On computational complexity reduction methods for Kalman filter extensions

Matti Raitoharju and Robert Piché

***Abstract*—The Kalman filter and its extensions are used in a vast number of aerospace and navigation applications for nonlinear state estimation of time series. In the literature, different approaches have been proposed to exploit the structure of the state and measurement models to reduce the computational demand of the algorithms. In this tutorial, we survey existing code optimization methods and present them using unified notation that allows them to be used with various Kalman filter extensions. We develop the optimization methods to cover a wider range of models, show how different structural optimizations can be combined, and present new applications for the existing optimizations. Furthermore, we present an example that shows that the exploitation of the structure of the problem can lead to improved estimation accuracy while reducing the computational load. This tutorial is intended for persons who are familiar with Kalman filtering and want to get insights for reducing the computational demand of different Kalman filter extensions.**

I. Introduction

Since its pioneering application to trajectory estimation in the Apollo program in the 1960's, the Kalman Filter (KF) and its nonlinear extensions have been used in a multitude of aerospace and navigation applications, including inertial navigation, radar systems, and global navigation satellite systems [18]. KFs are also used in many other application areas, for example state estimation of a lithium polymer battery [11] or brain imaging [21].

KF is the optimal Bayesian filter under certain conditions, which include linearity of models and Gaussianity and whiteness of noise [22]. Kalman Filter Extensions (KFEs) are based on approximate linear-Gaussian models that extend the use of KF to nonlinear models. In the literature, there are several different types of KFEs, with different demands on computational resources. The computational complexity of the KFE increases when the number of estimated state variables or dimensionality of the measurement vector increases; in some KFEs even exponentially [25]. The number of state variables varies depending on the application, for example, in some positioning applications only 2 position variables are estimated, but in other positioning applications the state may also contain variables for the locations of thousands of landmarks. In many applications the computational resources are limited, for example, in miniaturized satellites [3].

In this tutorial, we study various methods to reduce the computational load of state estimation with KFEs by exploiting the structure of the state transition and measurement models. This tutorial is intended for persons who are familiar with the basics of KFEs and want to know how to reduce the computational demand of KFEs. The presented algorithms are such that the result is exact when applied to a situation where the KF produces the exact result. This leaves algorithms that are not optimal in the linear-Gaussian case, such as Ensemble Kalman Filter (EnKF), out of the scope of this tutorial. However, some of the given optimizations can still be applied with such algorithms.

We present the algorithms in a general form so that they can be applied to as wide a range of problems as possible, but still in a form that they are easy to implement. Some of the algorithms in the original sources are given only for a certain KFE; the general notation used in this tutorial allows the optimizations to be implemented for different KFEs. In addition to surveying the algorithms in the literature, we give some generalizations of the algorithms and new ways of applying the optimizations with the KFEs. To our knowledge, there is no similar unified presentation of optimization methods in the literature.

A drawback of KFEs is that, because of the Gaussian approximation on which the algorithm is based, the estimate is inaccurate when the true posterior is far from normal. Gaussian Mixture Filters (GMFs) (a.k.a. Gaussian sum filters) use a sum of normal densities to estimate the probability distributions and can approximate any probability density function [39]. Because GMFs use several KFEs the computational load is larger than with algorithms that use only one Gaussian. The optimizations in this tutorial can be applied also in the GMFs for state propagation and update of individual components.

For implementations, we assume that a library for basic linear algebra is available. In many practical applications, the algorithms can be optimized further by taking the specific structure of the matrices into account. If the matrices are sparse, the sparsity can be exploited for optimization either by hand or by implementing the algorithms with sparsity-optimized subroutines. These can be found, for example, in Sparse Basic Linear Algebra Subprograms[1] or in Matlab[2]. The optimizations in matrix algebra libraries also make it impossible to provide accurate complexity estimates of the given algorithms. For example the naive matrix multiplication for square matrices with n columns has complexity $\mathcal{O}(n^3)$ while Strassen's algorithm has complexity $\mathcal{O}(n^{2.807})$ [17], [43] and algorithms with smaller complexity exist, although they are faster only with very large n.

The computation time of algorithms can also be reduced using parallel computing. The matrix operations can be effectively parallelized in the linear algebra library that handles the matrix operations. Thus, we do not consider parallelization in

[1] http://math.nist.gov/spblas/

[2] http://www.mathworks.com/products/matlab/

this tutorial.

The remainder of this tutorial is organized as follows. In the next section we present the common notations that are used throughout the tutorial. In Section III the background of KFEs is presented. Section IV contains linearity based optimizations of the KFEs. In Section V the solution of the set of linear equations containing the inverse of the innovation covariance is optimized. Section VI presents optimizations based on division of the state into multiple subprocesses. Section VII gives example applications of the optimization methods. Section VIII concludes the article.

II. Notations

In the following list there are the variables that are used throughout the tutorial. In different sections, there are algorithm specific notations that are explained as they occur.

- Scalars
  - j number of iterations
  - m dimension of measurement
  - n dimension of state (or augmented state)
- Random variables
  - x state
  - z augmented state
  - ε noise
  - ε^x state transition noise
  - ε^y measurement noise
- Subscripts
  - i iteration index
  - t time index
- Functions
  - $f(\cdot)$ state transition function
  - $g(\cdot)$ non-specific function
  - $G(\cdot)$ Matrix valued function
  - $h(\cdot)$ measurement function
  - $\mathrm{diag}(\cdot)$ diagonal matrix with function arguments on its diagonal
- Expected values
  - $\mathrm{E}[x]$ expected value of x
  - $P_{g(x)x}$ covariance of $g(x)$ and x
  - μ_x mean of x
- Other variables
  - I identity matrix
  - J matrix that defines the statistically linearized relationship between state and measurement, which is the Jacobian matrix in linear systems
  - K Kalman gain
  - S innovation covariance
  - $\mathbf{0}$ zero matrix
  - $\mathbf{1}$ matrix containing ones
- Acronyms
  - **CKF** Cubature Kalman Filter
  - **DD** Divided Difference
  - **EKF** Extended Kalman Filter
  - **EKF2** Second Order Extended Kalman Filter
  - **EnKF** Ensemble Kalman Filter
  - **GF** Gaussian Filter
  - **GMF** Gaussian Mixture Filter
  - **KF** Kalman Filter
  - **KFE** Kalman Filter Extension
  - **PDR** Pedestrian Dead Reckoning
  - **QKF** Gauss-Hermite Quadrature Kalman filter
  - **RBPF** Rao-Blackwellized Particle Filter
  - **RUF** Recursive Update Filter
  - **SLAM** Simultaneous Localization and Mapping
  - **SOKF2** Second Order Polynomial Kalman Filter
  - **S^{2}KF** Smart Sampling Kalman Filter
  - **TDoA** Time Difference of Arrival
  - **UKF** Unscented Kalman Filter

III. Background

In discrete-time Bayesian filtering, the state of a dynamical system is estimated based on noisy measurements. The state model describes how the n-dimensional state x is propagated in time. A state model can be expressed as

$$x_t = f_t(x_{t-1}, \varepsilon_t^x), \tag{1}$$

where $f(\cdot)$ is the state transition function, x_{t-1} is the state at the previous time step, and ε_t^x is the state transition noise. The state's distribution is updated using measurements of the form

$$y_t = h_t(x_t, \varepsilon_t^y), \tag{2}$$

where y_t is the realized measurement, $h(\cdot)$ is the measurement function, and ε^y is the measurement noise.

To shorten notations we use the augmented state $z = \begin{bmatrix} x \\ \varepsilon \end{bmatrix}$, where applicable. In the augmented state ε is either state transition noise or measurement noise, depending on context. The state and measurement noises are assumed independent and white. We also assume that all the variables of interest are in the state. If the state or measurement noises are not white, they can be modeled using more variables in the state. In the Schmidt-Kalman filter, the additional noise bias states are not estimated, instead their effect on the covariance is approximated. However, the Schmidt-Kalman filter is suboptimal and we do not treat it in this paper. The interested reader may refer to [26, p.282]. We omit the time index t when its absence should not cause confusion.

In general, a KFE can be divided into two parts:

1) Prediction:

$$\mu_{x_t}^- = \mu_{f(z_{t-1})} \tag{3}$$

$$P_{x_t,x_t}^- = P_{f(z_{t-1})f(z_{t-1})}, \tag{4}$$

where $\mu_{x_t}^-$ is the predicted mean computed using the state transition function and posterior of the previous time step and P_{x_t,x_t}^- is the predicted covariance.

2) Update:

$$y_t^- = \mu_{h(z_t^-)} \tag{5}$$

$$S_t = P_{h(z_t^-)h(z_t^-)} \tag{6}$$

$$K_t = P_{x_t^- h(z_t^-)} S_t^{-1} \tag{7}$$

$$\mu_{x_t} = \mu_{x_t}^- + K(y_t - y_t^-) \tag{8}$$

$$P_{x_t,x_t} = P^-_{x_t,x_t} - K_t S_t K_t^T, \tag{9}$$

where z_t^- is the augmented state of predicted state and measurement noise, y_t^- is the predicted mean of the measurement, S_t is the innovation covariance, K_t is the Kalman gain, μ_{x_t} is the updated mean of the state, and P_{x_t,x_t} is the updated state covariance. In (7), $P_{x_t^- h(z_t^-)}$ refers to the rows of $P_{z_t^- h(z_t^-)}$ that correspond to the state variables.

The mean and covariance matrices associated to measurement and state transition functions can be written using expectations:

$$\mu_{g(z)} = \mathrm{E}\left[g(z)\right] \tag{10}$$

$$P_{g(z)g(z)} = \mathrm{E}\left[(g(z) - \mu_{g(z)})(g(z) - \mu_{g(z)})^T\right] \tag{11}$$

$$P_{zg(z)} = \mathrm{E}\left[(z - \mu_z)\left(g(z) - \mu_{g(z)}\right)^T\right]. \tag{12}$$

When a function is linear, that is, of the form

$$g(z) = Jz, \tag{13}$$

the expectations have analytic form:

$$\mu_{g(z)} = J\mu_z \tag{14}$$

$$P_{g(z)g(z)} = JP_{zz}J^T \tag{15}$$

$$P_{zg(z)} = P_{zz}J^T \tag{16}$$

and the algorithm is the KF.

There are extensions that approximate the expectations (10)–(12) using analytic differentiation (or integration) of the functions. For example, the Extended Kalman Filter (EKF) uses the first order Taylor expansion and the Second Order Extended Kalman Filter (EKF2) does the linearization based on the second order Taylor expansion [26]. There are also algorithms that use numerical differentiation to approximate (10)–(12). In Divided Difference (DD) filters [31] the computation is based on numerical differentiation of the functions to obtain a linear model. DD filters use $2n+1$ function evaluations. The Second Order Polynomial Kalman Filter (SOKF2) [25] uses $\frac{1}{2}n^2 + \frac{3}{2}n + 1$ points to fit a second order polynomial to the function and then computes the analytic covariance matrices for the polynomial.

One very commonly used and large group of KFEs algorithms approximate expectations (10)–(12) as:

$$\mu_{g(z)} \approx \sum w_i^s g(\chi_i) \tag{17}$$

$$P_{g(z)g(z)} \approx \sum w_i^c (g(\chi_i) - \mu_{g(z)})(g(\chi_i) - \mu_{g(z)})^T \tag{18}$$

$$P_{zg(z)} \approx \sum w_i^c (\chi_i - z)(g(\chi_i) - \mu_g(z))^T, \tag{19}$$

where χ_i are so-called sigma-points that are chosen according to the prior distribution and w_i^s and w_i^c are associated weights. Examples of this kind of filters are Unscented Kalman Filter (UKF) [46], different Cubature Kalman Filters (CKFs) [2], and Gauss-Hermite Quadrature Kalman filter (QKF) [25].

