

Research Article

Log-PF: Particle Filtering in Logarithm Domain

Christian Gentner , Siwei Zhang , and Thomas Jost 

German Aerospace Center (DLR), Institute of Communications and Navigation, Oberpfaffenhofen, 82234 Wessling, Germany

Correspondence should be addressed to Christian Gentner; christian.gentner@dlr.de

Received 25 August 2017; Revised 7 November 2017; Accepted 6 December 2017; Published 1 March 2018

Academic Editor: Víctor Elvira

Copyright © 2018 Christian Gentner et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper presents a particle filter, called Log-PF, based on particle weights represented on a logarithmic scale. In practical systems, particle weights may approach numbers close to zero which can cause numerical problems. Therefore, calculations using particle weights and probability densities in the logarithmic domain provide more accurate results. Additionally, calculations in logarithmic domain improve the computational efficiency for distributions containing exponentials or products of functions. To provide efficient calculations, the Log-PF exploits the Jacobian logarithm that is used to compute sums of exponentials. We introduce the weight calculation, weight normalization, resampling, and point estimations in logarithmic domain. For point estimations, we derive the calculation of the minimum mean square error (MMSE) and maximum a posteriori (MAP) estimate. In particular, in situations where sensors are very accurate the Log-PF achieves a substantial performance gain. We show the performance of the derived Log-PF by three simulations, where the Log-PF is more robust than its standard particle filter counterpart. Particularly, we show the benefits of computing all steps in logarithmic domain by an example based on Rao-Blackwellization.

1. Introduction

Many scientific problems involve dynamic systems, for example, in navigation applications. Dynamic systems can be described by state-space models where the state is only observable by noisy measurements. Recursive Bayesian filters are algorithms to estimate an unknown probability density function (PDF) of the state recursively by measurements over time. Such a filter consists of two steps: prediction and update. In the prediction step, the PDF of the state is calculated based on the system model. During the update step, the current measurement is used to correct the prediction based on the measurement model. In this way, the posterior PDF of the state is estimated recursively over time. Particle filters (PFs) are implementations of recursive Bayesian filters which approximate the posterior PDF by a set of random samples, called particles, with associated weights. Several types of PFs have been developed over the last few years [1–8]. They differ in their choice of the importance sampling density and the resampling step.

A common way is to choose the importance sampling density to be equal to the prior, for example, the bootstrap filtering algorithm [1]. However, if the width of the likelihood

distribution is too small in comparison to the width of the prior distribution or if measurements are located in the tail of the prior distribution, this choice may fail; see [4]. These situations may arise when sensors are very accurate or measurements rapidly change over time such that the particle states after the prediction step might be located in the tail of the likelihood. Additionally, numerical representation of numbers may limit the computational accuracy by floating point errors. In these situations, a common way is to use the likelihood particle filter (LPF) [3, 5]. The LPF uses the likelihood distribution for the importance sampling density and the prior for the weight update. The LPF is recommended when the width of the likelihood distribution is much smaller compared to the one of the prior and accordingly, the posterior density function is more similar to the likelihood than to the prior. However, in many situations, it is impossible to draw samples from the likelihood distribution. Furthermore, the LPF is not suitable for an underdetermined system where the number of measurements is lower than the number of states per time instant. Additionally, using the likelihood as proposal distribution might increase the variance of the simulated samples according to [5].

In this paper, we derive a PF that operates in logarithmic domain (log-domain), called Log-PF. The Log-PF represents the weights in log-domain which enables a more accurate representation of low weights with a limited number of bits. Particularly, when the involved distributions contain exponentials or products of functions, the log-domain representation is computationally more efficient [9]. The derived Log-PF uses the Jacobian logarithm [10–12] to describe all steps of the PF, including weight update, weight normalization, resampling, and point estimations in log-domain. In this paper, we derive the minimum mean square error (MMSE) and the maximum a posteriori (MAP) point estimators.

The paper is structured as follows: First, we describe in Section 2 standard PFs; thereafter we derive the proposed Log-PF in Section 2. Afterwards, we derive in Section 4 two point estimators in log-domain: Section 4.1 describes the MMSE estimator and Section 4.2 the MAP estimator. We evaluate the Log-PF by simulations and compare the results to standard PF implementations and Kalman filters (KFs) in Section 5. Particularly, we show by an example based on Rao-Blackwellization the benefits by computing all steps in log-domain. For distributed particle filters like [13] similar results are expected. Finally, Section 6 concludes the paper. Throughout the paper, we will use the following notations:

- (i) All vectors are interpreted as column vectors.
- (ii) \mathbf{I} denotes an identity matrix.
- (iii) Matrices are denoted by bold capital letters and vectors by bold small letters.
- (iv) $[\mathbf{x}]_l$ denotes the l th element of vector \mathbf{x} .
- (v) $a \sim \mathcal{N}(\mu_a, \sigma_a^2)$ denotes a Gaussian distributed or multivariate random variable a with mean μ_a and variance σ_a^2 .
- (vi) $\mathbf{E}[x]$ stands for expectation or sample mean of x .
- (vii) $1 : k$ stands for all integer numbers starting from 1 to k , thus $1, 2, \dots, k$.
- (viii) \hat{x} denotes the estimate of x .
- (ix) $\{x^{(i)}\}_{i=1}^N$ defines the set for x_i with $i = 1, \dots, N$.

2. Particle Filtering

PFs represent the probability density of the state vector \mathbf{x}_k at time step k by N_p particles. According to [1–3, 5] and assuming a first-order hidden Markov model, the posterior filtered density $p(\mathbf{x}_k | \mathbf{z}_{0:k})$ is approximated as

$$p(\mathbf{x}_k | \mathbf{z}_{0:k}) \approx \sum_{j=1}^{N_p} w_k^{(j)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(j)}), \quad (1)$$

where $\mathbf{z}_{0:k}$ defines the measurement vector for the time steps $0, \dots, k$, $\delta(\cdot)$ stands for the Dirac distribution, $\mathbf{x}_k^{(j)}$ denotes particle state, and $w_k^{(j)}$ denotes the normalized weight with

$$w_k^{(j)} = \frac{w_k^{*(j)}}{\sum_{i=1}^{N_p} w_k^{*(i)}}. \quad (2)$$

The unnormalized weight is denoted by $w_k^{*(j)}$, while the weight update is calculated as

$$w_k^{*(j)} = w_{k-1}^{(j)} \frac{p(\mathbf{z}_k | \mathbf{x}_k^{(j)}) p(\mathbf{x}_k^{(j)} | \mathbf{x}_{k-1}^{(j)})}{q(\mathbf{x}_k^{(j)} | \mathbf{x}_{k-1}^{(j)}, \mathbf{z}_k)}, \quad (3)$$

with the importance density $q(\mathbf{x}_k | \mathbf{x}_{k-1}^{(j)}, \mathbf{z}_k)$, the likelihood distribution $p(\mathbf{z}_k | \mathbf{x}_k^{(j)})$, and the transition prior distribution $p(\mathbf{x}_k^{(j)} | \mathbf{x}_{k-1}^{(j)})$; see [1–3, 5]. For $N_p \rightarrow \infty$, the approximation used in (33) approaches $p(\mathbf{x}_k | \mathbf{z}_{0:k})$. By (33), (2), and (3), the sequential importance sampling (SIS) PF can be described which is the basis of most PFs [2].

For numerical stability reasons, weights are often computed and stored in the log-domain, which is also computationally efficient when the distributions involved contain exponentials or products. Thus, we obtain from (3) the update equation in log-domain, with

$$\begin{aligned} \widehat{w}_k^{*(j)} &= \widehat{w}_{k-1}^{(j)} + \ln(p(\mathbf{z}_k | \mathbf{x}_k^{(j)})) + \ln(p(\mathbf{x}_k^{(j)} | \mathbf{x}_{k-1}^{(j)})) \\ &\quad - \ln(q(\mathbf{x}_k | \mathbf{x}_{k-1}^{(j)}, \mathbf{z}_k)), \end{aligned} \quad (4)$$

where we define with

$$\widehat{w}_k^{(j)} = \ln(w_k^{(j)}) \quad (5)$$

the log-domain weight (log-weight) $\widehat{w}_k^{(j)}$ for particle j . After calculating the weights $\widehat{w}_k^{*(j)}$ in log-domain, the weights are transferred for further processing to the linear domain (lin-domain) with $w_k^{*(j)} = e^{\widehat{w}_k^{*(j)}}$ for $j = 1, \dots, N_p$, where the numerical accuracy is lost due to floating point representation. In order to obtain a more stable PF implementation, the weights $\widehat{w}_k^{*(j)}$ can be transferred to the lin-domain by

$$w_k^{+(j)} = e^{\widehat{w}_k^{*(j)} - \max_i(\widehat{w}_k^{*(i)})}, \quad (6)$$

such that $w_k^{(j)} = w_k^{+(j)} / \sum_{i=1}^{N_p} w_k^{+(i)}$; see, for example, [9].

