The mutual information between the measurement y_τ and the state x_k is

$$I(y_\tau, x_k|W_k^{1:k}, Z_{k,\bar{\tau}}) = \int \log\left(\frac{p(y_\tau, x_k|W_k^{1:k}, Z_{k,\bar{\tau}})}{p(y_\tau|W_k^{1:k}, Z_{k,\bar{\tau}})p(x_k|W_k^{1:k}, Z_{k,\bar{\tau}})}\right) \times p(y_\tau, x_k|W_k^{1:k}, Z_{k,\bar{\tau}})\mathrm{d}y_\tau \mathrm{d}x_k.$$

Because of this it is enough to find the joint distribution $p(y_\tau, x_k|W_k^{1:k}, Z_{k,\bar{\tau}})$, which is done by the Gaussian approximation

$$p(y_\tau, x_k|W_k^{1:k}, Z_{k,\bar{\tau}}) \approx \mathcal{N}\left(\begin{pmatrix} x_k \\ y_\tau \end{pmatrix}; \begin{pmatrix} \mu_{x_k} \\ \mu_{y_\tau} \end{pmatrix}, \begin{pmatrix} R_{x_k} & R_{x_k y_\tau} \\ R_{y_\tau x_k} & R_{y_\tau} \end{pmatrix}\right),$$

where μ_{x_k} is the saved state estimate at time index k, and $\mu_{y_\tau} = h(\mu_{x_\tau})$ The resulting mutual information can according to [4] be calculated as

$$I(y_\tau, x_k|W_k^{1:k}, Z_{k,\bar{\tau}}) = \frac{1}{2}\log\left(\frac{\|R_{x_k}\|}{\|R_{x_k} - R_{x_k y_\tau} R_{y_\tau}^{-1} R_{y_\tau x_k}\|}\right). \quad (16)$$

The involved covariance matrices and means are found through repeated EKF recursion on a system formed by augmenting the state x_k with the measurement y_τ. The filter runs from time t_τ to the current time, and is initialized with the estimated mean and covariance at time t_τ; that is, $\mu_{x_\tau} = \hat{x}_{\tau|\tau}$, $\mu_{y_\tau} = h(\mu_{x_\tau})$, $R_{x_\tau} = P_{\tau|\tau}$, $R_{y_\tau} = H_\tau P_{\tau|\tau} H_\tau^{\mathrm{T}} + Q_\tau$, and $R_{y_\tau x_\tau} = R_{x_\tau y_\tau}^{\mathrm{T}} = H_\tau R_{x_\tau}$.

If the resulting mutual information approximation is above some threshold γ_1 the SEPF from Section III-B is run. In the SEPF, if $N_{\mathrm{eff}}^{\mathrm{post}}/N_{\mathrm{eff}}^{\mathrm{pri}} < \gamma_2$ the SEPF is terminated and a particle filter running from time index $\tau - 1$ to the current time is activated, involving all (ordered) measurements from time index τ to time index k. This rerun particle filter, denoted Gaussian approximation rerun particle filter (OOSM-GARP), is initialized with the estimated mean and covariance at time index $\tau - 1$. On the other hand, if the mutual information is below γ_1 the OOSM is discarded. The thresholds γ_1 and γ_2 should be seen as governing the trade off between complexity and accuracy. The rerun particle filter is typically more computationally demanding than the SEPF. However, since SEPF is applied sequentially when several OOSMs arrive in a given time step, SEPF can actually become more time consuming than the rerun filter.

Remark 2. One drawback with the selective processing extension is that all measurements are assumed to occur at the sampling instants. This may be a severe restriction for large sampling times and/or fast motions. It is possible to remove the assumption but it will typically give rise to large computation times when the OOSM-GARP is invoked, prohibiting the algorithm from many applications where real-time performance is critical. Note that neither the SEPF in Section III-B nor our proposed algorithm in Section IV have this restriction.

IV. Storage Efficient Particle Filter with CISI-FPS

In [2] the fixed-point smoother was found to be the most computationally efficient; this is also the smoother used here.