The selection and number of sigma-points and weights depend on the algorithm used. The UKF is usually used with $2n + 1$ sigma-points. The Gaussian Filter (GF) uses $kn + 1$ sigma-points, where $k > 1$ is an integer parameter [24]. CKFs are developed for different orders and they use $\mathcal{O}(n^o)$ sigma-points, where o is the order of the cubature rule. The number of sigma-points in QKFs increases exponentially, as the number of sigma-points is α^n, where $\alpha > 1$ [25]. There are also algorithms that allow an arbitrary number of points, for example Smart Sampling Kalman Filter (S^2KF) [41], which uses at least $n + 1$ sigma-points.

Algorithms that use (17)–(19) do not compute J explicitly. In some optimizations J is required and can be computed using statistical linearization [37]

$$J = P_{g(z)z}P_{zz}^{-1}. \tag{20}$$

For nonlinear systems, this cannot be substituted into (15) to obtain $P_{g(z)g(z)}$. Because the computation of (20) requires solving a set of linear equations it should be avoided when the dimension of z is large.

The computational requirements of the algorithms can be reduced in several ways. One way is to compute the expected values numerically only for state variables that are transformed by a nonlinear function and compute the expectations for linear parts analytically [10]. The improving of implementation efficiency using the linear KF update for the linear state variables is more familiar in particle filtering. Specifically Rao-Blackwellized Particle Filter (RBPF) solves the estimates of conditionally linear variables using a KF and rest of the variables using particles [14]. The dimension of the output of the nonlinear part can also be reduced to reduce the computational burden. In (18) the $m \times m$ covariance matrix is updated for every sigma point. Thus, halving the dimension m of $g(\cdot)$ reduces the operations applied to the covariance matrix by a factor of four. Such update optimizations are considered in Section IV.

It is also possible to compute the Kalman gain (7) faster by exploiting the structure of the innovation covariance (6) as shown in Section V.

When the state can be divided into decoupled blocks, the blocks can be updated independently and the global estimate can be computed only when needed. The updates of the blocks can be made at different rates. These optimizations are presented in Section VI.

IV. Optimizations Based on the Linearity in Nonlinear functions

A. Partially Linear Functions

In [7], algorithms for treating different setups of UKF were presented. The algorithms considered different implementations when the state transition model or measurement model has one of the following three forms

$$g(x, \varepsilon) = Gx + \varepsilon \tag{21}$$

$$g(x, \varepsilon) = g_1(x) + \varepsilon \tag{22}$$

$$g(x, \varepsilon) = g_2(x, \varepsilon), \tag{23}$$

where noise ε is assumed independent of the state. The algorithms improve in computation efficiency when either or both of the state and measurement models belong to the first or second group. If both are in the first group the updates can

be computed exactly using the KF. If the function belongs to the second group then (10)–(12) become

$$\mu_{g(z)} = \mathrm{E}\left[g(x)\right] + \mu_{\varepsilon} \tag{24}$$

$$P_{g(z)g(z)} = \mathrm{E}\left[(g(x) - \mu_{g(x)})(g(x) - \mu_{g(x)})^T\right] + P_{\varepsilon\varepsilon} \tag{25}$$

$$P_{xg(z)} = \mathrm{E}\left[(x - \mu_x)\left(g(x) - \mu_{g(x)}\right)^T\right] \tag{26}$$

and the approximation can be computed for the dimension of the state instead of the dimension of the augmented state. This optimization is widely used among different filters and is often considered to be the standard model.

In [30] the function is split into a nonlinear $g_1(\cdot)$ part that depends only on a part of the state z_n and linear parts

$$h(z) = \begin{bmatrix} g_1(z_n) \\ H_1 z \end{bmatrix}. \tag{27}$$

The UKF is used for computing the expected values of the nonlinear part and the correlation between nonlinear and linear parts.

We generalize the above models to functions of form:

$$g(z) = A g_n\left(Tz\right) + Hz, \tag{28}$$

where T has full row rank. To reduce the required resources in computation T is chosen to have the minimal number of rows and $g(\cdot)$ to have the minimal number of elements. Expectations (10)–(12) are computed for Tz, which should have a smaller dimension than z.

For computing the update, an algorithm that approximates expectations (10)–(12) is required. These are usually computed in the update stage of a KFE. The cross correlation matrix $P_{zg(z)}$ (19) is not needed in the normal state propagation, but the algorithm from the update stage can be used for this.

The transformed augmented state is denoted

$$\tilde{z} = Tz \sim \mathrm{N}(\mu_{\tilde{z}} = T\mu_z, P_{\tilde{z}\tilde{z}} = TP_{zz}T^T). \tag{29}$$

When expectations $\mu_{g_n(\tilde{z})}$ (10), $P_{g_n(\tilde{z})g_n(\tilde{z})}$ (11), and $P_{\tilde{z}g_n(\tilde{z})}$ (12) are known, the expectations for (28) are

$$\mu_{g(z)} = A\mu_{g_n(\tilde{z})} + H\mu_z \tag{30}$$

$$P_{g(z)g(z)} = \begin{bmatrix} A & H \end{bmatrix} \begin{bmatrix} P_{g_n(\tilde{z})g_n(\tilde{z})} & P_{zg_n(\tilde{z})}^T \\ P_{zg_n(\tilde{z})} & P_{zz} \end{bmatrix} \begin{bmatrix} A^T \\ H^T \end{bmatrix} \tag{31}$$

$$P_{zg(z)} = \begin{bmatrix} P_{zg_n(\tilde{z})} & P_{zz} \end{bmatrix} \begin{bmatrix} A^T \\ H^T \end{bmatrix}, \tag{32}$$

where

$$P_{zg_n(\tilde{z})} = P_{zz}T^T\left(TP_{zz}T^T\right)^{-1} P_{\tilde{z}g_n(\tilde{z})}. \tag{33}$$

This is based on the fact that the cross term $P_{\tilde{z}g_n(\tilde{z})}$ describes the linear dependence of function $g_n(\tilde{z})$ and $\tilde{z}$ as in (20):

$$P_{\tilde{z}g_n(\tilde{z})} = P_{\tilde{z}\tilde{z}}J^T = TP_{zz}T^TJ^T. \tag{34}$$

and the term $P_{zg_n(\tilde{z})}$ is

$$P_{zg_n(\tilde{z})} = P_{zz}T^TJ^T. \tag{35}$$

Solving J from (34) and substituting it into (35) we get (33).

In the update, the matrix with cross terms is required for state variables only. This can be extracted by

$$P_{xh(z)} = \begin{bmatrix} I & 0 \end{bmatrix} P_{zh(z)} \tag{36}$$

Algorithm 1: State transition using a partially linear measurement

Input: $x_{t-1} \sim \mathrm{N}(\mu_{x_{t-1}}, P_{x_{t-1}x_{t-1}})$ `// State estimate from previous time step`
$\varepsilon_q \sim \mathrm{N}(\mu_{\varepsilon_q}, P_{\varepsilon_q\varepsilon_q})$, `// State transition noise`
$P_{x_{t-1}\varepsilon_q}$ `// cross covariance between state and state transition noise`
$f(x, \varepsilon_q) = f(z) = Ag\left(Tz\right) + Hz$, `// State transition function of given form`
Output: $x \sim \mathrm{N}(\mu_{x^-}, P_{x^-x^-})$ `// Propagated state`
$\mu_z = \begin{bmatrix} \mu_{x_{t-1}} \\ \mu_{\varepsilon_q} \end{bmatrix}$ `// Augmented mean`
$P_{zz} = \begin{bmatrix} P_{x_{t-1}x_{t-1}} & P_{x_{t-1}\varepsilon_q} \\ P_{x_{t-1}\varepsilon_q}^T & P_{\varepsilon_q\varepsilon_q} \end{bmatrix}$ `// Augmented covariance`
$\mu_{\tilde{z}} = T\mu_z$ `// Transformed mean`
$P_{\tilde{z}\tilde{z}} = TP_{zz}T^T$ `// Transformed covariance`
Compute $\mu_{g(\tilde{z})}$,$P_{g(\tilde{z})g(\tilde{z})}$, and $P_{\tilde{z}g(\tilde{z})}$ using a KFE
$P_{zg(\tilde{z})} = P_{zz}T^T\left(TP_{zz}T^T\right)^{-1} P_{\tilde{z}g(\tilde{z})}$
$\mu_{x^-} = A\mu_{g(Tz)} + H\mu_z$ `// Predicted mean of state`
$P_{x^-x^-} = \begin{bmatrix} A & H \end{bmatrix} \begin{bmatrix} P_{g(\tilde{z})g(\tilde{z})} & P_{zg(\tilde{z})}^T \\ P_{zg(\tilde{z})} & P_{zz} \end{bmatrix} \begin{bmatrix} A^T \\ H^T \end{bmatrix}$ `// Predicted covariance`

when the state variables appear first in the augmented state. Naturally, in a computationally efficient code (36) is done with indices, not matrix multiplication.

The algorithm for state transition is given in Algorithm 1 and an algorithm for state update is given in Algorithm 2. Use of these algorithms is beneficial when the dimension of $\tilde{z}$ is smaller than z and the matrix inverse in (33) is applied in a small dimension. Algorithms 1 and 2 are given in a general form and for applications they can be optimized further by considering the structures of matrices. Examples of exploiting the partially linear state are given in sections VII-A,VII-B, and VII-D.

B. Conditionally Linear Measurements

In [29], a situation where a function can be divided into nonlinear z_n and conditionally linear z_l parts

$$g(z) = g_n(z_n) + G_n(z_n)z_l \tag{37}$$

is considered. In the original article, the distribution of this nonlinear function is approximated using a modification of the UKF. The number of sigma-points depends on the dimension of z_n instead of the dimension of the full state.

This algorithm is converted for use with the GF in [5]. Here we present the algorithm in a general form that allows it to be used with any KFE that uses weighted points to compute the expectations, as in (24)–(26), although it cannot be used with other types of filters e.g. DD or SOKF2.