In the following we investigate a different approach, where the transformation from the log-domain to the lin-domain is not necessary. Hence, we show that all steps of the PF can be computed in log-domain.

3. Algorithm Derivation

To compute all steps of the PF in log-domain, we obtain for the approximation of the posterior filtered density from (33)

$$p(\mathbf{x}_k | \mathbf{z}_{0:k}) \approx \sum_{j=1}^{N_p} e^{\widehat{w}_k^{(j)}} \delta(\mathbf{x}_k - \mathbf{x}_k^{(j)}) \quad (7)$$

using (5). The normalization of the log-weight can be calculated directly in log-domain as a simple subtraction, with

$$\widehat{w}_k^{(j)} = \widehat{w}_k^{*(j)} - \widehat{W}_k, \quad (8)$$

$\ln(\sum_{i=1}^n e^{\delta_i}) = \text{Jacob}(\{\delta_l\}_{l=1}^n)$

(1) Init: $\Delta_1 = \delta_1$;
(2) **for** $l = 2 : n$ **do**
(3) $\Delta_l = \max(\delta_l, \Delta_{l-1}) + \ln(1 + e^{-|\delta_l - \Delta_{l-1}|})$;
(4) $\ln(\sum_{i=1}^n e^{\delta_i}) = \Delta_n$;

ALGORITHM 1: Iterative Jacobian algorithm.

where \widehat{W}_k denotes the normalization factor with

$$\widehat{W}_k = \ln \left(\sum_{i=1}^{N_p} e^{\widehat{w}_k^{*(i)}} \right). \quad (9)$$

To compute the normalization factor \widehat{W}_k of (9) without transferring the log-weights to the lin-domain, the Jacobian logarithm [10, 11] can be used. The Jacobian logarithm computes the logarithm of a sum of two exponentials $\ln(e^{\delta_1} + e^{\delta_2})$ using the $\max(\cdot)$ operator and adding a correction term; that is,

$$\ln(e^{\delta_1} + e^{\delta_2}) = \max(\delta_1, \delta_2) + \ln(1 + e^{-|\delta_2 - \delta_1|}). \quad (10)$$

With (10) and as derived in [12], the expression $\ln(\sum_{l=1}^n e^{\delta_l})$ can be calculated iteratively as

$$\begin{aligned} \ln(e^{\delta_1} + \dots + e^{\delta_n}) &= \ln(\Delta + e^{\delta_n}) \\ &= \max(\ln(\Delta), \delta_n) \\ &\quad + \ln(1 + e^{-|\ln(\Delta) - \delta_n|}), \end{aligned} \quad (11)$$

where $\delta = \ln(e^{\delta_1} + \dots + e^{\delta_{n-1}})$ and $\Delta = e^{\delta_1} + \dots + e^{\delta_{n-1}}$. Hence, using the Jacobian logarithm allows computing operations such as summations like in (9) efficiently in the log-domain. For later conveniences, we express (11) by an iterative algorithm shown in Algorithm 1 by a pseudocode; that is,

$$\ln \left(\sum_{l=1}^n e^{\delta_l} \right) = \text{Jacob} \left(\{\delta_l\}_{l=1}^n \right), \quad (12)$$

where $\{\delta_l\}_{l=1}^n$ defines the set for δ_l with $l = 1, \dots, n$. Thus, the normalization factor \widehat{W}_k of (9) can be calculated iteratively by

$$\widehat{W}_k = \text{Jacob} \left(\{\widehat{w}_k^{*(i)}\}_{i=1}^{N_p} \right). \quad (13)$$

Hence, we obtain for the log-weight normalization of (8),

$$\widehat{w}_k^{(j)} = \widehat{w}_k^{*(j)} - \text{Jacob} \left(\{\widehat{w}_k^{*(i)}\}_{i=1}^{N_p} \right). \quad (14)$$

Please note that a complexity reduction can be obtained if the term $\ln(1 + e^{-|\delta_l - \Delta_{l-1}|})$ of Algorithm 1 is read from a hard-coded lookup table as a function of $|\delta_l - \Delta_{l-1}|$.

By using (14), the SIS PF can be described in log-domain as shown in Algorithm 2 by a pseudocode. Algorithm 2 is

$\{\mathbf{x}_k^{(j)}, \widehat{w}_k^{(j)}\}_{j=1}^{N_p} = \text{Log-SIS}(\{\mathbf{x}_{k-1}^{(j)}, \widehat{w}_{k-1}^{(j)}\}_{j=1}^{N_p})$

(1) **for** $j = 1 : N_p$ **do**
(2) Draw: $\mathbf{x}_k^{(j)} \sim q(\mathbf{x}_k^{(j)} | \mathbf{x}_{k-1}^{(j)}, \mathbf{z}_k)$;
(3) Calculate: $\widehat{w}_k^{*(j)}$ according to (4);
(4) Calculate: $\widehat{W}(k) = \text{Jacob}(\{\widehat{w}_k^{*(i)}\}_{i=1}^{N_p})$;
(5) **for** $j = 1 : N_p$ **do**
(6) Normalize: $\widehat{w}_k^{(j)} = \widehat{w}_k^{*(j)} - \widehat{W}(k)$;

ALGORITHM 2: SIS-PF in log-domain (SIS Log-PF).

evaluated at each time step k , where $\{\mathbf{x}_k^{(j)}, \widehat{w}_k^{(j)}\}_{j=1}^{N_p}$ denotes the set for the particle states $\mathbf{x}_k^{(j)}$ and log-weights $\widehat{w}_k^{(j)}$ with $j = 1, \dots, N_p$ for time step k .

One of the crucial problems of the SIS PF is degeneracy (another problem which is not discussed in this paper is the selection of the importance density $q(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{z}_k)$ of (4); see e.g., [2]).

After a few time steps all particles except for one have low weights and do not contribute anymore to the computation of the posterior PDF; that is, the distribution estimation degenerates. A suitable measure of degeneracy is the effective sample size N_{eff} [1–3, 5]. A widely used approximation for the effective sample size is

$$N_{\text{eff}} \approx P_{N_p}^{(2)}(\mathbf{w}_k) = \frac{1}{\sum_{j=1}^{N_p} (w_k^{(j)})^2} = \frac{\left(\sum_{j=1}^{N_p} (w_k^{*(j)}) \right)^2}{\sum_{j=1}^{N_p} (w_k^{*(j)})^2} \quad (15)$$

with $\mathbf{w}_k = (w_k^{(1)}, \dots, w_k^{(N_p)})^T$ and $1 \leq P_{N_p}^{(2)}(\mathbf{w}_k) \leq N_p$. A small value of $P_{N_p}^{(2)}(\mathbf{w}_k)$ indicates a severe degeneracy. By using the Jacobian logarithm of (12), we obtain from (15) the effective sample size in log-domain, with

$$\begin{aligned} \ln(N_{\text{eff}}) &\approx \ln \left(P_{N_p}^{(2)}(\mathbf{w}_k) \right) = -\ln \left(\sum_{j=1}^{N_p} e^{2 \cdot \widehat{w}_k^{(j)}} \right) \\ &= -\text{Jacob} \left(\{2 \cdot \widehat{w}_k^{(j)}\}_{j=1}^{N_p} \right) \\ &= 2 \cdot \text{Jacob} \left(\{\widehat{w}_k^{(j)}\}_{j=1}^{N_p} \right) \\ &\quad - \text{Jacob} \left(\{2 \cdot \widehat{w}_k^{*(j)}\}_{j=1}^{N_p} \right). \end{aligned} \quad (16)$$

Alternative effective sample size approximations as introduced in [14] can also be represented in log-domain. Table 1 summarizes four generalized alternative effective sample size approximations in lin-domain and log-domain which depend on the parameter r . Please note $P_{N_p}^{(2)}(\mathbf{w}_k)$ as in (15) is obtained from $P_{N_p}^{(r)}(\mathbf{w}_k)$ with $r = 2$.