The goal with fixed-point smoothing is to estimate the state at time t_j given data up to time $t_k > t_j$— that is, to estimate $\hat{x}_{j|k}$.

Assume that estimates $\hat{x}_{j|j}$, $j = k - l_{\max}, \ldots, k$ and their associated covariances $P_{j|j}$ are available. In addition, assume that the N particles $\hat{x}_k^{(i)}$ together with their weights, $w_k^{(i)}$, from the last measurement exist. Then the extension of the CISI-FPS algorithm works as follows: For each OOSM that arrives, start with the one with largest delay. The timestamp t_τ is bounded as $t_{k-l} \leq t_\tau < t_{k-l+1}$ for a positive integer l, where $1 \leq l \leq l_{\max}$. Iterating the dynamics forward by using (1) gives

$$\hat{x}_{\tau|k-l} = f_{\tau,k-l}(\hat{x}_{k-l|k-l}). \quad (17)$$

Furthermore, by calculating $F_{\tau,k-l} = \frac{\partial f_{\tau,k-l}(x)}{\partial x}\Big|_{x=\hat{x}_{k-l|k-l}}$,

the covariance matrix can be forward propagated as

$$P_{\tau|k-l} = F_{\tau,k-l} P_{k-l|k-l} F_{\tau,k-l}^{\mathrm{T}} + Q_{\tau,k-l}. \tag{18}$$

For $j = k-l+1, \ldots, k-1$, the mean and covariances are updated as

$$\hat{x}_{j|j,\tau} = \hat{x}_{j|j} + K_j(y_\tau - h_\tau(\hat{x}_{\tau|j})), \tag{19}$$
$$P_{j|j,\tau} = P_{j|j} - K_j S_\tau K_j^{\mathrm{T}}, \tag{20}$$

where $H_\tau = \frac{\partial h_\tau(x)}{\partial x}\Big|_{x=\hat{x}_{\tau|j}}$, $S_\tau = H_\tau P_{\tau|j} H_\tau^{\mathrm{T}} + R_\tau$, and $K_j = P_{j,\tau|j} H_\tau^{\mathrm{T}} S_\tau^{-1}$. $P_{j,\tau|j}$ is the crosscovariance between the states at time index j and τ. To calculate (19) and (20), the smoothed estimates and covariances as well as the crosscovariances are needed. If $j = k-l+1$ the quantities are given by

$$\hat{x}_{\tau|j} = \hat{x}_{\tau|j-1} + V_j P_{j|j-1}^{-1}(\hat{x}_{j|j} - \hat{x}_{j|j-1}), \tag{21}$$
$$P_{\tau|j} = P_{\tau|j-1} - V_j P_{j|j-1}^{-1}(P_{j|j-1} - P_{j|j}) P_{j|j-1}^{-1} V_j^{\mathrm{T}}, \tag{22}$$
$$P_{j,\tau|j} = P_{j|j} P_{j|j-1}^{-1} V_j^{\mathrm{T}}, \tag{23}$$
$$V_j = P_{\tau|j-1} F_{j,\tau}^{\mathrm{T}}. \tag{24}$$

For $j = k-l+2, \ldots, k-1$, the quantities are instead given by

$$\hat{x}_{\tau|j} = \hat{x}_{\tau|j-1} + V_j P_{j|j-1}^{-1}(\hat{x}_{j|j} - \hat{x}_{j|j-1}), \tag{25}$$
$$P_{\tau|j} = P_{\tau|j-1} - V_j P_{j|j-1}^{-1}(P_{j|j-1} - P_{j|j}) P_{j|j-1}^{-1} V_j^{\mathrm{T}}, \tag{26}$$
$$P_{j,\tau|j} = P_{j|j} P_{j|j-1}^{-1} V_j^{\mathrm{T}}, \tag{27}$$
$$V_j = P_{j-1,\tau|j-1} F_{j,j-1}^{\mathrm{T}}. \tag{28}$$

The approach of using the particles $\hat{x}_k^{(i)}$ as measurements is adopted. This means that at the last step, $j = k$, $\hat{x}_{\tau|j}$ is updated using Kalman smoother formulae, see [3], according to

$$\hat{x}_{\tau|k^{(i)}}^{(i)} = \hat{x}_{\tau|k-1} + K_k\left(\hat{x}_k^{(i)} - f_{k,k-1}(\hat{x}_{k-1|k-1})\right), \tag{29}$$
$$P_{\tau|j-1,k^{(i)}} = P_{\tau|j-1} - P_{j-1,\tau|j-1} H_j^T K_j^T, \tag{30}$$

for $i = 1, \ldots, N$.