Algorithm 2: Update using a partially linear measurement

Input: $x^- \sim \mathrm{N}(\mu_{x^-}, P_{x^-x^-})$ // Prior state
$\varepsilon^y \sim \mathrm{N}(\mu_{\varepsilon^y}, P_{\varepsilon^y\varepsilon^y})$, // Measurement noise
$P_{x^-\varepsilon^y}$ // cross covariance between state and measurement noise
$h(x, \varepsilon^y) = h(z) = Ag(Tz) + Hz$,
// Measurement function of given form
Output: $x \sim \mathrm{N}(\mu_x, P_{xx})$ // Posterior estimate
$\mu_z = \begin{bmatrix} \mu_{x^-} \\ \mu_{\varepsilon^y} \end{bmatrix}$ // Mean of the augmented state
$P_{zz} = \begin{bmatrix} P_{x^-x^-} & P_{x^-\varepsilon^y} \\ P_{x^-\varepsilon^y}^T & P_{\varepsilon^y\varepsilon^y} \end{bmatrix}$ // Covariance matrix of the augmented state
$\mu_{\tilde{z}} = T\mu_z$ // Transformed mean
$P_{\tilde{z}\tilde{z}} = TP_{zz}T^T$ // Transformed covariance
Compute $\mu_{g(\tilde{z})}$,$P_{g(\tilde{z})g(\tilde{z})}$, and $P_{\tilde{z}g(\tilde{z})}$ using a KFE
$P_{zg(\tilde{z})} = P_{zz}T^T \left(TP_{zz}T^T\right)^{-1} P_{\tilde{z}g(\tilde{z})}$
$\mu_{h(z)} = A\mu_{g(Tz)} + H\mu_z$ // Predicted mean of measurement
$P_{h(z)h(z)} = \begin{bmatrix} A & H \end{bmatrix} \begin{bmatrix} P_{g(\tilde{z})g(\tilde{z})} & P_{zg(\tilde{z})}^T \\ P_{zg(\tilde{z})} & P_{zz} \end{bmatrix} \begin{bmatrix} A^T \\ H^T \end{bmatrix}$
// Innovation covariance
$P_{xh(z)} = \begin{bmatrix} P_{zg(\tilde{z})} & P_{zz} \end{bmatrix}_{[1:n,:]} \begin{bmatrix} A^T \\ H^T \end{bmatrix}$
// State-measurement cross correlation
$K = P_{xh(z)}P_{h(z)h(z)}^{-1}$ // Kalman Gain
$\mu_x = \mu_x^- + K(y - \mu_{h(z)})$ // Posterior mean
$P_{xx} = P_{x^-x^-} - KP_{h(z)h(z)}K^T$ // Posterior covariance

In the algorithm the sigma-points are computed for the nonlinear part only and then used for computing the conditional probabilities of the conditionally linear part:

$$\mu_{z_l|\chi_i} = \mu_{z_l} + P_{z_l z_n} P_{z_n z_n}^{-1} (\chi_i - \mu_{z_n}) \tag{38}$$
$$P_{z_l z_l|\chi_i} = P_{z_l z_l} - P_{z_l z_n} P_{z_n z_n}^{-1} P_{z_n z_l}. \tag{39}$$

The matrices in (38–39) are independent of the sigma-point χ_i. Thus, $P_{z_l z_n} P_{z_n z_n}^{-1}$ and $P_{z_l z_l|\chi_i}$ need to be computed only once. According to [29], the expectations for a function of form (37) can be approximated as

$$\mathcal{Y}_i = g_n(\chi_i) + G_n(\chi_i)\mu_{z_l|\chi_i} \tag{40}$$
$$\mu_{g(z)} = \sum w_i \mathcal{Y}_i \tag{41}$$
$$P_{g(z)g(z)} = \sum w_i \left((\mathcal{Y}_i - \mu_{g(z)}) (\mathcal{Y}_i - \mu_{g(z)})^T + G_n(\chi_i) P_{z_l z_l|\chi_i} G(\chi_i)^T \right) \tag{42}$$
$$P_{zg(z)} = \sum w_i \left(\left(\begin{bmatrix} \chi_i \\ \mu_{z_l|\chi_i} \end{bmatrix} - \mu_z \right) (\mathcal{Y}_i - \mu_{g(z)})^T + \begin{bmatrix} 0 \\ P_{z_l z_l|z_n} G_n(\chi_i)^T \end{bmatrix} \right). \tag{43}$$

These formulas can be used to compute the expectations for the nonlinear part $g(\tilde{z})$ in algorithms 1 and 2, if there are conditionally linear state variables. This is done in the example in sections VII-B and VII-C.

V. Optimizations Related to the Inverse of the Innovation Covariance

A. Block Diagonal Measurement Covariance

The formula for the Kalman gain (7) contains the inverse of the innovation covariance S. When the dimension of the innovation covariance (i.e. the number of measurements) is large the computation of the inverse or solving the set of linear equations is a computationally expensive operation.

When measurements are linear and the measurement covariance is diagonal, the Kalman update can be applied one measurement element at a time and the partially updated state used as a prior for the next measurement [42]. This kind of update can be generalized also to block diagonal measurement covariances and the update can be done by applying one block at a time. This reduces the computation time when the state dimension is small, the measurement dimension is large, and the measurements are independent.

When some measurements are independent and these different measurements contain different state terms, updating the state using only a part of measurements at once can be effectively combined with the algorithms presented in Section IV. The block update does not change the estimate when the measurement model is linear, but when the model is nonlinear the linearization is redone in the partially updated state. This may alter the output of the algorithm. In [34], [35] application of measurements one element at a time is used to improve the estimation accuracy by applying the most linear measurements first. The algorithm for updating by blocks is given in Algorithm 3.

B. Applying the Matrix Inversion Lemma to Innovation Covariance

When the measurements are not independent the block update formula cannot be used. In some situations the innovation

Algorithm 3: Block Update KF

Input: $x_0 \sim \mathrm{N}(\mu_{x,0}, P_{xx,0})$ // Prior state
$\varepsilon_i^y \sim \mathrm{N}(\mu_{\varepsilon_i^y}, P_{\varepsilon_i^y\varepsilon_i^y})$, $1 \le i \le n$ // Measurement noises
$h_i(x, \varepsilon_i^y)$, $1 \le i \le n$ // Measurement functions
Output: $x_n \sim \mathrm{N}(\mu_{x,n}, P_{xx,n})$ // Posterior estimate
for i=1 to n **do**
  $y_i^- = \mu_{h_i(x_{i-1}, \varepsilon_i^y)}$
  $S_i = P_{h_i(z_{t,i}^-)h(x_{i-1}, \varepsilon_i^y)}$
  $K_i = P_{xh_i(z_{t,i}^-)} S_i^{-1}$
  $\mu_{x,i} = \mu_{x,i-1} + K(y_i - y_i^-)$
  $P_{xx,i} = P_{xx,i-1} - K_i S_i K_i^T$
end

covariance can be written in the form

$$S = P_s + UP_vU^T, \tag{44}$$

where P_s is easy to invert, P_v has a small dimension and U is a transformation matrix.

Using the matrix inversion lemma [42, p. 62] the inverse of (44) is

$$S^{-1} = P_s^{-1} - P_s^{-1}U\left(P_v^{-1} + UP_s^{-1}U^T\right)^{-1}U^TP_s^{-1}. \tag{45}$$

This formula is worth using if the inverse of P_s is easy to compute or can be computed offline and the dimension of P_v is smaller than the dimension of P_s. An example of its use is given in Section VII-D.

VI. Optimization Based on Dividing the State into Individual Estimation Processes

In this section, we study optimizations that can be applied when the state can be divided into separate subprocesses. The situation where only part of state variables occur in the measurements and the part of the state that is not in measurements does not change in time has been considered in [12], [20]. In that situation the estimation process can be done at each time step only for the part of the state that has been observed and then the whole state is updated only occasionally.

Such algorithms are developed further in [19]. The algorithm given in [19] assumes that the state can be divided into substates that can be updated individually. The division can be done when the measurement and state transition models in a time interval from t_0 to t_1 are decoupled between the substates. State blocks $x_{\underline{1}}, x_{\underline{2}}, \ldots$ are considered to be decoupled if the state transition and measurement models can be expressed as

$$f_t(x, \varepsilon) = \begin{bmatrix} f_{\underline{1},t}(x_{\underline{1},t-1}, \varepsilon^x_{\underline{1},t}) \\ f_{\underline{2},t}(x_{\underline{2},t-1}, \varepsilon^x_{\underline{2},t}) \\ \vdots \end{bmatrix} \tag{46}$$

$$h_t(x, \varepsilon) = \begin{bmatrix} h_{\underline{1},t}(x_{\underline{1},t}, \varepsilon^y_{\underline{1},t}) \\ h_{\underline{2},t}(x_{\underline{2},t}, \varepsilon^y_{\underline{2},t}) \\ \vdots \end{bmatrix}, \tag{47}$$

where noise terms ε^x and ε^y are independent. The purpose of this optimization is to allow to track state variables that are decoupled from time index t_0 to time index t_1. Note that there is no requirement for blocks $x_{\underline{1}}, x_{\underline{2}}, \ldots$ to be independent at t_0. If the blocks were independent at t_0 and models could be expressed using (46) and (47) in all time instances, then the system could be solved using a set of independent KFEs.

A similar idea for splitting the state into multiple blocks is presented in [8], [45] under the name of multiple quadrature Kalman filtering. In multiple quadrature Kalman filtering, there is no strict requirement for blocks to be decoupled, which makes the algorithm applicable to a larger set of problems, but leads to additional approximations, so we follow [19].

To update block $x_{\underline{a}}$ from time t_0 to t_1 a new Gaussian variable $\bar{x}$, which is partitioned into two parts x and $\hat{x}$, is introduced:

$$\bar{x}_{\underline{a},t_0} = \begin{bmatrix} x_{\underline{a},t_0} \\ \hat{x}_{\underline{a},t_0} \end{bmatrix} \sim \mathrm{N}\left(\begin{bmatrix} \mu_{x_{\underline{a},t_0}} \\ \mu_{\hat{x}_{\underline{a},t_0}} \end{bmatrix}, \begin{bmatrix} P_{x_{\underline{a},t_0}x_{\underline{a},t_0}} & P_{x_{\underline{a},t_0}\hat{x}_{\underline{a},t_0}} \\ P_{\hat{x}_{\underline{a},t_0}x_{\underline{a},t_0}} & P_{\hat{x}_{\underline{a},t_0}\hat{x}_{\underline{a},t_0}} \end{bmatrix}\right)$$

$$= \mathrm{N}\left(\begin{bmatrix} \mu_{x_{\underline{a},t_0}} \\ \mu_{x_{\underline{a},t_0}} \end{bmatrix}, \begin{bmatrix} P_{x_{\underline{a},t_0}x_{\underline{a},t_0}} & P_{x_{\underline{a},t_0}x_{\underline{a},t_0}} \\ P_{x_{\underline{a},t_0}x_{\underline{a},t_0}} & P_{x_{\underline{a},t_0}x_{\underline{a},t_0}} \end{bmatrix}\right). \tag{48}$$

In the estimation of this, initially degenerate, variable from time t_0 to t_1 $x_{\underline{a}}$ is updated and propagated normally and $\hat{x}_{\underline{a}}$ changes only through its dependence on part $x_{\underline{a}}$.

In the prediction, state elements belonging to $x_{\underline{a}}$ are transformed and $\hat{x}_{\underline{a}}$ remains static:

$$\begin{bmatrix} x_{\underline{a},t} \\ \hat{x}_{\underline{a},t} \end{bmatrix} = \begin{bmatrix} f_{\underline{a},t}(x_{\underline{a},t-1}, \varepsilon^x_{\underline{a},t}) \\ \hat{x}_{\underline{a},t-1} \end{bmatrix}. \tag{49}$$

The measurement function is applied only to the first elements

$$h_{\underline{a},t}\left(\begin{bmatrix} x_{\underline{a},t} \\ \hat{x}_{\underline{a},t} \end{bmatrix}, \varepsilon^y_{\underline{a},t}\right) = h\left(x_{\underline{a},t}, \varepsilon^y_{\underline{a},t}\right). \tag{50}$$

These updates can be done with any suitable KF or other estimation method. The division of the state into blocks allows also updating blocks at different rates.

The number of elements in the covariance matrix of each block is $(2n_{\text{block}})^2$, thus if a 100-dimensional state is divided into 100 decoupled blocks, the updates of blocks change 400 values in covariance matrices, while the update of the full state would change 10000 values. The estimates for each block contain only information about the measurements and state transition of that block, and not information that comes from other blocks through the dependencies of the blocks at time t_0.