The Generic PF extends the SIS PF by a resampling step to prevent degeneration as shown in Algorithm 3 by a pseudocode. The basic idea of resampling is to eliminate

TABLE 1: Generalized effective sample size functions $P_{N_p}^{(r)}(\mathbf{w}_k)$, $D_{N_p}^{(r)}(\mathbf{w}_k)$, $V_{N_p}^{(r)}(\mathbf{w}_k)$, $S_{N_p}^{(r)}(\mathbf{w}_k)$ to approximate N_{eff} in lin-domain as defined in [14] and log-domain with their coefficients $a_{\{\cdot\}}^{(r)}$ and $b_{\{\cdot\}}^{(r)}$. Please note $P_{N_p}^{(2)}(\mathbf{w}_k)$ as in (15) is obtained from $P_{N_p}^{(r)}(\mathbf{w}_k)$ with $r = 2$.

$P_{N_p}^{(r)}(\mathbf{w}_k) = \frac{1}{a_p^{(r)} \sum_{j=1}^{N_p} (w_k^{(j)})^r + b_p^{(r)}}$	$a_p^{(r)} = \frac{1 - N_p}{N_p^{2-r} - N_p}$
$\ln(P_{N_p}^{(r)}(\mathbf{w}_k)) = -\text{Jacob}\left(\left\{\left\{\ln(a_p^{(r)}) + r \cdot \widehat{w}_k^{(j)}\right\}_{j=1}^{N_p}, \ln(b_p^{(r)})\right\}\right)$	$b_p^{(r)} = \frac{N_p^{2-r} - 1}{N_p^{2-r} - N_p}$
$D_{N_p}^{(r)}(\mathbf{w}_k) = \frac{1}{a_D^{(r)} \left[\sum_{j=1}^{N_p} (w_k^{(j)})^r\right]^{1/r} + b_D^{(r)}}$	$a_D^{(r)} = \frac{N_p - 1}{N_p - N_p^{1/r}}$
$\ln(D_{N_p}^{(r)}(\mathbf{w}_k)) = -\text{Jacob}\left(\left\{\ln(a_D^{(r)}) + \frac{1}{r} \text{Jacob}\left(\left\{r \cdot \widehat{w}_k^{(j)}\right\}_{j=1}^{N_p}\right), \ln(b_D^{(r)})\right\}\right)$	$b_D^{(r)} = \frac{1 - N_p^{1/r}}{N_p - N_p^{1/r}}$
$V_{N_p}^{(r)}(\mathbf{w}_k) = a_V^{(r)} \sum_{j=1}^{N_p} (w_k^{(j)})^r + b_V^{(r)}$	$a_V^{(r)} = \frac{N_p^{r-1}(N-1)}{1 - N_p^{r-1}}$
$\ln(V_{N_p}^{(r)}(\mathbf{w}_k)) = \text{Jacob}\left(\left\{\left\{\ln(a_V^{(r)}) + r \cdot \widehat{w}_k^{(j)}\right\}_{j=1}^{N_p}, \ln(b_V^{(r)})\right\}\right)$	$b_V^{(r)} = \frac{N_p^r - 1}{N_p^{r-1} - 1}$
$S_{N_p}^{(r)}(\mathbf{w}_k) = a_S^{(r)} \left[\sum_{j=1}^{N_p} (w_k^{(j)})^r\right]^{1/r} + b_S^{(r)}$	$a_S^{(r)} = \frac{N_p - 1}{N_p^{(1-r)/r} - 1}$
$\ln(S_{N_p}^{(r)}(\mathbf{w}_k)) = \text{Jacob}\left(\left\{\ln(a_S^{(r)}) + \frac{1}{r} \text{Jacob}\left(\left\{r \cdot \widehat{w}_k^{(j)}\right\}_{j=1}^{N_p}\right), \ln(b_S^{(r)})\right\}\right)$	$b_S^{(r)} = \frac{N_p^{(1-r)/r} - N_p}{(1-r)/r - 1}$

$\{\mathbf{x}_k^{(j)}, \widehat{w}_k^{(j)}\}_{j=1}^{N_p} = \text{Generic-Log}(\{\mathbf{x}_{k-1}^{(j)}, \widehat{w}_{k-1}^{(j)}\}_{j=1}^{N_p})$

- (1) **for** $j = 1 : N_p$ **do**
- (2) Draw: $\mathbf{x}_k^{(j)} \sim \mathbf{q}(\mathbf{x}_k^{(j)} | \mathbf{x}_{k-1}^{(j)}, \mathbf{z}_k)$;
- (3) Calculate: $\widehat{w}_k^{(j)}$ according to (4);
- (4) Calculate: $\widehat{W}_k = \text{Jacob}(\{\widehat{w}_k^{(j)}\}_{j=1}^{N_p})$;
- (5) **for** $j = 1 : N_p$ **do**
- (6) Normalize: $\widehat{w}_k^{(j)} = \widehat{w}_k^{(j)} - \widehat{W}_k$;
- (7) Calculate $\ln(N_{\text{eff}})$ according to Table 1;
- (8) **if** $\ln(N_{\text{eff}}) < \ln(N_{\text{thr}})$ **then**
- (9) Resample with Algorithm 4: Obtaining $\{\mathbf{x}_k^{(j)}, \widehat{w}_k^{(j)}\}_{j=1}^{N_p}$;

ALGORITHM 3: Generic-PF in log-domain (Generic Log-PF).

particles with low weights and reproduce particles with high weights. Whenever a significant degeneracy is observed in the Generic PF, that is, $\ln(N_{\text{eff}})$ is less than a threshold $\ln(N_{\text{thr}})$, the particles are resampled. Algorithm 4 shows a pseudocode of the systematic resampling algorithm [15] transferred into log-domain. In Algorithm 4, $\mathcal{U}[0, N_p^{-1}]$ denotes the uniform distribution on the interval $[0, N_p^{-1}]$ (cf. Algorithm 4 Line 5). Similarly to the descriptions before, the Jacobian logarithm is used to construct the estimated sampled cumulative distribution function in log-domain (log-CDF); see Algorithm 4 Line 3. The estimated sampled log-CDF is presented by a vector \mathbf{c}

$\{\bar{\mathbf{x}}_k^{(j)}, \widehat{w}_k^{(j)}\}_{j=1}^{N_p} = \text{Log-Resampling}(\{\mathbf{x}_k^{(j)}, \widehat{w}_k^{(j)}\}_{j=1}^{N_p})$

- (1) Initialize the log-CDF: $[\mathbf{c}]_1 = \widehat{w}_k^{(1)}$;
- (2) **for** $j = 2 : N_p$ **do**
- (3) Construct log-CDF using the Jacobian logarithm: $[\mathbf{c}]_j = \max(\widehat{w}_k^{(j)}, [\mathbf{c}]_{j-1}) + \ln(1 + e^{-|[\mathbf{c}]_{j-1} - \widehat{w}_k^{(j)}|})$;
- (4) $i = 1$;
- (5) Draw starting point: $[\mathbf{u}]_1 \sim \mathcal{U}[0, N_p^{-1}]$;
- (6) **for** $j = 1 : N_p$ **do**
- (7) $[\mathbf{u}]_j = \ln([\mathbf{u}]_1 + N_p^{-1}(j-1))$;
- (8) **while** $[\mathbf{u}]_j > [\mathbf{c}]_i$ **do**
- (9) $i = i + 1$;
- (10) Assign: $\{\bar{\mathbf{x}}_k^{(j)}, \widehat{w}_k^{(j)}\} = \{\mathbf{x}_k^{(i)}, -\ln(N_p)\}$;

ALGORITHM 4: Resampling in log-domain.

with length N_p and element $[\mathbf{c}]_j$ with $j = 1, \dots, N_p$. By $[\mathbf{x}]_j$, we denote the j th element of the vector \mathbf{x} . According to the estimated sampled log-CDF, particles with high weights are reproduced and particles with low weights are eliminated.