For each j, linearization of (2) at the estimate $\hat{x}_{\tau|j}$ for (19) and (20) has to be performed. Furthermore, for $j = k-l+1$ linearization of (1) at $\hat{x}_{\tau|j}$ for use in (21)–(24) is necessary, and for $j = k-l+2, \ldots, k-1$ linearization of (1) at $\hat{x}_{j-1|j-1}$ (for (25)–(28)) has to be performed to propagate the covariances. In addition, for each step the inverses of S_τ and $P_{j|j-1}$ are needed, whereas the standard state-augmented Kalman smoother only needs the inverse of S_j. Also, a couple of extra matrix multiplications and additions are performed in each step, since the mean and associated covariances are also updated. For small state-space models an inversion is inexpensive to perform. Additionally, for small delays and/or large sample times the extra inversion is only done a few times at each time step. Hence the difference in computational speed should be small, for reasonable state-space models and time delays.

A summary of the resulting algorithm, denoted OOSM CISI-FPS, is found in Algorithm 2. Note that the difference when comparing with the method in Section III-B is in the implementation of the EKS, where now all estimates between the OOSM time and the current time are updated.

Algorithm 2. *OOSM CISI-FPS*

At time k, process in-sequence measurements according to Algorithm 1.
When n OOSMs arrive, sort them with longest delay first.
for $m = 1, \ldots, n$ ***do***
  Calculate $\hat{x}_{\tau|k-l}$ and $P_{\tau|k-l}$ using (17) and (18).
  for $j = k-l+1, \ldots, k-1$ ***do***
    if $j = k-l+1$ ***then***
      calculate $\hat{x}_{\tau|j}$, $P_{\tau|j}$, and $P_{j,\tau|j}$ using (21)–(24).
    else
      calculate $\hat{x}_{\tau|j}$, $P_{\tau|j}$, and $P_{j,\tau|j}$ using (25)–(28).
    end if
    Calculate $\hat{x}_{j|j,\tau}$ and $P_{j|j,\tau}$ using (19) and (20).
  end for
  for $i = 1, \ldots, N$ ***do***
    Calculate $\hat{x}_{\tau|k^{(i)}}^{(i)}$ and $P_{\tau|k^{(i)}}$ by applying (29)–(30).
    Apply (13)–(15) and update weights using (12).
  end for
  Calculate $N_{\mathrm{eff}}^{\mathrm{post}} = \frac{1}{\sum_i w^2}$.
  if $N_{\mathrm{eff}}^{\mathrm{post}} < \gamma N_{\mathrm{eff}}^{\mathrm{pri}}$ ***then***
    discard OOSM.
  end if
end for
Form the new estimates from (7) and (8).
Resample according to (9).

V. Numerical Results

We validate Algorithm 2 using root-mean-squared position and velocity errors, and compare it against a number of different particle filters. First, the simulation model is presented.

A. Simulation Model

The simulation model is similar to the ones used in [1] and [4]: A target moves in a plane, the motion being a turn of radius 500 m with constant velocity 200 km/h. The initial position is $p_0 = (-500 \quad 500)^{\mathrm{T}}$, and the motion lasts for 40 seconds. By introducing position, velocity, and turn rate as states, $x_k = (p_k^x \quad p_k^y \quad v_k^x \quad v_k^y \quad \omega_k)^{\mathrm{T}}$, the discrete-time model for the coordinated turn is