At time t_1 the information of the different blocks is merged in a global update that is done in two phases: First a virtual update is constructed from all blocks and applied to the full state estimate at time index t_0. Then a virtual state prediction is applied to the virtually updated full state. This order is the opposite of the usual Kalman filter update where prediction is usually done first and the update is done after that. The virtual update fuses the individual blocks so that the correlations that existed between the blocks at time t_0 are taken into account. To denote state parameters that have been updated with virtual update we use superscript v; parameters for which both the virtual update and prediction has been applied are denoted with superscript $+$.

A. Virtual update

The virtual update updates the state with a Kalman update that uses virtual observations y^v, measurement matrix J^v, and measurement covariance R^v [19]. The parameters for a virtual update are

$$y^v_{t_1} = \begin{bmatrix} y^v_{\underline{1},t_1} \\ y^v_{\underline{2},t_1} \\ y^v_{\underline{3},t_1} \\ \vdots \end{bmatrix} \tag{51}$$

$$J^v_{t_1} = I \tag{52}$$

$$R^v_{t_1} = \begin{bmatrix} R^v_{\underline{1},t_1} & \mathbf{0} & \cdots & \\ \mathbf{0} & R^v_{\underline{2},t_1} & \mathbf{0} & \ddots \\ \vdots & \mathbf{0} & R^v_{\underline{3},t_1} & \ddots \\ & \ddots & \ddots & \ddots \end{bmatrix}, \tag{53}$$

where components have the form

$$y^v_{\underline{a},t_1} = \mu_{x_{\underline{a}},t_0} + P_{x_{\underline{a}}x_{\underline{a}},t_0}\left(P_{x_{\underline{a}}x_{\underline{a}},t_0} - P_{\hat{x}_{\underline{a}}\hat{x}_{\underline{a}},t_1}\right)^{-1}\left(\hat{\mu}_{\underline{a},t_1} - \mu_{x_{\underline{a}},t_0}\right) \tag{54}$$

$$R^v_{\underline{a},t_1} = P_{x_{\underline{a}}x_{\underline{a}},t_0}\left(P_{x_{\underline{a}}x_{\underline{a}},t_0} - P_{\hat{x}_{\underline{a}}\hat{x}_{\underline{a}},t_1}\right)^{-1} P_{x_{\underline{a}}x_{\underline{a}},t_0} - P_{x_{\underline{a}}x_{\underline{a}},t_0}. \tag{55}$$

These both require the inversion of $\left(P_{x_{\underline{a}}x_{\underline{a}},t_0} - P_{\hat{x}_{\underline{a}}\hat{x}_{\underline{a}},t_1}\right)$ which may be singular. One example of it being singular is the situation of not having any measurements considering the current block. In [19], the singular situation is handled by using singular value decomposition.

In the following, we give a new formulation that does not require the singular value decomposition and is equivalent to the update presented in [19]. The formulation is derived in the Appendix. In the new formulation, the "posterior" parameters after the virtual update are

$$\mu^v_{x,t_1} = \mu_{x,t_0} + \left((P_{\hat{x}\hat{x},t_1})^{-1} - \left(P^D_{xx,t_0}\right)^{-1} + (P_{xx,t_0})^{-1}\right)^{-1} \cdot \left((P_{\hat{x}\hat{x},t_1})^{-1}\hat{\mu}_{x,t_1} - (P_{xx,t_0})^{-1}\mu_{x,t_0}\right) \tag{56}$$

$$P^v_{xx,t_1} = \left((P_{\hat{x}\hat{x},t_1})^{-1} - \left(P^D_{xx,t_0}\right)^{-1} + (P_{xx,t_0})^{-1}\right)^{-1}, \tag{57}$$

where $P_{\hat{x}\hat{x},t_1}$ and $P^D_{\hat{x}\hat{x},t_0}$ are defined as

$$P_{\hat{x}\hat{x},t_1} = \begin{bmatrix} P_{\hat{x}_{\underline{1}}\hat{x}_{\underline{1}},t_1} & \mathbf{0} & \cdots & \\ \mathbf{0} & P_{\hat{x}_{\underline{2}}\hat{x}_{\underline{2}},t_1} & \mathbf{0} & \ddots \\ \vdots & \mathbf{0} & P_{\hat{x}_{\underline{3}}\hat{x}_{\underline{3}},t_1} & \ddots \\ & \ddots & \ddots & \ddots \end{bmatrix} \tag{58}$$

$$P^D_{xx,t_0} = \begin{bmatrix} P_{x_{\underline{1}}x_{\underline{1}},t_0} & \mathbf{0} & \cdots & \\ \mathbf{0} & P_{x_{\underline{2}}x_{\underline{2}},t_0} & \mathbf{0} & \ddots \\ \vdots & \mathbf{0} & P_{x_{\underline{3}}x_{\underline{3}},t_0} & \ddots \\ & \ddots & \ddots & \ddots \end{bmatrix}. \tag{59}$$

Because these matrices are block diagonal, their inverses can be computed blockwise.

This formulation does not require inversion of singular matrices, provided that

- Prior P_{xx,t_0} has full rank
- Posterior P_{xx,t_1} has full rank

B. Virtual state propagation

After all blocks are updated as presented in the previous subsection the state is propagated with a virtual state propagation to obtain the global posterior. The state propagation model is linear

$$x = F^v(x^v - \hat{\mu}) + \mu + \varepsilon^v, \tag{60}$$

where ε^v has zero mean and covariance Q^v. Thus the posterior mean and covariance are

$$\mu^+_{x,t_1} = F^v\mu^v - F^v\hat{\mu} + \mu \tag{61}$$

$$P^+_{x,x,t_1} = F^v P^v F^{vT} + Q^v \tag{62}$$

The state propagation parameters have the following form

$$\mu = \begin{bmatrix} \mu^T_{x_{\underline{1}},t_1} & \mu^T_{x_{\underline{2}},t_1} & \cdots \end{bmatrix}^T \tag{63}$$

$$\hat{\mu} = \begin{bmatrix} \hat{\mu}^T_{x_{\underline{1}},t_1} & \hat{\mu}^T_{x_{\underline{2}},t_1} & \cdots \end{bmatrix}^T \tag{64}$$

$$F^v = \begin{bmatrix} F_{\underline{1}} & \mathbf{0} & \cdots & \\ \mathbf{0} & F_{\underline{2}} & \mathbf{0} & \ddots \\ \vdots & \mathbf{0} & F_{\underline{3}} & \ddots \\ & \ddots & \ddots & \ddots \end{bmatrix} \tag{65}$$

$$Q^v = \begin{bmatrix} Q_{\underline{1}} & \mathbf{0} & \cdots & \\ \mathbf{0} & Q_{\underline{2}} & \mathbf{0} & \ddots \\ \vdots & \mathbf{0} & Q_{\underline{3}} & \ddots \\ & \ddots & \ddots & \ddots \end{bmatrix}, \tag{66}$$

where the matrix blocks are

$$F^v_{\underline{a}} = P_{x_{\underline{a}}\hat{x}_{\underline{a}},t_1}P^{-1}_{\hat{x}_{\underline{a}}\hat{x}_{\underline{a}},t_1} \tag{67}$$

$$Q^v_{\underline{a}} = P_{x_{\underline{a}}x_{\underline{a}},t_1} - P_{x_{\underline{a}}\hat{x}_{\underline{a}},t_1}P^{-1}_{\hat{x}_{\underline{a}}\hat{x}_{\underline{a}},t_1}P_{\hat{x}_{\underline{a}}x_{\underline{a}},t_1}. \tag{68}$$

The full algorithm for filtering a model whose state has decoupled blocks is given in Algorithm 4.

C. Static blocks

In this section, we consider a common special case that allows to optimize the formulas in the previous sections further. This corresponds to the algorithm presented in [27].

Let $x_{\underline{s}}$ be a block that does not change during the estimation from time t_0 to t_1. For this block

$$\mu_{x_{\underline{s}},t_0} = \mu_{x_{\underline{s}},t_1} = \mu_{\hat{x}_{\underline{s}},t_0} = \mu_{\hat{x}_{\underline{s}},t_1} \tag{69}$$

$$P_{x_{\underline{s}}x_{\underline{s}},t_0} = P_{\hat{x}_{\underline{s}}\hat{x}_{\underline{s}},t_0} = P_{x_{\underline{s}}x_{\underline{s}},t_1} = P_{\hat{x}_{\underline{s}}\hat{x}_{\underline{s}},t_1} = P_{\hat{x}_{\underline{s}}x_{\underline{s}},t_1} \tag{70}$$

and there is no need to compute their values before the final update where the whole state is updated. We can see by looking at (54) and (55) that the formulation presented in [19] would require inversion of zero matrices.

In (57) the inverses can be computed blockwise. Because block $\underline{s}$ of $(P_{\hat{x}\hat{x},t_1})^{-1}$ and $\left(P^D_{xx,t_0}\right)^{-1}$ are identical their inverses cancel each other and so do not need to be computed. In the virtual state propagation $F^v_{\underline{s}}$ is an identity matrix and $Q^v_{\underline{s}}$ is a zero matrix. An example of applying the block algorithm is presented in Section VII-E.

VII. Example Applications

A. Fourier series

In this example, we show how the general form of Algorithm 2 is simplifed for a specific application and compare the computational complexity of the optimized version when we are using UKF as the estimation algorithm. In this example an n dimensional state that is inferred given a scalar measurement of form

$$y = a_0 + \sum_{j=1}^{k}\left(a_j\sin(jx) + b_j\cos(jx)\right) + \varepsilon^y, \tag{71}$$

Algorithm 4: KF with decoupled blocks

Input: $x_{t_0} \sim \mathrm{N}(\mu_{x,0}, P_{xx,0})$ // Prior state that can be divided into decoupled blocks
$f_{t,\underline{a}}(x_{\underline{a}}, \varepsilon^x_{t,\underline{a}})$ // State transition functions for blocks for each time step
$h_{t,\underline{a}}(x_{\underline{a}}, \varepsilon^y_{t,\underline{a}})$ // Measurement functions for blocks for each time step
// Note that each block may have different number of state transition and measurement functions between t_0 and t_1
Output: $x \sim \mathrm{N}(\mu^+_{x,t_1}, P^+_{xx,t_1})$ // Posterior estimate
for a=1 to n **do**
  $\overline{\mu}_{x_{\underline{a}}} = \begin{bmatrix} \mu_{x_{\underline{a}},t_0} \\ \mu_{x_{\underline{a}},t_0} \end{bmatrix}$ // Initialize block mean
  $\overline{P}_{x_{\underline{a}}x_{\underline{a}}} = \begin{bmatrix} P_{x_{\underline{a}},t_0 x_{\underline{a}},t_0} & P_{x_{\underline{a}},t_0 x_{\underline{a}},t_0} \\ P_{x_{\underline{a}},t_0 x_{\underline{a}},t_0} & P_{x_{\underline{a}},t_0 x_{\underline{a}},t_0} \end{bmatrix}$ // Initialize block covariance
end
for $\underline{a = 1 \text{ to } n}$ **do**
  for $\underline{t_{\underline{a}} = t_0 \text{ to } t_1}$ **do**
    Update $\overline{\mu}_{x_{\underline{a}}}$ and $\overline{P}_{x_{\underline{a}}x_{\underline{a}}}$ with the state propagation and measurement models at time $t_{\underline{a}}$ using a suitable filter and (49)–(50).
  end
end
// Update global state after each block is updated to time t_1
Compute virtual prediction mean μ^v_{x,t_1} with (56) and covariance P^v_{xx,t_1} with (57).
Compute virtual prediction mean μ^+_{x,t_1} with (61) and covariance P^+_{xx,t_1} with (62).