In Section 5, we use the sequential importance resampling (SIR) PF; see [1], as an example for comparing the performance of the linear domain PF (Lin-PF) and Log-PF. Therefore, Algorithm 5 shows a pseudocode of the SIR PF in log-domain, which is derived from the Generic PF by setting the importance density to be equal to the transitional prior

```

 $\{\mathbf{x}_k^{(j)}, \widehat{w}_k^{(j)}\}_{j=1}^{N_p} = \text{Log-SIR}(\{\mathbf{x}_{k-1}^{(j)}, \widehat{w}_{k-1}^{(j)}\}_{j=1}^{N_p})$ 
(1) for  $j = 1 : N_p$  do
(2)   Draw:  $\mathbf{x}_k^{(j)} \sim p(\mathbf{x}_k^{(j)} | \mathbf{x}_{k-1}^{(j)});$ 
(3)   Calculate:  $\widehat{w}_k^{*(j)} = \ln(p(\mathbf{z}_k | \mathbf{x}_k^{(j)}));$ 
(4) Calculate:  $\widehat{W}_k = \text{Jacob}(\{\widehat{w}_k^{*(j)}\}_{j=1}^{N_p});$ 
(5) for  $j = 1 : N_p$  do
(6)   Normalize:  $\widehat{w}_k^{(j)} = \widehat{w}_k^{*(j)} - \widehat{W}_k;$ 
(7) Resample with Algorithm 4: Obtaining
 $\{\mathbf{x}_k^{(j)}, \widehat{w}_k^{(j)}\}_{j=1}^{N_s};$ 

```

ALGORITHM 5: SIR-PF in log-domain (SIR Log-PF).

```

 $\{\mathbf{x}_k^{(j)}, w_k^{(j)}\}_{j=1}^{N_p} = \text{Generic-Lin-Log}(\{\mathbf{x}_{k-1}^{(j)}, w_{k-1}^{(j)}\}_{j=1}^{N_p})$ 
(1) for  $j = 1 : N_p$  do
(2)   Draw:  $\mathbf{x}_k^{(j)} \sim q(\mathbf{x}_k^{(j)} | \mathbf{x}_{k-1}^{(j)}, \mathbf{z}_k);$ 
(3)   Calculate:  $\widehat{w}_k^{*(j)}$  according to (4);
(4) for  $j = 1 : N_p$  do
(5)   Transfer and Normalize:
 $w_k^{+(j)} = e^{\widehat{w}_k^{*(j)} - \max_l(\widehat{w}_k^{*(l)})};$ 
(6) Calculate:  $W_k = \sum_{j=1}^{N_p} w_k^{+(j)};$ 
(7) for  $j = 1 : N_p$  do
(8)   Normalize:  $w_k^{(j)} = w_k^{+(j)} / W_k;$ 
(9) Calculate  $N_{\text{eff}}$  according to Table 1;
(10) if  $N_{\text{eff}} < N_{\text{thr}}$  then
(11)   Resample with Algorithm 4: Obtaining
 $\{\mathbf{x}_k^{(j)}, w_k^{(j)}\}_{j=1}^{N_s};$ 

```

ALGORITHM 6: Generic-PF in lin-domain with weight calculation in log-domain (Generic Lin-Log-PF).

distribution, with $q(\mathbf{x}_k | \mathbf{x}_{k-1}^{(j)}, \mathbf{z}_k) = p(\mathbf{x}_k^{(j)} | \mathbf{x}_{k-1}^{(j)})$ and using $N_{\text{eff}} = 1$, that is, performing resampling at each time step [2].

Additionally, we compare in Section 5 the proposed Log-PF to the PF implementation which computes the weights in log-domain and uses (6) to obtain the weights in lin-domain, called Lin-Log-PF in the following. A pseudocode of the Generic Lin-Log-PF is shown in Algorithm 6: the weights are calculated in log-domain according to (4) and normalized and transferred to the lin-domain according to $w_k^{+(j)} = e^{\widehat{w}_k^{*(j)} - \max_l(\widehat{w}_k^{*(l)})}$ as mentioned in Section 3 and, for example, [9]. Please note further improvements can be obtained if the weights $\widehat{w}_k^{*(j)}$ are directly propagated in log-domain if resampling is not necessary.

4. Log-PF Point Estimators

In many applications, we are interested in a point estimate of the state instead of its a posteriori PDF. In this section we derive the MMSE and MAP point estimators based on the a posteriori density estimated by the Log-PF.

4.1. Minimum Mean Square Error Estimate. The MMSE point estimate using the approximated a posteriori density, see, for example, [16], is defined by

$$\widehat{\mathbf{x}}_k^{\text{MMSE}} = \sum_{j=1}^{N_p} e^{\widehat{w}_k^{(j)}} \mathbf{x}_k^{(j)}, \quad (17)$$

where the l th element of the vector $\widehat{\mathbf{x}}_k^{\text{MMSE}}$ can be expressed as

$$[\widehat{\mathbf{x}}_k^{\text{MMSE}}]_l = \sum_{j=1}^{N_p} e^{\widehat{w}_k^{(j)}} [\mathbf{x}_k^{(j)}]_l. \quad (18)$$

In order to use the Jacobian logarithm to compute (18) in log-domain, we separate the positive and negative values of $[\mathbf{x}_k^{(j)}]_l$ with

$$\mathcal{A}_{+,l} = \{j | j \in \{1, \dots, N_p\} \wedge [\mathbf{x}_k^{(j)}]_l > 0\}, \quad (19)$$

$$\mathcal{A}_{-,l} = \{j | j \in \{1, \dots, N_p\} \wedge [\mathbf{x}_k^{(j)}]_l < 0\}$$

and the corresponding log-weight $\widehat{w}_k^{(j)}$ accordingly. Please note $[\mathbf{x}_k^{(j)}]_l = 0$ is not considered in (19) because $(e^{\widehat{w}_k^{(j)}} \cdot [\mathbf{x}_k^{(j)}]_l) = 0$. Thus, we obtain from (18) for the MMSE estimate,

$$\begin{aligned} [\widehat{\mathbf{x}}_k^{\text{MMSE}}]_l &= e^{\ln(\sum_{j \in \mathcal{A}_{+,l}} e^{\widehat{w}_k^{(j)} + \ln([\mathbf{x}_k^{(j)}]_l)})} \\ &\quad - e^{\ln(\sum_{j \in \mathcal{A}_{-,l}} e^{\widehat{w}_k^{(j)} + \ln([\mathbf{x}_k^{(j)}]_l)})} \\ &= e^{\ln(\sum_{j \in \mathcal{A}_{+,l}} e^{m^{j,l}})} - e^{\ln(\sum_{j \in \mathcal{A}_{-,l}} e^{m^{j,l}})} \\ &= e^{\text{Jacob}(\{m^{j,l}\}_{j \in \mathcal{A}_{+,l}})} - e^{\text{Jacob}(\{m^{j,l}\}_{j \in \mathcal{A}_{-,l}})}, \end{aligned} \quad (20)$$

where we introduced $m^{j,l}$ with

$$m^{j,l} = \widehat{w}_k^{(j)} + \ln([\mathbf{x}_k^{(j)}]_l). \quad (21)$$

4.2. Maximum A Posteriori Estimate. The MAP point estimate is defined as

$$\widehat{\mathbf{x}}_k^{\text{MAP}} = \arg \max_{\mathbf{x}_k} p(\mathbf{x}_k | \mathbf{z}_{1:k}), \quad (22)$$

which can be approximated, see [17], by

$$\widehat{\mathbf{x}}_k^{\text{MAP}} \approx \arg \max_{\mathbf{x}_k^{(j)}} p(\mathbf{z}_k | \mathbf{x}_k^{(j)}) \sum_{i=1}^{N_p} p(\mathbf{x}_k^{(j)} | \mathbf{x}_{k-1}^{(i)}) w_{k-1}^{(i)}, \quad (23)$$

for $j = 1, \dots, N_p$. The corresponding MAP state estimator using weights in log-domain can be calculated using the Jacobian logarithm of (12) with

$$\begin{aligned} \widehat{\mathbf{x}}_k^{\text{MAP}} &\approx \arg \max_{\mathbf{x}_k^{(j)}} p(\mathbf{z}_k | \mathbf{x}_k^{(j)}) \sum_{i=1}^{N_p} e^{\ln(p(\mathbf{x}_k^{(j)} | \mathbf{x}_{k-1}^{(i)})) + \widehat{w}_{k-1}^{(i)}} \\ &= \arg \max_{\mathbf{x}_k^{(j)}} p(\mathbf{z}_k | \mathbf{x}_k^{(j)}) \\ &\quad \times \text{Jacob} \left(\left\{ \ln(p(\mathbf{x}_k^{(j)} | \mathbf{x}_{k-1}^{(i)})) + \widehat{w}_{k-1}^{(i)} \right\}_{i=1}^{N_p} \right). \end{aligned} \quad (24)$$

5. Simulations

In this section, we demonstrate the performance of the Log-PF using floating point 64-bit number accuracy according to IEEE Standard 754 for double precision with three simulations.