$$x_{k+1} = \begin{pmatrix} 1 & 0 & \frac{\sin(\omega_k T)}{\omega_k} & -\frac{1-\cos(\omega_k T)}{\omega_k} & 0 \\ 0 & 1 & \frac{1-\cos(\omega_k T)}{\omega_k} & \frac{\sin(\omega_k T)}{\omega_k} & 0 \\ 0 & 0 & \cos(\omega_k T) & -\sin(\omega_k T) & 0 \\ 0 & 0 & \sin(\omega_k T) & \cos(\omega_k T) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} x_k + v_{k+1,k}, \tag{31}$$

which in short is written as $x_{k+1} = f_{k+1,k}(x_k) + v_{k+1,k}$. The process noise is independent Gaussian, with zero mean and covariance $Q = \mathrm{diag}(30^2, 30^2, 10^2, 10^2, 0.1^2)$. To track the target three bearings-only sensors, common in military applications, send measurements to the fusion centre, with the sample time being $T = 1$ s. They are placed according to

$S_1 = (-200,\, 0)$, $S_2 = (200,\, 0)$, and $S_3 = (-750,\, 750)$. The measurement model for the bearings-only measurements is

$$h_k = \arctan\left(\frac{p_k^y - S_j^y}{p_k^x - S_j^x}\right) + e_k,\ j = 1, 2, 3.$$

The noise e_k is assumed independent Gaussian with zero mean and covariance $R = 0.05$. Sensors 2 and 3 are assumed to have serious communication issues, yielding OOSMs. For each of the two sensors, a measurement arrives with probability $p_{\text{oosm}} = 0.7$. To be able to compare the proposed Algorithm 2 with the selective processing algorithm in Section III-B1 the delay is modeled as a discrete uniform distribution in the interval $[0, 5]$. This should be interpreted as that each of the two sensors lose 30 % of the packages on the way to the communication centre, and those who arrive are delayed between zero and five seconds. Note that this scenario is of the type depicted in Fig. 2. The trajectory together with the sensors are shown in Fig. 3.

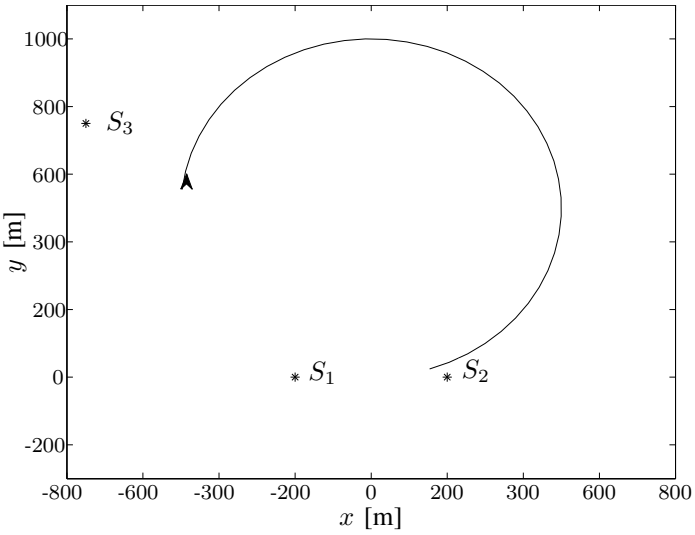


Fig. 3. The trajectory and the sensors used in the simulation example. The arrow indicates movement direction.

B. Results

Seven different particle filters were implemented in Matlab, all based on a standard SIR filter, see [17] and Section III-A. The code was highly optimized for all seven filters. The filters are:

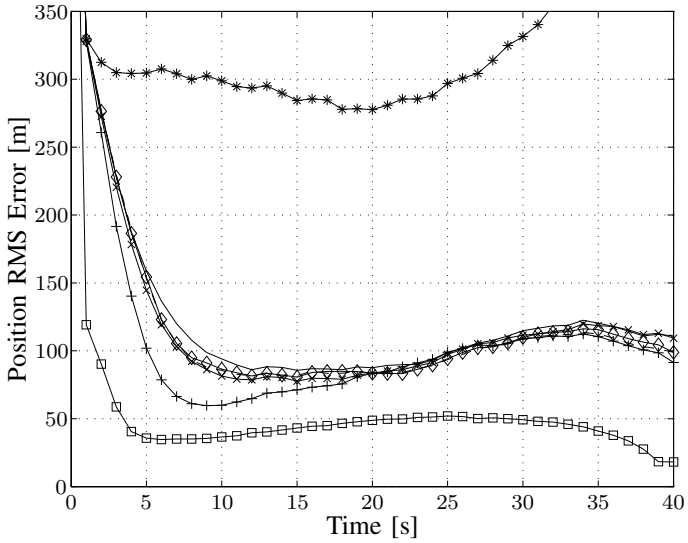
- PFideal: An idealized particle filter assuming zero delay. Better results than with this filter should not be achievable.
- PFdisc: A particle filter implementation that discards all the OOSMs. Thus, it only uses the measurements that are not delayed. This means that it processes sensor S_1 every sample, but only S_2 and S_3 when they arrive with zero delay. This filter is used to show the performance decrease when discarding the delayed measurements.
- SEPF: The storage efficient particle filter used in [1], see Section III-B. The Kalman smoother was implemented efficiently, using the formulas found in [3].
- SEPF-GARP: An implementation of the storage efficient particle filter with selective processing derived in [4], see Section III-B1.
- OOSM-GARP: The Gaussian approximation rerun particle filter described in [4] and in Section III-B1.
- CISI-FPSMI: The particle filter outlined in Algorithm 2 but with selective processing, similar to SEPF-GARP.
- CISI-FPS: The particle filter outlined in Algorithm 2, see Section IV.

An idea was to include EKF-OOSM algorithms for comparison; for example the one in [18], but since these algorithms only sporadically converged they are left out. A similar conclusion was made in [1]. The initial estimate for all filters was $x_0 = (0 \quad 0 \quad 0 \quad 0 \quad 0)^{\text{T}}$, with initial covariance set to $P_0 = \text{diag}(250^2,\, 250^2,\, 30^2,\, 30^2,\, 0.1^2)$. The measure used to compare the filters is the root-mean squared (RMS) error, which is defined as follows: Assume M Monte Carlo simulations, and denote the position error at time step k of the jth of M runs as $\text{err}_{k,j}$. Then the RMS position error at time index k is

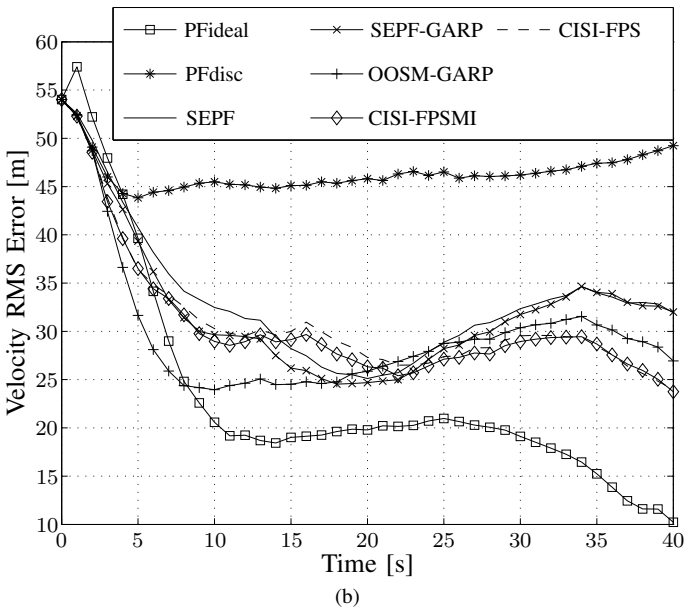
$$\text{RMS} = \sqrt{\frac{1}{M}\sum_{j=1}^{M} \|\text{err}_{k,j}\|^2}.$$

RMS position and velocity errors are shown in Figs. 4a-5b for 2000 Monte Carlo simulations. The threshold for when to discard the OOSM and when to run the GARP in SEPF-GARP, respectively, was set to $\gamma_1 = 0.05$ and $\gamma_2 = 0.025$. Filters SEPF and CISI-FPS used $\gamma_2 = 0.025$ for choosing when to discard the OOSMs. The number of particles was $N = 2000$ for Figs. 4a–4b, and $N = 20000$ for Figs. 5a–5b. Unsurprisingly PFideal performs best, with RMS errors decaying much faster than for the other filters except for some transient behavior in the velocity estimation. As expected PFdisc performs worst, with both position and velocity errors being large at all times. This serves as a reminder of the amount of information needed to perform satisfactory estimation. OOSM-GARP performs second best, using the ordered measurement data to improve performance. Also, the performance difference between SEPF and SEPF-GARP is significant, showing the benefits with selective processing.