where $\varepsilon^y \sim \mathrm{N}(0, R)$ is independent of the state. The augmented state is $\begin{bmatrix} x \\ \varepsilon^y \end{bmatrix}$. Matrices for Algorithm 2 are $T = \begin{bmatrix} 1 & 0 & \dots \end{bmatrix}$ and $H = \begin{bmatrix} 0 & \dots & 0 & 1 \end{bmatrix}$. Using these the transformed mean is $\mu_{\tilde{z}} = \mu_{x_1}$ and the transformed covariance is $P_{\tilde{z}\tilde{z}} = P_{x_1x_1}$. Sigma-points are generated using one-dimensional mean μ_{x_1} and covariance $P_{x_1x_1}$ and the moments are

$$\begin{aligned} \mu_{g(\tilde{z})} &= \sum w_i g(\chi_i) \\ &= \sum \left(w_i a_0 + \sum_{j=1}^{k} \left(a_j \sin(j\chi_i) + b_j \cos(j\chi_i) \right) \right) \\ P_{g(\tilde{z})g(\tilde{z})} &= \sum w_i (g(\chi_i) - \mu_{g(\tilde{z})})(g(\chi_i) - \mu_{g(\tilde{z})})^T \\ P_{\tilde{z}g(\tilde{z})} &= \sum w_i (\chi_j - \mu_{\tilde{z}})(g(\chi_i) - \mu_{g(\tilde{z})})^T. \end{aligned} \tag{72}$$

and equation (33) is

$$P_{zg(\tilde{z})} = P_{zz[:,1]} P^{-1}_{zz[1,1]} P_{\tilde{z}g(\tilde{z})} \tag{73}$$

TABLE I
COMPUTATIONAL COMPLEXITY OF PARTS OF THE UPDATE OF THE STATE WITH DIFFERENT OPTIMIZATIONS

Computation of	Optimized version	Basic UKF
Sigma points	$\mathcal{O}(1)$	$\underline{\mathcal{O}(n^3)}$
$\mu_{g(z)}$	$\mathcal{O}(k)$	$\underline{\mathcal{O}(nk)}$
$P_{h(z)h(z)}$	$\mathcal{O}(k)$	$\mathcal{O}(nk)$
$P_{zg(z)}$	$\underline{\mathcal{O}(n+k)}$	$\underline{\mathcal{O}(n^2+nk)}$
K	$\mathcal{O}(n)$	$\mathcal{O}(n)$
μ	$\mathcal{O}(n)$	$\mathcal{O}(n)$
P	$\underline{\mathcal{O}(n^2)}$	$\mathcal{O}(n^2)$

Because measurement error has zero mean we have

$$\mu_{h(z)} = \mu_{g(\tilde{z})} \tag{74}$$

and, as it is independent of the state, the last element of (73) is 0. Thus, we can simplify (31) to

$$P_{h(z)h(z)} = P_{g(\tilde{z})g(\tilde{z})} + R \tag{75}$$

and (36) to

$$P_{xh(z)} = P_{zz[:,1]} P^{-1}_{zz[1,1]} P_{\tilde{z}g(\tilde{z})}. \tag{76}$$

Using the above equations we evaluate the complexity improvements when they are applied with UKF that uses $2n+1$ sigma-points. Table I shows the asymptotic complexities of application of basic UKF and optimized version. The complexities are given as functions of k to take into account the complexity of the measurement function and of n, the number of state variables. The highest complexities are underlined.

For the optimized version, the most complex parts of the update have complexities $\underline{\mathcal{O}(n+k)}$ for computing the cross correlation and $\underline{\mathcal{O}(n^2)}$ for updating the covariance matrix. The most demanding parts for basic UKF are also computing the cross correlation which has complexity $\underline{\mathcal{O}(n^2+nk)}$ and the computation of sigma points which has complexity $\underline{\mathcal{O}(n^3)}$ due to computing the matrix square root of the covariance matrix. This implies that the given optimization reduces computational load in both situations where n or k dominates.

B. Pedestrian Dead Reckoning

In Pedestrian Dead Reckoning (PDR), a pedestrian's position is estimated based on steps and heading changes that are measured with accelerometers and gyroscopes. In [32], a state consists of two-dimensional position r, heading θ, step length l, and possibly a floor index. The state transition is computed with a particle filter. Here we show how the state model without floor index and with normal noise can be optimized for use with KFEs. We use in this example algorithms presented in Sections IV-A and IV-B.

The nonlinear state transition model is

$$\begin{aligned} x_{t+1} &= \begin{bmatrix} r_{1,t+1} \\ r_{2,t+1} \\ \theta_{t+1} \\ l_{t+1} \end{bmatrix} \\ &= \begin{bmatrix} r_{1,t} + (l_t + \varepsilon^x_l) \cos(\theta_t + \varepsilon^x_\theta) + \varepsilon^x_{r,1} \\ r_{2,t} + (l_t + \varepsilon^x_l) \sin(\theta_t + \varepsilon^x_\theta) + \varepsilon^x_{r,2} \\ \theta_t + \varepsilon^x_\theta \\ l_t + \varepsilon^x_l \end{bmatrix}, \end{aligned} \tag{77}$$

where $\varepsilon_\theta^x \sim \mathrm{N}(\Delta_t, \sigma_\theta^2)$ is the noisy heading change, with Δ_t measured from gyroscopes, l is the footstep length and r is the position of the pedestrian. The augmented state is $z = \begin{bmatrix} r_1 & r_2 & \theta & l & \varepsilon_{r,1}^x & \varepsilon_{r,2}^x & \varepsilon_\theta^x & \varepsilon_l^x \end{bmatrix}^T$. The state transition model can be written in the form of (28):

$$\begin{aligned} \tilde{z} &= \begin{bmatrix} \theta_t + \varepsilon_\theta^x \\ l_t + \varepsilon_l^x \end{bmatrix} \\ T &= \begin{bmatrix} \mathbf{0}_{2\times 2} & I_{2\times 2} & \mathbf{0}_{2\times 2} & I_{2\times 2} \end{bmatrix} \\ g(\tilde{z}) &= \begin{bmatrix} \tilde{z}_2 \cos \tilde{z}_1 \\ \tilde{z}_2 \sin \tilde{z}_1 \end{bmatrix} \\ A &= \begin{bmatrix} I_{2\times 2} \\ \mathbf{0}_{2\times 2} \end{bmatrix} \\ H &= \begin{bmatrix} I_{4\times 4} I_{4\times 4} \end{bmatrix}. \end{aligned} \tag{78}$$

In this formulation the reduced state $\tilde{z}$ is two-dimensional and the nonlinear part of the state transition function is also two-dimensional.

If this problem is to be solved with a KFE that uses sigma-points, we can optimize the implementation further. This can be done because $\tilde{z}_2$ in (78) is conditionally linear, given $\tilde{z}_1$. The parts of function (37) are

$$\begin{aligned} z_n &= \tilde{z}_1 \\ z_l &= \tilde{z}_2 \\ g_n(\tilde{z}_1) &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ G_n(\tilde{z}_1) &= \begin{bmatrix} \sin \tilde{z}_1 \\ \cos \tilde{z}_1 \end{bmatrix}. \end{aligned} \tag{79}$$

Using this the dimension of the nonlinear part of the state is reduced to 1 and the sigma-points are generated for only one dimension.

The reduction of the computational complexity using these optimizations depend on the used KFE. Using QKF with parameter $\alpha = 3$ the number of sigma-points for computing (17)-(19) is reduced from 6561 to 3. With UKF the reduction is not as significant, the number of sigma-points is reduced from 17 to 3.

Figure 1 shows mean estimates of the first position variable r_1 after one propagation step in a simulation. The initial state has step length 1 and step direction $50\frac{\pi}{180}$ degrees. The initial covariance is strongly correlated being $\mathrm{diag}\left(1, 1, 1, 0.01, .01, .01, \frac{\pi^2}{180^2}, .00001\right) + \mathbf{1}$. The estimates are computed using S^{2}KF with a varying number of sigma-points. An implementation of the sigma-point generator and S^{2}KF can be found in [40]. S^{2}KF is applied to the original state (77) and to the above presented reduced states with 2 and 1 nonlinear variables. The propagation of the state with the full state model cannot be computed with fewer than 9 sigma-points.

The state computed using the model that uses conditional linearity in addition to the linearity has the smoothest convergence to an estimate and the estimate with full state estimate has the largest variation. This shows that the use of optimizations can also improve the estimation accuracy as the sigma-points will be better aligned along the dimensions with nonlinearities. However, if one would use a KFE whose number of sigma-points cannot be altered, such as having $2n+1$ sigma-points in UKF, one may end up with better (or sometimes worse) accuracy without optimizations just because there are more sigma-points. We don't compute the accuracy of the estimates in the following examples as it is dependent on the selected KFE and parameters. One can expect the accuracy to be approximately the same when the optimizations are applied.

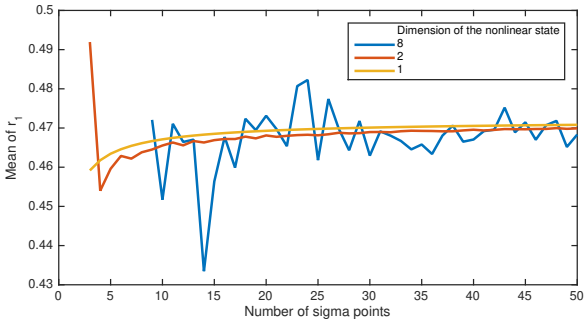

Fig. 1. Estimated mean of r_1 with different number of function evaluation points using S^{2}KF

C. Remaining useful life prediction of lithium-ion battery

In this example we apply the optimizations to prediction of the useful lifetime of a lithium-ion battery. The original model is proposed in [28].

The state consists of four estimated variables $x = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T$ and the state transition model is linear $x_{t+1} = x_t + \varepsilon^x$, where $\varepsilon^x \sim \mathrm{N}(0, Q)$ and measurement model is

$$y_t = x_{3,t} e^{t x_{1,t}} + x_{4,t} e^{t x_{2,t}} + \varepsilon^y, \tag{80}$$

where y_t is the measured capacity of the lithium-ion battery at cycle t, and ε^y has variance R.

Because the state transition model is completely linear it is evident that the propagation should be done using the linear KF, which becomes

$$\begin{aligned} \mu_{x_t}^- &= I \mu_{x_{t-1}} = \mu_{x_t} \\ P_{x_t x_t}^- &= I P_{x_{t-1} x_{t-1}} I^T + Q_t = P_{x_{t-1} x_{t-1}} + Q_t \end{aligned} \tag{81}$$

This prediction requires 16 additions when Q is added to the state covariance, or 4 if Q is diagonal. If the prediction were computed with UKF and an augmented state, there would be 11 sigma-points with 5 elements each, thus computation of predicted covariance (18) requires 550 products and 275 summations. In addition, generation of sigma points involving matrix square root computation and computation of the mean (17) have to be performed. The absolute numbers of computations are small for modern computers either way, but reducing the number of computations by a factor greater than 200 is a significant reduction of operations.