5.1. Linear Processes. First, we simulate a linear Gaussian model. The KF introduced in [18] is an optimal recursive Bayesian filter which can be used if the considered system is linear and the probabilistic model is Gaussian. Hence, we compare the Log-PF and the Lin-PF to the KF as benchmark. The simulation considers the linear transition model

$$\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{v}_k = \begin{pmatrix} 1 & 0.5 \\ 0 & 1 \end{pmatrix} \mathbf{x}_k + \mathbf{v}_k, \quad (25)$$

with the transition matrix \mathbf{F} , the state vector $\mathbf{x}_k = (a_k, b_k)^T$, and the zero-mean multivariate Gaussian distributed process noise $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \sigma_v \mathbf{I})$ with standard deviation σ_v and the identity matrix \mathbf{I} . The measurement model is defined by

$$z_k = \mathbf{H}\mathbf{x}_k + n_k = (1 \ 1) \mathbf{x}_k + n_k, \quad (26)$$

with the measurement matrix \mathbf{H} and the zero-mean Gaussian distributed measurement noise $n_k \sim \mathcal{N}(0, \sigma_n)$ with standard deviation σ_n . Based on the measurements z_k , the state sequence \mathbf{x}_k for $k = 1, \dots, 60$ is estimated using a KF, the Lin-PF and the Log-PF with $N_p = 200$ particles and known initial state $\mathbf{x}_1 = (2, 2)^T$. For the Lin-PF, we use the standard PF implementation as well as the Lin-Log-PF.

First, we compare the KF to the SIR Lin-PF, SIR Lin-Log-PF, and the SIR Log-PF. In order to see the robustness of the SIR Log-PF, we variate the measurement noise standard deviation σ_n from 10^8 down to 10^{-150} . We simulate 1000 different realizations with known initial state for each run. Figure 1 shows the root mean square error (RMSE) averaged over all time steps and simulations versus the decreasing measurement noise standard deviation σ_n . The abbreviation SIR Log-PF MAP stands for the MAP point estimate and SIR Log-PF MMSE for the MMSE point estimate of the SIR Log-PF. Respectively, the abbreviations SIR Lin-PF MAP, SIR Lin-Log-PF MAP, SIR Lin-PF MMSE, and SIR Lin-Log-PF MMSE stand for the SIR Lin-PF and SIR Lin-Log-PF point estimates. We see that the KF obtains the best estimation results followed by the SIR Log-PF and SIR Lin-Log-PF. Figure 1 shows additionally an enlarged subfigure of the region for $10^8 < \sigma_n < 10^{-8}$. For $\sigma_n > 10^{-2}$, all SIR PFs obtain equivalent performance. As soon as σ_n decreases, the RMSE decreases for the SIR PFs until $\sigma_n = 10^{-2}$. For lower measurement noise standard deviations, the RMSE of the SIR Lin-PF and increases up to a limit of 0.7 whereas the accuracy of the SIR Log-PF and SIR Lin-Log-PF are limited by the number of particles. This effect is caused by the number representation of the particle weights of the SIR Lin-PF. For $\sigma_n < 10^0$, the particle weights of the SIR Lin-PF are small and at similar order as numerical errors due to number representation. Therefore, numerical errors dominate the resampling step such that the resampling algorithm draws

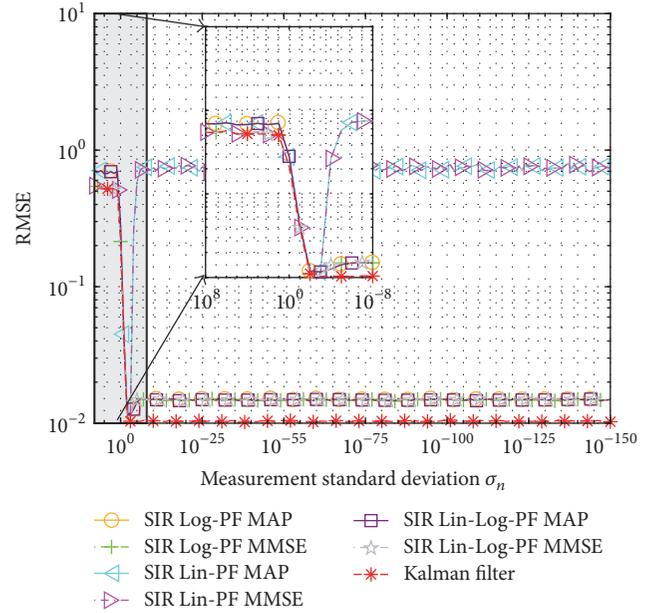


FIGURE 1: Performance evaluation of the SIR Log-PF, SIR Lin-PF, SIR Lin-Log-PF, and KF by the RMSE versus a decreasing measurement standard deviation σ_n .

particles based on numerical inaccuracies. Thus, the update step of the SIR Lin-PF loses its effect and we obtain 0.7 as the expected error of the process model in (25). For lower standard deviations, the accuracy of the SIR Log-PF and Lin-Log-PF is limited by the number of particles. The RMSE of the SIR Log-PF and Lin-Log-PF stay constant around 0.05 for the standard deviations between $10^{-1} > \sigma_n > 10^{-150}$. The KF is the optimal filter for this simulation; hence, the RMSE of the KF is limited by the process noise; hence, the RMSE is constant around 0.01.

To summarize, we obtain for all considered standard deviations equal or better estimation results using the SIR Log-PF compared to the SIR Lin-PF. However, the Lin-Log-PF which computes the weights in log-domain using (6) obtains similar simulation results compared to the SIR Log-PF. Hence, in the following we show the benefits of the Log-PF by using SIS PFs, where no resampling is performed. The resampling step is a key point for the success of a PF which is applied to avoid the degeneracy. However, resampling yields to a loss of diversity in the propagation of particles and entails an additional computational cost [14]. Similar to the SIR PF, we set the importance density to be equal to the transitional prior distribution, with $q(\mathbf{x}_k | \mathbf{x}_{k-1}^{(j)}, z_k) = p(\mathbf{x}_k^{(j)} | \mathbf{x}_{k-1}^{(j)})$.

Figure 2 shows the RMSE averaged over all time steps and simulations versus the decreasing measurement noise standard deviation σ_n . The curve looks similar to Figure 1. For $\sigma_n > 10^{-2}$, the SIS PFs obtain equivalent performance. As soon as σ_n decreases, the RMSE decreases for the SIS PFs until $\sigma_n = 10^{-2}$. For lower measurement noise standard deviations, the RMSE of the SIS Lin-PF and Lin-Log-PF increase up to the limit of 0.7. The RMSE of the SIS Log-PF stays constant

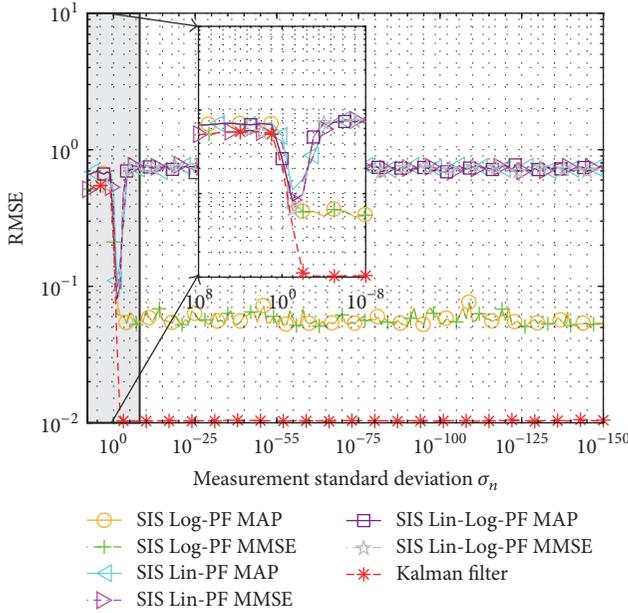


FIGURE 2: Performance evaluation of the SIS Log-PF, SIS Lin-PF, SIS Lin-Log-PF, and KF by the RMSE versus a decreasing measurement standard deviation σ_n .

around 0.03 for the standard deviations between $10^{-2} > \sigma_n > 10^{-150}$.

However, since the SIS PF uses no resampling step, the transformation of the weights from the log-domain to the lin-domain is not essential. Hence, if the weights of the SIS Lin-Log-PF are not transferred to the lin-domain and the weights are normalized with $\hat{w}_k^{(j)} = \hat{w}_k^{*(j)} - \max_l(\hat{w}_k^{*(l)})$ and propagated to the next time instant, the Lin-Log-PF obtains equivalent estimation results than the SIS Log-PF. Therefore, the transformation to the lin-domain in Line 6 in Algorithm 6 may introduce numerical inaccuracies. As long as no normalization is needed, the Lin-Log-PF can be computed completely in the log-domain.