CISI-FPS and CISI-FPSMI, however, have approximately the same performance. The reason for this can be found in Table I. As seen SEPF-GARP discards about 21 % of the OOSMs and runs approximately 0.4 % of the OOSMs through GARP. On the contrary, CISI-FPSMI discards almost the same number of OOSMs, but it only processes 0.06 % through GARP. We believe that the reason for this is that the CISI based Kalman smoother used in CISI-FPSMI and CISI-FPS is more robust than the Kalman smoother employed in SEPF-GARP and SEPF. The robustness is also seen when comparing CISI-FPS and SEPF: CISI-FPS only rejects approximately 0.1 % of the OOSMs, but SEPF rejects almost 1 %. These numbers indicate that the CISI based smoother creates more robust estimates,

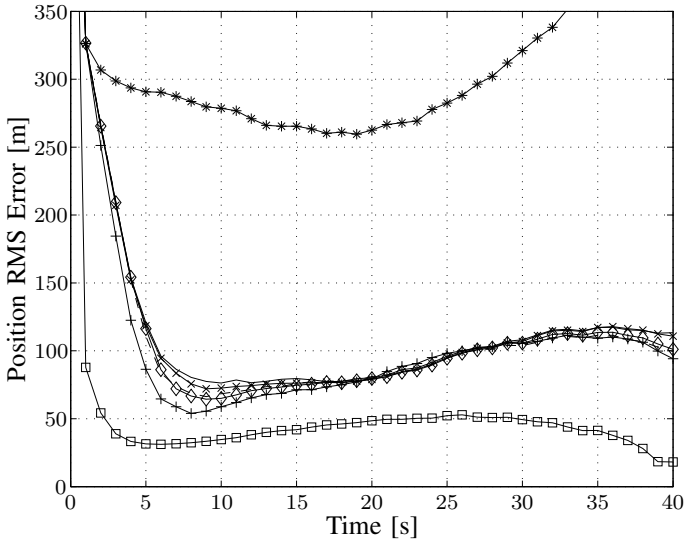


(a)

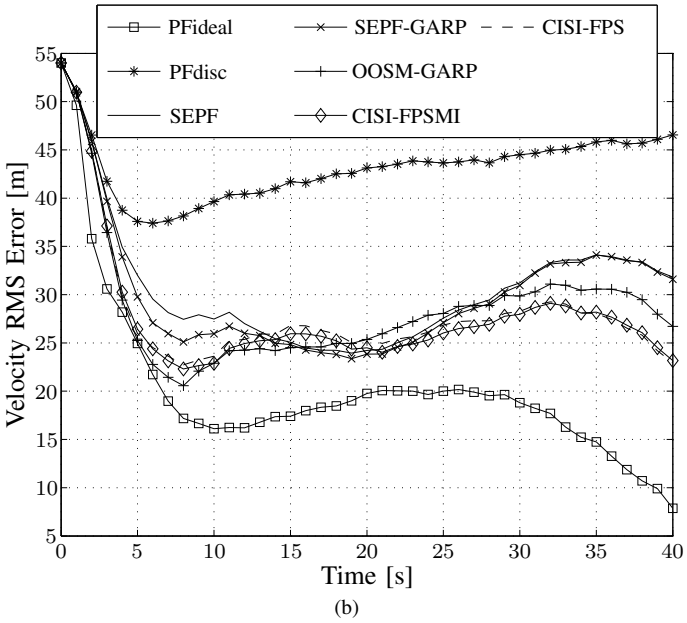


(b)

Fig. 4. Position (a), and velocity (b) tracking performance of the different particle filters, measured as the RMS error. The curves show the results after 2000 Monte Carlo simulations. The number of particles is 2000.