The measurement model has one additive component, ε^y, and we can use (24)-(26) for it. Variables x_3 and x_4 are conditionally linear and we can apply equations from Section IV-B.

In this case

$$\begin{aligned}\mu_{x_l} &= \mu_{[3,4]}\\ \mu_{x_n} &= \mu_{[1,2]}\\ P_{x_l x_l} &= P_{[3,4],[3,4]}\\ P_{x_l x_n} &= P_{[3,4],[1,2]}\\ P_{x_n x_n} &= P_{[1,2],[1,2]}\\ g_n(\chi_i) &= 0\\ G_n(\chi_i) &= \begin{bmatrix} e^{t\mu_1} & e^{t\mu_2}\end{bmatrix}\end{aligned} \tag{82}$$

These can be used to compute $\mu_{x_l|\chi_i}$ (38) and $P_{x_l x_l|\chi_i}$ (39). Merging (24)-(26) and (40)-(43) gives us

$$\begin{aligned}\mathcal{Y}_i =& G_n(\chi_i)\mu_{x_l|\chi_i}\\ \mu_{g(z)} =& \sum w_i \mathcal{Y}_i\\ P_{g(z)g(z)} =& \sum w_i \left((\mathcal{Y}_i - \mu_{g(z)})(\mathcal{Y}_i - \mu_{g(z)})^T\right.\\ &\left. + G_n(\chi_i) P_{x_l x_l} G_n(\chi_i)^T\right) + R\\ P_{z g(z)} =& \sum w_i \left(\left(\begin{bmatrix}\chi_i\\ \mu_{z_l|\chi_i}\end{bmatrix} - \mu_x\right)(\mathcal{Y}_i - \mu_{g(z)})^T\right.\\ &\left. + \begin{bmatrix}0\\ P_{x_l x_l|\chi_i} G_n(\chi_i)^T\end{bmatrix}\right).\end{aligned} \tag{83}$$

In this situation, the computation gains come mostly from the reduction of the number of sigma-points as the dimension is reduced from 5 to 2. When using UKF the number of sigma-points is reduced from 11 to 5. However, the above computations are more complex for each summand than if the moments would have been computed using (17)-(19). So the optimizations based on conditional linearity are not reducing the computational complexity. However, when using QKF with parameter $\alpha = 4$, the number of sigma-points is reduced from $4^5 = 1024$ to $4^2 = 16$ and the optimizations are useful.

D. Source Tracking Using a Microphone Array Time Difference of Arrival (TDoA)

In tracking of a sound source using a microphone array, the received acoustic signals of two microphones are compared so that the TDoA of the acoustic signal is measured. In the ideal situation the TDoA is

$$h_{i,j}(r_0) = \frac{\|r_i - r_0\| - \|r_j - r_0\|}{v}, \tag{84}$$

where r_0 is the location of the sound source, r_i is the location of the ith microphone and v is the speed of sound. When there are m microphones in the array there are at most $\frac{1}{2}m(m-1)$ TDoA measurements available [33].

In practice, measurements contain noise. Here we consider model with noises

$$h_{i,j}(x) = \|r_i - x\| - \|r_j - x\| + \varepsilon_i^y - \varepsilon_j^y + \varepsilon_{i,j}^y, \tag{85}$$

where ε_i is the noise corresponding to ith microphone and $\varepsilon_{i,j}$ is the error corresponding to the correlation of the measured signals from the microphone pair. The speed of sound is multiplied out from the equation.

When $\varepsilon_{i,j} = 0$ for all microphone pairs the measurement noise covariance is not full rank and the measurement equations can be modeled with $m-1$ measurements of the form

$$h_i(x) = \|r_i - x\| - \|r_m - x\| + \varepsilon_i^y - \varepsilon_m^y, \tag{86}$$

which has a full rank noise covariance matrix. This kind of assumption is done for example in [4]. In practice, this assumption does not hold and the selection of microphone pairs is a tradeoff, where pairs close to each other have smaller correlation error $\varepsilon_{i,j}$, but worse geometry than pairs far away from each other [38]. Here we consider the situation where all possible microphone pairs are used and errors ε_i and $\varepsilon_{i,j}$ are modeled as Gaussians. The augmented state model is

$$z = \begin{bmatrix} x^T & \varepsilon_{1,1}^y & \varepsilon_{1,2}^y & \dots & \varepsilon_{m,m}^y & \varepsilon_1^y & \dots & \varepsilon_m^y\end{bmatrix}^T, \tag{87}$$

where x contains 3 position and velocity variables.

Using (24)–(26) the nonlinear part of the estimation can be done only for the state variables and sigma-points would be required only for the 6 dimensional state instead of using sigma-points also for $\frac{1}{2}m(m+1)$ noise terms. The measurement model can be written in compact form using (28):

$$\begin{aligned}\tilde{z} &= x_{1:3}\\ T &= \begin{bmatrix} I_{3\times 3} & \mathbf{0_{3\times 3}}\end{bmatrix}\\ g(\tilde{z}) &= \begin{bmatrix}\|r_1 - \tilde{z}\| & \dots & \|r_m - \tilde{z}\|\end{bmatrix}^T\\ A &= \begin{bmatrix} \mathbf{1}_{m-1\times 1} & -I_{m-1\times m-1} & \\ \mathbf{0}_{m-2\times 1} & \mathbf{1}_{m-2\times 1} & -I_{m-2\times m-2}\\ \mathbf{0}_{m-3\times 2} & \mathbf{1}_{m-3\times 1} & -I_{m-3\times m-3}\\ & \vdots & \\ \mathbf{0}_{1\times m-2} & 1 & -1\end{bmatrix}\\ H &= \begin{bmatrix}\mathbf{0}_{\frac{m(m-1)}{2}\times 6} & A & I_{\frac{m(m-1)}{2}\times\frac{m(m-1)}{2}}\end{bmatrix}.\end{aligned} \tag{88}$$

Using these the dimension of the nonlinear function is reduced from $\frac{1}{2}m(m-1)$ to m. This reduces the number of updated elements for each sigma-point in (18) from $\left(\frac{m(m-1)}{2}\right)^2$ to m^2. A and H matrices are sparse and an application of sparse linear algebra codes would further enhance the performance of the algorithm.

The dimension of the innovation covariance S is $\left(\frac{m(m-1)}{2}\right)^2 \times \left(\frac{m(m-1)}{2}\right)^2$. The computation of the inverse of the innovation covariance for Kalman gain (7) can be optimized using the inversion formula (45). The noise terms are assumed independent so the noise covariance matrix R is diagonal. We partition noises into two parts, and denote the covariance matrices D_1 and D_2. D_1 corresponds to microphone specific terms ε_i^y and D_2 to microphone pair specific terms $\varepsilon_{i,j}^y$. Because $\tilde{z}$ contains only state variables and they do not contribute to the linear part ($AP_{zg(\tilde{z})^T}H^T = \mathbf{0}$), the innovation covariance can be written as

$$S = D_2 + A(P_{g(\tilde{z})g(\tilde{z})} + D_1)A^T \tag{89}$$

and its inverse is

$$S^{-1} = D_2^{-1} + D_2^{-1}AQ^{-1}A^T D_2^{-1}, \tag{90}$$

where $Q = \left[(P_{g(\tilde{z})g(\tilde{z})} + D_1)^{-1} + AD_2^{-1}A^T\right]$. Because D_2^{-1} is diagonal its inverse is diagonal with the reciprocal of the diagonal elements of D_2 on its diagonal. Other inverses that appear in this formula are applied to $m \times m$ matrices instead of $\left(\frac{m(m-1)}{2}\right)^2 \times \left(\frac{m(m-1)}{2}\right)^2$ matrices. If the used matrix

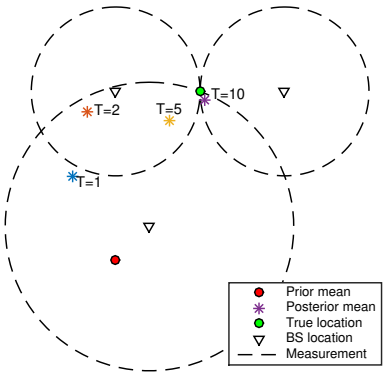

Fig. 2. Posterior means computed with RUF with varying number of iterations.

inversion algorithm has complexity of $\mathcal{O}(k^3)$, the complexity of the inversion operation is reduced from $\mathcal{O}(m^{12})$ to $\mathcal{O}(m^3)$.

E. Optimization of Iterative KFEs

The optimizations based on the division of the state into multiple sub-blocks in Section VI are most widely used in the field of Simultaneous Localization and Mapping (SLAM) [15], [19], [20]. In SLAM, the state contains a dynamic part that represents the agent (robot) that is mapping static landmarks in the environment and localizing itself. The landmark coordinates are the static part of the state.

We propose that these optimizations can be applied also to KFEs that do the update iteratively e.g. Recursive Update Filter (RUF) [47] and its extension for sigma-point filters [23] or posterior linearization filters [16], [36]. In RUF, the state is updated by applying the update in parts that have smaller impact than the original update. After each part of the update the linearization is recomputed. These iterative filters are developed assuming additive Gaussian noise. Instead of making the updates for the full state the update could be done iteratively only for the observed variables. Because the iterative update is actually for a single observation, the unobserved state variables are static during the partial updates.

In this example we consider a state model with 3D position, velocity, and acceleration i.e. 9 state variables. Measurements used are range measurements from 3 base stations. The measurement model for the range from the ith base station is

$$y_i = \left\| x_{[1:3]} - r_i \right\| + \varepsilon_i^y, \tag{91}$$

where r_i is the location of the ith base station. In our example, the prior has a large variance and its mean is chosen so that the linearization about the mean is not accurate. Figure 2 shows the example situation and the posterior means which are computed with different number of iterations of RUF. The estimate with one iteration ($T = 1$) is identical to the EKF estimate and is not close to the true location. The estimate with 10 iterations is close to the true location. The computation of 10 iterations with RUF involves the updates to the 9×9 state covariance matrices 10 times. Because the measurement model (91) depends only on the three first state variables this block can be updated 10 times using methods presented in Section VI and the remaining variables can be left in a block that does not need any update until the full covariance update. The full covariance update is computed after all iterations are applied. In this update scheme, the 3×3 covariance matrix is updated T times and the 9×9 matrix is updated only once using (56)–(68). Thus, in each iteration the covariance update is 9 times faster. Moreover, if the KFE computes sigma-points using Cholesky decomposition, as UKF does, there is more gain in speed. This is because the complexity of Cholesky decomposition is $\mathcal{O}(n^3)$.

TABLE II
SUMMARY OF PRESENTED OPTIMIZATIONS

Section	Exploited structure
Section IV-A	Partially linear functions $f(z) = Ag(Tz) + Hz$
Section IV-B	Conditionally linear functions $f(z) = g_n(z_n) + G(z_n)z_l$
Section V-A	Measurement covariance is block diagonal
Section V-B	Innovation covariance of form $S = D_2 + A(P_{g(\bar{z})g(\bar{z})} + D_1)A^T$
Section VI	State can be divided into multiple subprocesses

VIII. Conclusions

In this tutorial, we presented different optimizations for KFEs that exploit the structure of the state and measurement models to reduce the computational load. Table II summarizes the structures of models that are exploited in different optimizations. These structures are present in various nonlinear estimation problems.