5.2. Nonlinear Processes: Generic Particle Filter. In this section, we use the Generic PF which decides adaptively when the resampling step is performed. Similar to the SIR PF, we set the importance density to be equal to the transitional prior distribution, with $q(\mathbf{x}_k | \mathbf{x}_{k-1}^{(j)}, \mathbf{z}_k) = p(\mathbf{x}_k^{(j)} | \mathbf{x}_{k-1}^{(j)})$. We compare the Generic Log-PF to the Generic Lin-PF and Generic Lin-Log-PF. As mentioned before, a pseudocode of the Generic Lin-Log-PF is shown in Algorithm 6. In all algorithms, resampling is performed whenever the effective sample size N_{eff} falls below a threshold N_{thr} . We use the approximation $P_{N_p}^{(2)}(\mathbf{w}_k)$ for the effective sample size in the lin-domain and respectively $\ln(P_{N_p}^{(2)}(\mathbf{w}_k))$ in the log-domain. Similar to [14], we consider a stochastic volatility model

$$\begin{aligned} x_k &= \alpha x_{k-1} + v_k, \\ z_k &= e^{x_k/2} n_k \end{aligned} \quad (27)$$

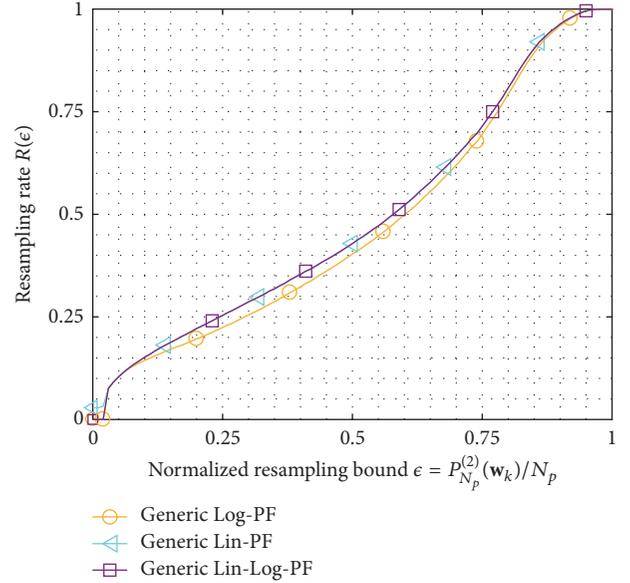


FIGURE 3: Performance evaluation of the Generic Log-PF, Generic Lin-PF, and Generic Lin-Log-PF by the resampling rate $R(\epsilon)$ versus the normalized resampling bound $\epsilon = P_{N_p}^{(2)}(\mathbf{w}_k)/N_p$.

with $\alpha = 0.99$ and the zero-mean Gaussian distributed process and measurement noise $v_k \sim \mathcal{N}(0, \sigma_v)$ and $n_k \sim \mathcal{N}(1, \sigma_n)$, where n_k is a multiplicative noise. Based on the measurements z_k , the state sequence x_k for $k = 1, \dots, 3000$ is estimated using the Generic PFs with $N_p = 100$ particles. We set the measurement noise standard deviation to $\sigma_n = 0.01$ and the process noise standard deviation to $\sigma_v = 0.01$. For performance evaluation, we try to recreate situations with rapidly changing measurements, that is, a certain model mismatch between the true likelihood of the process and the likelihood representation inside the PF. Inside the Generic PFs, we use the measurement noise standard deviation $\sigma_{n,p} = 10^{-4}$. Hence, taking the model mismatch into account it is more likely that particle states are located in the tail of the PF's likelihood after the prediction step. For the simulations, we vary the normalized resampling bound $\epsilon = P_{N_p}^{(2)}(\mathbf{w}_k)/N_p$ from 0 to 1 using a grid of 0.01 resolution (equivalently for the log-domain). We simulate 1000 different realizations with known initial state and count the number of performed resampling for each run. The resampling rate is afterwards calculated as $R(\epsilon) = \mathbf{E}[\text{Number Resampling using } \epsilon/3000]$, where $\mathbf{E}[x]$ stands for the sample mean of x .

Figure 3 shows the resampling rate $R(\epsilon)$ averaged over all time steps and simulations versus the normalized resampling bound ϵ . For $0.2 < \epsilon < 0.8$, the Generic Log-PF has a lower resampling rate than the Generic Lin-PF and the Generic Lin-Log-PF. Figure 4 shows the RMSE versus the resampling rate $R(\epsilon)$, where the RMSE decreases with increasing resampling rate for all Generic PFs. However, we can observe that, for a resampling rate of $0.15 < R(\epsilon) < 0.5$, we obtain a slightly lower RMSE with the Generic Log-PF than using the Generic Lin-PF or Generic Lin-Log-PF.

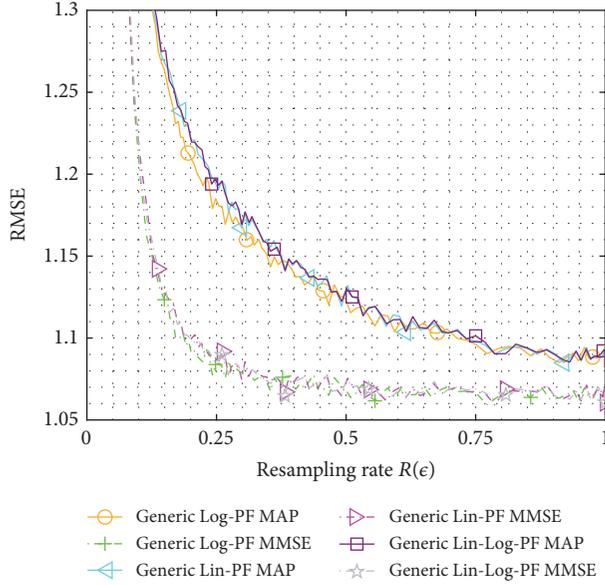


FIGURE 4: Performance evaluation of the Generic Log-PF, Generic Lin-PF, and Generic Lin-Log-PF by the RMSE versus the resampling rate $R(\epsilon)$.

5.3. *Rao-Blackwellization.* In this section we consider a simultaneous localization and mapping (SLAM) example with radio signals indicated in Figure 5 according to the system model in [19] (similar results are expected in, e.g., belief propagation [20] or distributed PFs [13]). A receiver which is moving along an arbitrary trajectory is measuring the distances

$$\mathbf{d}_k = [d_1(k), \dots, d_{N_k}(k)]^T, \quad (28)$$

with

$$d_i(k) = \|\mathbf{r}_{u,k} - \mathbf{r}_{T,i,k}\| + c_{i,k} + n_{i,k}, \quad (29)$$

between the receiver at location $\mathbf{r}_{u,k}$ and $i = 1, \dots, N_k$ transmitters at location $\mathbf{r}_{T,i,k}$ with the distance offset $c_{i,k}$ and zero-mean Gaussian distributed measurement noise $n_{i,k} \sim \mathcal{N}(0, \sigma_i)$ with standard deviation σ_i . The receiver uses the control input \mathbf{u}_k to move from state $\mathbf{x}_{u,k-1}$ to state $\mathbf{x}_{u,k}$. In order to use the distance measurements \mathbf{d}_k for positioning, the positioning algorithm estimates the receiver and transmitter states simultaneously. The state vector \mathbf{x}_k describing the complete system at time instant k is

$$\mathbf{x}_k = (\mathbf{x}_{u,k}^T, \mathbf{x}_{T,k}^T)^T, \quad (30)$$

with the receiver $\mathbf{x}_{u,k}$ and the transmitter states $\mathbf{x}_{T,k}$ which are also unknown. The receiver state $\mathbf{x}_{u,k} = (\mathbf{r}_{u,k}^T, \mathbf{v}_{u,k}^T)^T$ includes the receiver position $\mathbf{r}_{u,k}$ and the receiver velocity $\mathbf{v}_{u,k}$, while the transmitter states are defined by

$$\mathbf{x}_{T,k} = (\mathbf{x}_{T,1,k}^T, \dots, \mathbf{x}_{T,N_k,k}^T)^T \quad (31)$$

with

$$\mathbf{x}_{T,i,k} = (\mathbf{r}_{T,i,k}^T, c_{i,k})^T, \quad (32)$$

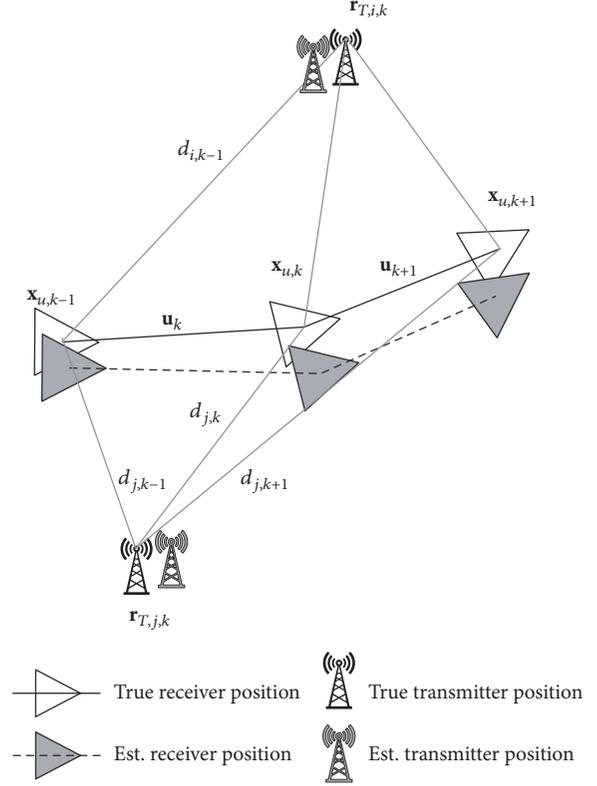


FIGURE 5: Overview of the SLAM example: the moving receiver simultaneously estimates its position and the location of the transmitters.

for $i = 1, \dots, N_k$ transmitters. We consider a static environment with a fixed number of transmitters and a receiver moving along an arbitrary trajectory. However, for notational convenience, a time dependence on k is introduced here for the transmitter positions $\mathbf{r}_{T,i,k}$. Additionally, we assume based on [19] that the distance offset $c_{i,k}$ is constant.