(a)



(b)

Fig. 5. Position (a) and velocity (b) tracking performance of the different particle filters, measured as the RMS error. The curves show the results after 2000 Monte Carlo simulations. The number of particles is 20000. Note that CISI-FPS and CISI-FPSMI have almost identical performance in (a), despite CISI-FPSMI being more complex.

correlating better with the delayed measurements. CISI-FPS performs almost as well as the more complex OOSM-GARP, SEPF-GARP, and CISI-FPSMI despite not having to store the measurements. When increasing the number of particles, as done in Fig. 5, CISI-FPS even performs better than SEPF-GARP in some cases. Also, the performance compared to SEPF is usually significantly better. In this example the only increase in storage requirements for SEPF, SEPF-GARP and CISI-FPSMI compared to CISI-FPS is a vector of five measurements, but it is easy to imagine setups where the increase is much larger.

As mentioned the Matlab code was highly optimized. Still, for several reasons a comparison between computation times should be interpreted with care. Mean computation times of the seven algorithms for a full run were measured using Matlab's built in functionality and are shown in Table II, in increasing order. As expected the algorithm that only processes the in-sequence measurements is the fastest, followed by PFideal. OOSM-GARP is the slowest algorithm, caused by that it runs a particle filter for several time steps as soon as an OOSM arrives. CISI-FPS is about 15 % slower than SEPF and SEPF-GARP. Note that CISI-FPSMI, which combines the CISI fixed-point smoother with the selective processing algorithm, is as fast as SEPF while maintaining the accuracy

TABLE I
THE PERCENTAGE OF THE OOSMS DISCARDED BY THE SELECTIVE PROCESSING LOGIC (γ_1), SEE SECTION III-B1, AND THE PERCENTAGE OF THE OOSMS EITHER DISCARDED (FOR CISI-FPS AND SEPF) OR RUN THROUGH THE RERUN PARTICLE FILTER (FOR SEPF-GARP AND CISI-FPSMI), RESPECTIVELY.

Algorithm	γ_1 [%]	γ_2 [%]
SEPF	-	0.9
SEPF-GARP	21.26	0.4
CISI-FPS	-	0.07
CISI-FPSMI	22.41	0.06

TABLE II
COMPARISON OF MEAN EXECUTION TIMES, MEASURED IN MILLISECONDS, FOR 2000 MONTE CARLO SIMULATIONS USING 2000 PARTICLES. THE TIME IS FOR RUNNING THE ALGORITHM ONCE. MATLAB'S TIC AND TOC FUNCTIONALITY WAS USED. THE SIMULATIONS WERE RUN ON A LENOVO LAPTOP WITH INTEL I5 M480 CPU 2.67 GHZ, 4 GB RAM AND LINUX FEDORA 15.

Algorithm	Time [ms]
PFdisc	2.3
PFideal	2.7
SEPF	5.3
CISI-FPSMI	5.5
SEPF-GARP	5.6
CISI-FPS	6
OOSM-GARP	7.3

of SEPF-GARP and CISI-FPS. In reality, however, the delay is typically not restricted to be a multiple of the sampling time; in that case SEPF-GARP and CISI-FPSMI will be slower.

Finally, inspecting the mean computation times clarifies that the number of particles used, $N = 2000$, is not unrealistic for real-time applications using any of the four smoothing based particle filters. However, if high sampling rate real-time applications are desired, SEPF or CISI-FPS should be chosen because of the large maximum computation time of SEPF-GARP and CISI-FPSMI, caused by the rerun filter.

VI. CONCLUSIONS AND FUTURE WORK

We presented an alternative to the storage efficient particle filter in [1] where not only the estimate at the last time step is updated, but also all estimates and covariances between the timestamp of the OOSM and the current time. The proposed CISI based smoother in combination with selective processing, CISI-FPSMI, proved to be the most viable algorithm in terms of mean computation time and estimation accuracy. However, if storage efficiency, estimation accuracy, and hard real-time constraints are critical the solution in Algorithm 2, CISI-FPS, is recommended. Even though the proposed Algorithm 2 performs as good as, or sometimes even better, than both SEPF and SEPF-GARP it is more storage efficient. It was shown that the computational load is only moderately larger, despite a clear performance gain.