Table III shows some properties of selected KFEs. Different optimizations give different amounts of speed increase with different KFEs. Optimizations presented in sections IV-A and IV-B are most useful with KFEs that have a high number of function evaluations as a function of state dimension. The exploitation of conditionally linear part of functions in Section IV-B requires that the expectations are approximated with equations of form (17)–(19).

In addition to surveying existing code optimization methods and using a general unified notation that allows them to be used with various KFEs, we make the following contributions:

1) We point out the possibility to use a linear transformation to find the minimal nonlinear subspace (Section IV-A).
2) We introduce an algorithm for systems with conditionally linear states (Section IV-B); it can be used to compute the moments for the nonlinear part and solve others using linearity (Section IV-A) as is done in the example in Section VII-B.
3) We present a new formulation for estimation with a state that can be divided into separate estimation processes (Section VI). The new formulation avoids dealing with inverses of singular matrices.
4) We present an example showing that in addition to gaining increase in speed of computation the optimization algorithms may also lead to better estimation accuracy (Section VII-B).
5) We show how to use the matrix inversion lemma to reduce the computational complexity in TDoA-based positioning (Section VII-D).

TABLE III
SUMMARY OF PROPERTIES OF SELECTED KFES

Algorithm	Order of measurement function evaluations α = algorithm specific parameter ($\alpha > 0$) j = number of iterations	Computation of J A=analyticall I=using (20) N=numerically inside algorithm	Expectations of form (17)–(19)	Iterative algorithm
EKF [26]	1	A		
EKF2 [26]	1	A		
UKF [46]	n	I	X	
CKF [2]	n^α	I	X	
S^2KF [41]	$n+\alpha$	I	X	
GF [24]	αn	I	X	
QKF [25]	α^n	I	X	
DD [31]	n	N		
SOKF2 [25]	n^2	N		
RUF [47]	j	A		X

6) We present an example showing that the optimization that exploits a static state and partly unobserved variables can be applied with iterative KFEs (Section VII-E).

Optimizations in this tutorial were applied to KFEs that use a mean vector and covariance matrix to represent the state. However, there is another flavour of KFs, called square root filters, that propagate a square root of the covariance matrix instead of the full matrix for better numerical properties [1], [6], [13], [44]. In [9], square root filtering is optimized for the situation where some of the state variables are linear and applying independent measurements sequentially as in Section V-A is straightforward. However, most of the optimizations presented in this tutorial have not been considered for square-root form in literature and this remains an open topic.

Another interesting topic is whether optimizations such as the ones in this tutorial could be applied with EnKF. EnKF is mainly used for problems whose state space dimension is so large that it is infeasible to work with the covariance matrix. In such a context, one can expect that the effort invested in finding and implementing computation optimizations would be especially worthwhile.

REFERENCES

[1] ARASARATNAM, I., AND HAYKIN, S. Square-root quadrature Kalman filtering. IEEE Transactions on Signal Processing 56, 6 (June 2008), 2589–2593.

[2] ARASARATNAM, I., AND HAYKIN, S. Cubature Kalman filters. IEEE Transactions on Automatic Control 54, 6 (June 2009), 1254–1269.

[3] ARNOLD, S. S., NUZZACI, R., AND GORDON-ROSS, A. Energy budgeting for CubeSats with an integrated FPGA. In 2012 IEEE Aerospace Conference (March 2012), pp. 1–14.

[4] BECHLER, D., SCHLOSSER, M. S., AND KROSCHEL, K. System for robust 3d speaker tracking using microphone array measurements. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Sendai, Japan, 2004), vol. 3, IEEE, pp. 2117–2122.

[5] BEUTLER, F., HUBER, M. F., AND HANEBECK, U. D. Gaussian filtering using state decomposition methods. In Proceedings of the 12th International Conference on Information Fusion (FUSION) (Seattle, WA, USA, July 2009), IEEE, pp. 579–586.

[6] BIERMAN, G. J. Factorization Methods for Discrete Sequential Estimation. Academic Press, Inc., New York, 1977.

[7] BRIERS, M., MASKELL, S. R., AND WRIGHT, R. A Rao-Blackwellised Unscented Kalman filter. In Proceedings of the 6th International Conference of Information Fusion (FUSION) (Cairns, Australia, July 2003), vol. 1, IEEE, pp. 55–61.

[8] CLOSAS, P., FERNANDEZ-PRADES, C., AND VILA-VALLS, J. Multiple quadrature kalman filtering. IEEE Transactions on Signal Processing 60, 12 (Dec 2012), 6125–6137.

[9] CLOSAS, P., AND FERNÁNDEZ-PRADES, C. The marginalized square-root quadrature Kalman filter. In 2010 IEEE 11th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC) (June 2010), pp. 1–5.

[10] CLOSAS, P., VILÀ-VALLS, J., AND FERNÁNDEZ-PRADES, C. Computational complexity reduction techniques for quadrature kalman filters. In 2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP) (Dec 2015), pp. 485–488.

[11] DA COSTA, S. C. L., ARAUJO, A. S., AND D. S. CARVALHO, A. Battery state of charge estimation using extended Kalman filter. In 2016 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM) (June 2016), pp. 1085–1092.

[12] DAVISON, A. J. Mobile Robot Navigation Using Active Vision. PhD thesis, Department of Engineering Science, University of Oxford, 1998.

[13] DER MERWE, R. V., AND WAN, E. A. The square-root unscented Kalman filter for state and parameter-estimation. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '01) (2001), vol. 6, pp. 3461–3464 vol.6.

[14] DOUCET, A., FREITAS, N. D., MURPHY, K. P., AND RUSSELL, S. J. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (San Francisco, CA, USA, 2000), UAI '00, Morgan Kaufmann Publishers Inc., pp. 176–183.

[15] ESTRADA, C., NEIRA, J., AND TARDOS, J. D. Hierarchical slam: Real-time accurate mapping of large environments. IEEE Transactions on Robotics 21, 4 (Aug 2005), 588–596.

[16] GARCÍA-FERNÁNDEZ, A. F., SVENSSON, L., MORELANDE, M. R., AND SÄRKKA, S. Posterior linearisation filter: principles and implementation using sigma points. IEEE Transactions on Signal Processing, 99 (2015), 1–1.

[17] GOLUB, G. H., AND VAN LOAN, C. F. Matrix computations, vol. 3. JHU Press, 2012.

[18] GREWAL, M. S., AND ANDREWS, A. P. Applications of Kalman filtering in aerospace 1960 to the present [historical perspectives]. IEEE Control Systems Magazine 30, 3 (June 2010), 69–78.

[19] GUIVANT, J. E. The generalized compressed Kalman filter. Robotica 35, 8 (2017), 1639–1669.

[20] GUIVANT, J. E., AND NEBOT, E. M. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. IEEE Transactions on Robotics and Automation 17, 3 (Jun 2001), 242–257.

[21] HILTUNEN, P., SÄRKKÄ, S., NISSILÄ, I., LAJUNEN, A., AND LAMPINEN, J. State space regularization in the nonstationary inverse problem for diffuse optical tomography. Inverse Problems 27, 2 (2011), 025009.

[22] HO, Y.-C., AND LEE, R. A Bayesian approach to problems in stochastic estimation and control. IEEE Transactions on Automatic Control 9, 4 (October 1964), 333 – 339.

[23] HUANG, Y., ZHANG, Y., LI, N., AND ZHAO, L. Design of sigma-point Kalman filter with recursive updated measurement. Circuits, Systems, and Signal Processing (2015), 1–16.

[24] HUBER, M. F., AND HANEBECK, U. D. Gaussian filter based on deterministic sampling for high quality nonlinear estimation. In Proceedings of the 17th {IFAC} World Congress (2008), vol. 41, pp. 13527 – 13532.

[25] ITO, K., AND XIONG, K. Gaussian filters for nonlinear filtering problems. IEEE Transactions on Automatic Control 45, 5 (May 2000), 910–927.

[26] JAZWINSKI, A. H. Stochastic Processes and Filtering Theory. Academic Press, New York, NY, USA, 1970.

[27] LEFEBVRE, T., BRUYNINCKX, H., AND SCHUTTER, J. D. Nonlinear Kalman Filtering for Force-Controlled Robot Tasks, vol. 19 of Springer Tracts in Advanced Robotics. Springer-Verlag, Berlin, Germany, 2005.

[28] MIAO, Q., XIE, L., CUI, H., LIANG, W., AND PECHT, M. Remaining useful life prediction of lithium-ion battery with unscented particle filter technique. Microelectronics Reliability 53, 6 (2013), 805 – 810.

[29] MORELANDE, M. R., AND MORAN, B. An unscented transformation for conditionally linear models. In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP) (Honolulu, HI, USA, April 2007), vol. 3, IEEE, pp. III–1417–III–1420.

[30] MORELANDE, M. R., AND RISTIC, B. Reduced sigma point filtering for partially linear models. In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP) (Toulouse, France, May 2006), vol. 3, IEEE, pp. III–III.

[31] NØRGAARD, M., POULSEN, N. K., AND RAVN, O. New developments in state estimation for nonlinear systems. Automatica 36, 11 (Nov. 2000), 1627–1638.

[32] NURMINEN, H., RISTIMAKI, A., ALI-LÖYTTY, S., AND PICHÉ, R. Particle filter and smoother for indoor localization. In Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN) (Montbéliard, France, Oct 2013), IEEE, pp. 1–10.

[33] OUALIL, Y., FAUBEL, F., DOSS, M. M., AND KLAKOW, D. A TDOA Gaussian mixture model for improving acoustic source tracking. In Proceedings of the 20th European Signal Processing Conference (EUSIPCO) (Bucharest, Romania, 2012), IEEE, pp. 1339–1343.

[34] RAITOHARJU, M., GARCÍA-FERNÁNDEZ, A. F., AND PICHÉ, R. Kullback–Leibler divergence approach to partitioned update Kalman filter. Signal Processing 130 (2017), 289 – 298.

[35] RAITOHARJU, M., PICHÉ, R., ALA-LUHTALA, J., AND ALI-LÖYTTY, S. Partitioned update Kalman filter. Journal of Advances in Information Fusion (June 2016), 3–14. In press.

[36] RAITOHARJU, M., SVENSSON, L., GARCIA-FERNANDEZ, A. F., AND PICHE, R. Damped posterior linearization filter. IEEE Signal Processing Letters (2018). In press.

[37] SÄRKKÄ, S. Bayesian filtering and smoothing, vol. 3. Cambridge University Press, Cambridge, UK, 2013.

[38] SILVERMAN, H. F., PATTERSON, W. R., AND FLANAGAN, J. L. The huge microphone array. IEEE Concurrency 6, 4 (1998), 36–46.

[39] SORENSON, H. W., AND ALSPACH, D. L. Recursive Bayesian estimation using Gaussian sums. Automatica 7, 4 (1971), 465–479.

[40] STEINBRING, J. Nonlinear estimation toolbox, Accessed: Dec. 8. 2015.

[41] STEINBRING, J., AND HANEBECK, U. D. LRKF Revisited: The Smart Sampling Kalman Filter (S2KF). Journal of Advances in Information Fusion 9, 2 (Dec. 2014), 106 – 123.

[42] STENGEL, R. F. Stochastic optimal control. Wiley-Interscience, New York, 1986.

[43] STRASSEN, V. Gaussian elimination is not optimal. Numerische mathematik 13, 4 (1969), 354–356.

[44] THORNTON, C. L. Triangular Covariance Factorizations for Kalman Filtering. PhD thesis, NASA, 1993. NASA Technical Memorandum 33-798.