The state space is estimated by an algorithm according to [19, 21] based on Rao-Blackwellization [22], where the state space of \mathbf{x}_k is partitioned into subspaces. Hence, we use PFs to estimate the subspaces representing the transmitters inside a PF. The reason to use a PF instead of a low complexity extended Kalman filter (EKF) is the nonlinearity of the measurements in (29). The algorithm of [19, 21] is based on a superordinate particle filter (superPF) and subordinate particle filters (subPFs). Each particle $j = 1, \dots, N_p$ of the superPF with the state vector $\mathbf{x}_{u,k}^{(j)} = (\mathbf{r}_{u,k}^{(j)T}, \mathbf{v}_{u,k}^{(j)T})^T$ holds N_k subPFs. Each subPF is represented by the particles $\mathbf{x}_{T,i,k}^{(j,a)}$ with $a = 1, \dots, N_{p,j,i}$ where $N_{p,j,i}$ stands for the number of particles in the i th subPF with $i = 1, \dots, N_k$, estimating $\mathbf{x}_{T,i,k}^{(j)}$.

Similar to [19, 21], the posterior distribution $p(\mathbf{x}_{u,k}, \mathbf{x}_{T,k} | \mathbf{d}_{1:k}, \mathbf{u}_{1:k}, \mathbf{x}_{u,0})$ can be approximated by importance samples, as

$$p(\mathbf{x}_{u,k}, \mathbf{x}_{T,k} | \mathbf{d}_{1:k}, \mathbf{u}_{1:k}, \mathbf{x}_{u,0}) \approx \sum_{j=1}^{N_p} w_k^{(j)} \delta(\mathbf{x}_{u,k} - \mathbf{x}_{u,k}^{(j)}), \quad (33)$$

where $w_k^{(j)}$ defines the weight for the j th particle at time instant k with

$$w_k^{(j)} \propto p(\mathbf{d}_k | \mathbf{x}_{u,k}^{(j)}, \mathbf{d}_{k-1}) \propto \prod_{i=1}^{N_k} \sum_{a=1}^{N_{p,i,j}} w_{i,k}^{(j,a)} \quad (34)$$

and the weight $w_{i,k}^{(j,a)}$ of the a th particle of the i th subPF for the j th particle of the superPF at time instant k with

$$w_{i,k}^{(j,a)} \triangleq p(d_{i,k} | \mathbf{x}_{u,k}^{(j)}, \mathbf{x}_{T,i,k}^{(j,a)}). \quad (35)$$

Resampling is performed at each time instant to prevent degeneration; hence, (34) and (35) do not depend on the weights $w_{k-1}^{(j)}$ and $w_{i,k-1}^{(j,a)}$, respectively.

In the following we compare the position accuracy of three different implementations:

(i) *Lin-PF-SLAM*. SLAM algorithm which calculates the posterior distribution according to (33), (34), and (35) in the lin-domain as proposed in [19, 21].

(ii) *Log-PF-SLAM*. SLAM algorithm which calculates the posterior distribution in the log-domain. By transferring (33), (34), and (35) to the log-domain, we obtain

$$\begin{aligned} \widehat{w}_k^{(j)} &\propto \ln(p(\mathbf{d}_k | \mathbf{x}_{u,k}^{(j)}, \mathbf{d}_{k-1})) \\ &\propto \sum_{i=1}^{N_k} \ln \left(\sum_{a=1}^{N_{p,i,j}} e^{\widehat{w}_{i,k}^{(j,a)}} \right) = \sum_{i=1}^{N_k} \text{Jacob} \left(\{ \widehat{w}_{i,k}^{(j,a)} \}_{a=1}^{N_{p,i,j}} \right). \end{aligned} \quad (36)$$

(iii) *Lin-Log-PF-SLAM*. SLAM algorithm which calculates the weights in the log-domain and transfers the weights afterwards to the lin-domain using (6). Hence, using (6) for (34), we obtain

$$w_k^{+(j)} = e^{\widehat{w}_k^{(j)} - \max_l(\widehat{w}_k^{(l)})}, \quad (37)$$

where

$$\widehat{w}_k^{(j)} \propto \sum_{i=1}^{N_k} \max_m (\widehat{w}_{i,k}^{(j,m)}) + \ln \left(\sum_{a=1}^{N_{p,i,j}} e^{\widehat{w}_{i,k}^{(j,a)} - \max_m(\widehat{w}_{i,k}^{(j,m)})} \right). \quad (38)$$

We evaluate the performance of the different algorithms using a two dimensional scenario with three static transmitters and a moving receiver indicated in Figure 6. The transmitters are located at constant locations summarized in Table 2. The receiver is moving on a random pathway for 60 s with a system sampling interval of 1 s.

The transition model of the receiver states uses a standard discrete white noise acceleration model [23], with

$$\mathbf{x}_{u,k} = \mathbf{A}(\dot{\Psi}_k, \Delta_k) \mathbf{x}_{u,k-1} + \mathbf{b} \mathbf{n}_{t,k}^T, \quad (39)$$

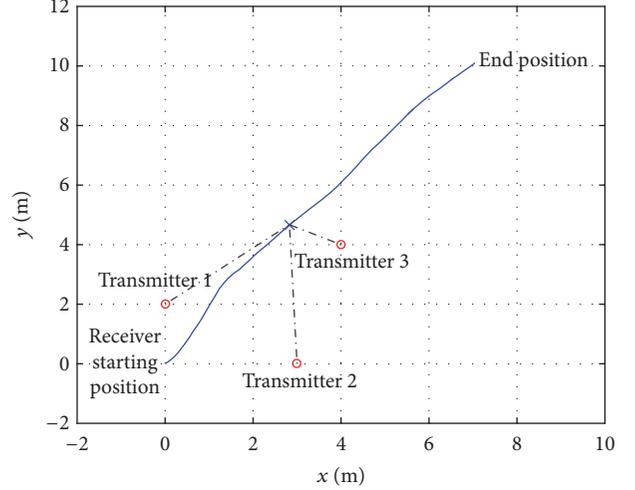


FIGURE 6: Simulated scenario with three transmitters indicated by the red circles and a receiver moving on the blue track. The figure indicates the propagation paths from the transmitters to the receiver for $k = 30$ s.

TABLE 2: Transmitter properties.

Transmitter i	x -pos [m]	y -pos [m]	$c_{i,k}$ [m]	σ_i [m]
Transmitter 1	0	2	1	0.001
Transmitter 2	3	0	1	0.01
Transmitter 3	4	0	0	0.01

with

$$\mathbf{A}(\dot{\Psi}_k, \Delta_k) = \begin{pmatrix} 1 & 0 & \Delta_k & 0 \\ 0 & 1 & 0 & \Delta_k \\ 0 & 0 & \cos(\dot{\Psi}_k) & -\sin(\dot{\Psi}_k) \\ 0 & 0 & \sin(\dot{\Psi}_k) & \cos(\dot{\Psi}_k) \end{pmatrix}, \quad (40)$$

$$\mathbf{b} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}, \quad (41)$$

in a two-dimensional Cartesian coordinate system with the zero-mean multivariate Gaussian distributed process noise $\mathbf{n}_{t,k} \sim \mathcal{N}(\mathbf{0}, \sigma_v \mathbf{I})$ with standard deviation σ_v . The receiver state vector $\mathbf{x}_{u,k} = (\mathbf{r}_{u,k}, \mathbf{v}_{u,k})^T$ consists of the x - y positions $\mathbf{r}_{u,k} = (r_{u,x,k}, r_{u,y,k})^T$ and the velocities $\mathbf{v}_{u,k} = (v_{u,x,k}, v_{u,y,k})^T$ where $v_{u,x,k}$, $v_{u,y,k}$ are the corresponding velocities in x - y direction. The transition matrix in (40) includes a rotation matrix with the heading changes $\dot{\Psi}_k$, which are used as control input \mathbf{u}_k . The transmitter states $\mathbf{x}_{T,i,k}$ are time-invariant; hence, we obtain for the transition prior $p(\mathbf{x}_{T,i,k} | \mathbf{x}_{T,i,k-1})$ of the i th transmitter $p(\mathbf{x}_{T,i,k} | \mathbf{x}_{T,i,k-1}) = \delta(\mathbf{x}_{T,i,k} - \mathbf{x}_{T,i,k-1})$.