Future works include implementing the algorithm in a real-world mobile robotics setting, where vision algorithms, wheel encoders, and an inertial measurement unit should be fused for localization of a mobile manipulator. Moreover, a rigorous complexity analysis would be beneficial.

ACKNOWLEDGMENTS

This work was supported by the Swedish Foundation for Strategic Research through the project ENGROSS, by the Swedish Research Council through the LCCC Linnaeus Center, and by the ELLIIT Excellence Center.

REFERENCES

[1] U. Orguner and F. Gustafsson, "Storage efficient particle filters for the out of sequence measurement problem," in *11th International Conference on Information Fusion, 2008*, June 30-July 3 2008, pp. 1–8.
[2] S. Zhang and Y. Bar-Shalom, "Optimal update with multiple out-of-sequence measurements," *IEEE Transactions on Signal Processing*, April 2011.
[3] K. Biswas and A. Mahalanabis, "Suboptimal algorithms for nonlinear smoothing," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-9, no. 4, pp. 529–534, July 1973.
[4] X. Liu, B. Oreshkin, and M. Coates, "Efficient delay-tolerant particle filtering through selective processing of out-of-sequence measurements," in *13th International Conference on Information Fusion, 2010*, July 2010, pp. 1–8.
[5] S. Blackman and R. Popoli, *Design and analysis of modern tracking systems*, ser. Artech House radar library. Artech House, 1999.
[6] Y. Bar-Shalom, "Update with out-of-sequence measurements in tracking: exact solution," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 3, pp. 769 – 777, July 2002.
[7] Y. Bar-Shalom, H. Chen, and M. Mallick, "One-step solution for the multistep out-of-sequence-measurement problem in tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 1, pp. 27–37, Jan. 2004.
[8] S. Zhang, Y. Bar-Shalom, and G. Watson, "Tracking with multisensor out-of-sequence measurements with residual biases," in *13th International Conference on Information Fusion, 2010*, July 2010, pp. 1–8.
[9] K. Zhang, X. Li, and Y. Zhu, "Optimal update with out-of-sequence measurements," *IEEE Transactions on Signal Processing*, vol. 53, no. 6, pp. 1992 – 2004, June 2005.
[10] X. Shen, Y. Zhu, E. Song, and Y. Luo, "Optimal centralized update with multiple local out-of-sequence measurements," *IEEE Transactions on Signal Processing*, vol. 57, no. 4, pp. 1551 –1562, April 2009.
[11] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*. Prentice Hall, 1979.
[12] M. Orton and A. Marrs, "A Bayesian approach to multi-target tracking and data fusion with out-of-sequence measurements," *IEE Seminar Digests*, vol. 2001, no. 174, 2001.
[13] ——, "Particle filters for tracking with out-of-sequence measurements," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 2, pp. 693 – 702, April 2005.
[14] M. Mallick, T. Kirubarajan, and S. Arulampalam, "Out-of-sequence measurement processing for tracking ground target using particle filters," in *Aerospace Conference Proceedings, 2002. IEEE*, vol. 4, 2002, pp. 4–1809 – 4–1818 vol.4.
[15] M. Mallick and A. Marrs, "Comparison of the KF and particle filter based out-of-sequence measurement filtering algorithms," in *Proceedings of the Sixth International Conference of Information Fusion, 2003*, vol. 1, 2003, pp. 422 – 429.
[16] B. Oreshkin, X. Liu, and M. Coates, "Efficient delay-tolerant particle filtering," *IEEE Transactions on Signal Processing*, vol. 59, no. 7, pp. 3369–3381, July 2011.
[17] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, Feb 2002.
[18] K. Berntorp, K.-E. Årzén, and A. Robertsson, "Sensor Fusion for Motion Estimation of Mobile Robots with Compensation for Out-of-Sequence Measurements," *11th International Conference on Control, Automation, and Systems*, October 2011.