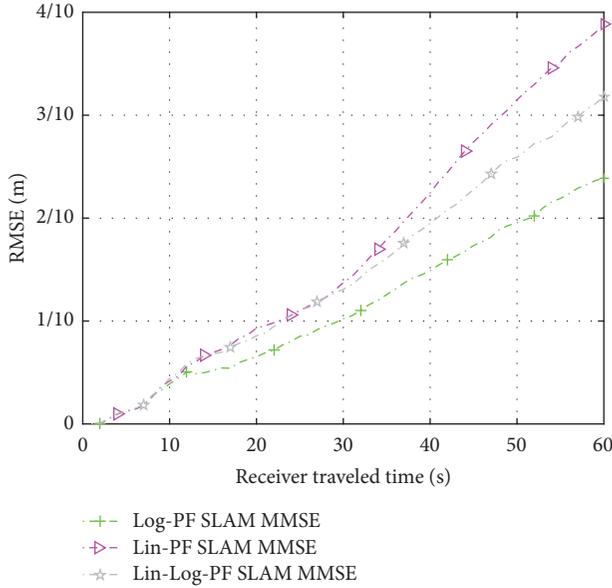


FIGURE 7: RMSEs of the estimated receiver positions versus receiver traveled time for different algorithms.

The simulations are performed using $N_p = 3000$ and $N_{P,j,i} = 1000$ particles for all transmitters $i = 1, \dots, N_k$. For the initialization, we use prior information $\mathbf{x}_{u,0}$ which is the knowledge on the starting position and velocity. Please note that an unknown starting position and direction or larger initial uncertainties may result in a biased and rotated coordinate system for the estimation. For simplicity, we use also prior information on the transmitter states $\mathbf{x}_{T,i,0}$ (please see [19, 21], e.g., on unknown transmitter states). The prior information includes a Gaussian distribution with standard deviation of 0.5 m centered around the true transmitter states, that is, position and distance offset. For computing the position estimate $\hat{\mathbf{r}}_{u,k}$, we use the MMSE estimate as introduced in [19, 21].

Figure 7 shows the RMSE versus the receiver traveled time for the different implementations with $\text{RMSE}_{u,k} = \sqrt{\mathbf{E}\{\|\mathbf{r}_{u,k} - \hat{\mathbf{r}}_{u,k}\|^2\}}$ and 200 independent evaluations. At the starting time, the RMSE for all algorithms are similar because of the identical initialization. Afterwards, the RMSE increases caused by the dilution of precision (DOP). However, we can clearly see that we obtain a higher accuracy using the Log-PF compared to the Lin-PF and the Lin-Log-PF.

6. Conclusion

In this paper we derived a particle filter representation in logarithmic domain, called Log-PF. The derivations show that the weight calculation, weight normalization, resampling, and point estimations can be expressed in logarithmic domain using the Jacobian logarithm. Representing the weight of each particle in logarithmic domain allows reducing the effect of numerical issues. Furthermore, the algorithm derived in this paper can be generalized to multidimensional nonparametric marginalization.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors would like to thank Patrick Robertson and Luigi Bruno for many interesting discussions on this problem. This work has been performed in the framework of the DLR project *Navigation 4.0* and the European Union Horizon 2020 research and innovation programme under Grant Agreement no. 636537 *HIGHTS (High precision positioning for Cooperative-ITS applications)*.

References

- [1] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-gaussian Bayesian state estimation," *IEE proceedings Radar Signal Processing*, vol. 140, no. 2, pp. 107–113, 1993.
- [2] M. S. 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, 2002.
- [3] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the kalman filter: particle filters for tracking applications*, Artech House, 2004.
- [4] R. van der, N. Merwe, A. Doucet, and E. Wan, "The Unscented Particle Filter," Tech. Rep., Cambridge University Engineering Department, August 2000.
- [5] Z. Chen, "Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond," Tech. Rep., McMaster University, 2003.
- [6] F. Gustafsson, "Particle filter theory and practice with positioning applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 7, pp. 53–81, 2010.
- [7] L. Martino, J. Read, V. Elvira, and F. Louzada, "Cooperative parallel particle filters for online model selection and applications to urban mobility," *Digital Signal Processing*, vol. 60, pp. 172–185, 2017.
- [8] C. C. Drovandi, J. M. McGree, and A. N. Pettitt, "A sequential Monte Carlo algorithm to incorporate model uncertainty in Bayesian sequential design," *Journal of Computational and Graphical Statistics*, vol. 23, no. 1, pp. 3–24, 2014.
- [9] J. Lien, U. J. Ferner, W. Srichavengsup, H. Wymeersch, and M. Z. Win, "A comparison of parametric and sample-based message representation in cooperative localization," *International Journal of Navigation and Observation*, Article ID 281592, 2012.
- [10] J. Erfanian, S. Pasupathy, and G. Gulak, "Reduced Complexity symbol detectors with parallel structures for isi channels," *IEEE Transactions on Communications*, vol. 42, no. 234, pp. 1661–1671, 1994.
- [11] W. Koch and A. Baier, "Optimum and sub-optimum detection of coded data disturbed by time-varying intersymbol interference," in *Proceedings of the IEEE Global Telecommunications Conf.*, vol. 3, pp. 1679–1684, December 1990.
- [12] P. Robertson, E. Villebrun, and P. Hoehner, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proceedings of the IEEE Int. Conf. on Communications*, vol. 2, pp. 1009–1013, June 1995.

- [13] S. Zhang, E. Staudinger, W. Wang, C. Gentner, A. Dammann, and E. Sandgren, "DiPLoc: Direct signal domain particle filtering for network localization," in *Proceedings of the 28th International Technical Meeting of the Satellite Division of the Institute of Navigation, ION GNSS 2015*, pp. 2267–2274, usa, September 2015.
- [14] L. Martino, V. Elvira, and F. Louzada, "Effective sample size for importance sampling based on discrepancy measures," *Signal Processing*, vol. 131, pp. 386–401, 2017.
- [15] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models," *Journal of Computational and Graphical Statistics*, vol. 5, no. 1, pp. 1–25, 1996.
- [16] A. Doucet, N. J. Gordon, and V. Krishnamurthy, "Particle filters for state estimation of jump Markov linear systems," *IEEE Transactions on Signal Processing*, vol. 49, no. 3, pp. 613–624, 2001.
- [17] S. Saha, Y. Boers, H. Driessen, P. K. Mandal, and A. Bagchi, "Particle based map state estimation: a comparison," in *Proceedings of the In Proc. of the 12th Int. Conf*, pp. 278–283, July 2009.
- [18] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.
- [19] C. Gentner, T. Jost, W. Wang, S. Zhang, A. Dammann, and U.-C. Fiebig, "Multipath assisted positioning with simultaneous localization and mapping," *IEEE Transactions on Wireless Communications*, vol. 15, no. 9, pp. 6104–6117, 2016.
- [20] A. T. Ihler, J. W. Fisher, R. L. Moses, and A. S. Willsky, "Non-parametric belief propagation for self-localization of sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 809–819, 2005.
- [21] C. Gentner, R. Pöhlmann, M. Ulmschneider, T. Jost, and S. Zhang, "Positioning using terrestrial multipath signals and inertial sensors," *Mobile Information Systems*, vol. 2017, pp. 1–18, 2017.
- [22] A. Doucet, N. de Freitas, P. Kevin, and J. Stuart, "Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks," in *Proc. Conf. Uncertainty in Artificial Intelligence*, pp. 176–183, UAI, 2000.
- [23] Y. Bar-Shalom, Li. Rong, and T. Kirubarajan, *Estimation with Applications to Tracking And Navigation: Theory Algorithms and Software*, Wiley Sons, 2004.



Hindawi

Submit your manuscripts at
<https://www.hindawi.com>