[45] VILÀ-VALLS, J., CLOSAS, P., AND GARCÍA-FERNÁNDEZ, Á. F. Uncertainty exchange through multiple quadrature kalman filtering. IEEE Signal Processing Letters 23, 12 (Dec 2016), 1825–1829.

[46] WAN, E. A., AND VAN DER MERWE, R. The unscented Kalman filter for nonlinear estimation. In Proceedings of the Adaptive Systems for Signal Processing, Communications, and Control Symposium (AS-SPCC) (Lake Louise, AB, Canada, 2000), IEEE, pp. 153–158.

[47] ZANETTI, R. Recursive update filtering for nonlinear estimation. IEEE Transactions on Automatic Control 57, 6 (June 2012), 1481–1490.

APPENDIX

Derivation of the virtual update that does not require the inversion of a possibly singular matrix. First, we define matrices $P_{\hat{x}\hat{x},t_1}$ and $P^D_{\hat{x}\hat{x},t_0}$

$$P_{\hat{x}\hat{x},t_1} = \begin{bmatrix} P_{\hat{x}_{\underline{1}}\hat{x}_{\underline{1}},t_1} & \mathbf{0} & \dots & \\ \mathbf{0} & P_{\hat{x}_{\underline{2}}\hat{x}_{\underline{2}},t_1} & \mathbf{0} & \ddots \\ \vdots & \mathbf{0} & P_{\hat{x}_{\underline{3}}\hat{x}_{\underline{3}},t_1} & \ddots \\ & \ddots & \ddots & \ddots \end{bmatrix} \tag{92}$$

$$P^D_{xx,t_0} = \begin{bmatrix} P_{x_{\underline{1}}x_{\underline{1}},t_0} & \mathbf{0} & \dots & \\ \mathbf{0} & P_{x_{\underline{2}}x_{\underline{2}},t_0} & \mathbf{0} & \ddots \\ \vdots & \mathbf{0} & P_{x_{\underline{3}}x_{\underline{3}},t_0} & \ddots \\ & \ddots & \ddots & \ddots \end{bmatrix} \tag{93}$$

The virtual update covariance $R^v_{t_1}$ can be written using these and the matrix inversion lemma

$$\begin{aligned} R^v_{t_1} &= P^D_{xx,t_0}\left(P^D_{xx,t_0} - P_{\hat{x}\hat{x},t_1}\right)^{-1} P^D_{xx,t_0} - P^D_{xx,t_0} \\ &= \left(\left(P_{\hat{x}\hat{x},t_1}\right)^{-1} - \left(P^D_{xx,t_0}\right)^{-1}\right)^{-1} \end{aligned} \tag{94}$$

The predicted measurement is

$$\begin{aligned} y^v_{t_1} &= \mu_{x,t_0} + P^D_{xx,t_0}\left(P^D_{xx,t_0} - P_{\hat{x}\hat{x},t_1}\right)^{-1}\left(\hat{\mu}_{t_1} - \mu_{x,t_0}\right) \\ &= \mu_{x,t_0} + P^D_{xx,t_0}\left(\left(P^D_{xx,t_0}\right)^{-1} - \left(P^D_{xx,t_0}\right)^{-1}\left(\left(P^D_{xx,t_0}\right)^{-1} - \left(P_{\hat{x}\hat{x},t_1}\right)^{-1}\right)^{-1}\left(P^D_{xx,t_0}\right)^{-1}\right)\left(\hat{\mu}_{t_1} - \mu_{x,t_0}\right) \\ &= \mu_{x,t_0} + \left(I - \left(\left(P^D_{xx,t_0}\right)^{-1} - \left(P_{\hat{x}\hat{x},t_1}\right)^{-1}\right)^{-1}\left(P^D_{xx,t_0}\right)^{-1}\right)\left(\hat{\mu}_{t_1} - \mu_{x,t_0}\right) \\ &= \hat{\mu}_{t_1} - \left(\left(P^D_{xx,t_0}\right)^{-1} - \left(P_{\hat{x}\hat{x},t_1}\right)^{-1}\right)^{-1}\left(P^D_{xx,t_0}\right)^{-1}\left(\hat{\mu}_{t_1} - \mu_{x,t_0}\right) \end{aligned} \tag{95}$$

The Kalman gain is

$$\begin{aligned} K^v_{t_1} &= P_{xx,t_0}\left(P_{xx,t_0} + \left(R^v_{t_1}\right)^{-1}\right)^{-1} \\ &= P_{xx,t_0}\left(\left(P_{xx,t_0}\right)^{-1} - \left(P_{xx,t_0}\right)^{-1}\left(\left(R^v_{t_1}\right)^{-1} + \left(P_{xx,t_0}\right)^{-1}\right)^{-1}\left(P_{xx,t_0}\right)^{-1}\right) \\ &= I - \left(\left(R^v_{t_1}\right)^{-1} + \left(P_{xx,t_0}\right)^{-1}\right)^{-1}\left(P_{xx,t_0}\right)^{-1} \\ &= \left(\left(R^v_{t_1}\right)^{-1} + \left(P_{xx,t_0}\right)^{-1}\right)^{-1}\left(\left(R^v_{t_1}\right)^{-1} + \left(P_{xx,t_0}\right)^{-1}\right) - \left(\left(R^v_{t_1}\right)^{-1} + \left(P_{xx,t_0}\right)^{-1}\right)^{-1}\left(P_{xx,t_0}\right)^{-1} \\ &= \left(\left(R^v_{t_1}\right)^{-1} + \left(P_{xx,t_0}\right)^{-1}\right)^{-1}\left(R^v_{t_1}\right)^{-1} \\ &= \left(\left(P_{\hat{x}\hat{x},t_1}\right)^{-1} - \left(P^D_{xx,t_0}\right)^{-1} + \left(P_{xx,t_0}\right)^{-1}\right)^{-1}\left(\left(P_{\hat{x}\hat{x},t_1}\right)^{-1} - \left(P^D_{xx,t_0}\right)^{-1}\right) \end{aligned} \tag{96}$$

The posterior mean is

$$\begin{aligned} \mu^v_{x,t_1} &= \mu_{x,t_0} + K^v_{t_1}\left(y^v_{t_1} - \mu_{x,t_0}\right) \\ &= \mu_{x,t_0} + \left(\left(P_{\hat{x}\hat{x},t_1}\right)^{-1} - \left(P^D_{xx,t_0}\right)^{-1} + \left(P_{xx,t_0}\right)^{-1}\right)^{-1}\left(\left(P_{\hat{x}\hat{x},t_1}\right)^{-1} - \left(P^D_{xx,t_0}\right)^{-1}\right) \\ &\times \left(\hat{\mu}_{x,t_1} - \left(\left(P^D_{xx,t_0}\right)^{-1} - \left(P_{\hat{x}\hat{x},t_1}\right)^{-1}\right)^{-1}\left(P^D_{xx,t_0}\right)^{-1}\left(\hat{\mu}_{x,t_1} - \mu_{x,t_0}\right)\right) \\ &= \mu_{x,t_0} + \left(\left(P_{\hat{x}\hat{x},t_1}\right)^{-1} - \left(P^D_{xx,t_0}\right)^{-1} + \left(P_{xx,t_0}\right)^{-1}\right)^{-1} \\ &\times \left(\left(\left(P_{\hat{x}\hat{x},t_1}\right)^{-1} - \left(P^D_{xx,t_0}\right)^{-1}\right)\hat{\mu}_{x,t_1} + \left(P^D_{xx,t_0}\right)^{-1}\left(\hat{\mu}_{x,t_1} - \mu_{x,t_0}\right)\right) \\ &= \mu_{x,t_0} + \left(\left(P_{\hat{x}\hat{x},t_1}\right)^{-1} - \left(P^D_{xx,t_0}\right)^{-1} + \left(P_{xx,t_0}\right)^{-1}\right)^{-1}\left(\left(P_{\hat{x}\hat{x},t_1}\right)^{-1}\hat{\mu}_{x,t_1} - \left(P_{xx,t_0}\right)^{-1}\mu_{x,t_0}\right) \end{aligned} \tag{97}$$

The posterior covariance is

$$\begin{aligned} P^v_{xx,t_1} &= P_{xx,t_0} - K^v_{t_1}\left(P_{xx,t_0} + \left(R^v_{t_1}\right)^{-1}\right) K^{v\,T}_{t_1} \\ &= P_{xx,t_0} - P_{xx,t_0}\left((P_{\hat{x}\hat{x},t_1})^{-1} - \left(P^D_{xx,t_0}\right)^{-1}\right)\left((P_{\hat{x}\hat{x},t_1})^{-1} - \left(P^D_{xx,t_0}\right)^{-1} + (P_{xx,t_0})^{-1}\right)^{-1} \\ &= P_{xx,t_0}\left((P_{\hat{x}\hat{x},t_1})^{-1} - \left(P^D_{xx,t_0}\right)^{-1} + (P_{xx,t_0})^{-1}\right)\left((P_{\hat{x}\hat{x},t_1})^{-1} - \left(P^D_{xx,t_0}\right)^{-1} + (P_{xx,t_0})^{-1}\right)^{-1} \\ &- P_{xx,t_0}\left((P_{\hat{x}\hat{x},t_1})^{-1} - \left(P^D_{xx,t_0}\right)^{-1}\right)\left((P_{\hat{x}\hat{x},t_1})^{-1} - \left(P^D_{xx,t_0}\right)^{-1} + (P_{xx,t_0})^{-1}\right)^{-1} \\ &= \left((P_{\hat{x}\hat{x},t_1})^{-1} - \left(P^D_{xx,t_0}\right)^{-1} + (P_{xx,t_0})^{-1}\right)^{-1}. \end{aligned} \tag{98}$$

The posterior mean and covariance contain the inverses $(P_{\hat{x}\hat{x},t_1})^{-1}$, $\left(P^D_{xx,t_0}\right)^{-1}$, $(P_{xx,t_0})^{-1}$, and $\left((P_{\hat{x}\hat{x},t_1})^{-1} - \left(P^D_{xx,t_0}\right)^{-1} + (P_{xx,t_0})^{-1}\right)^{-1}$. If measurements have non-degenerate noise, the posterior $(P_{\hat{x}\hat{x},t_1})^{-1}$ is positive definite. Matrices $\left(P^D_{xx,t_0}\right)^{-1}$ and $(P_{xx,t_0})^{-1}$ depend only on the prior which is assumed positive definite. The posterior has smaller or equal covariance than the prior, thus $(P_{\hat{x}\hat{x},t_1})^{-1} - \left(P^D_{xx,t_0}\right)^{-1}$ is positive semidefinite and the sum of this and the positive definite matrix $(P_{xx,t_0})^{-1}$ is positive definite. Thus, assuming that the prior covariance is positive definite and that the posterior is positive definite, the matrix inverses in (98) are applied only to positive definite matrices.

Matti Raitoharju received M.Sc. and Ph.D. degrees in Mathematics in Tampere University of Technology, Finland, in 2009 and 2014 respectively. He works as a signal processing specialist at an aerospace and defence company Patria and as a postdoctoral researcher at Aalto University. He is a visiting scholar at Tampere University. His scientific interests include mathematical modeling and development and application of Kalman type filters.

Robert Piché received the Ph.D. degree in civil engineering in 1986 from the University of Waterloo, Canada. Since 2004 he holds a professorship in mathematics at Tampere University of Technology (now Tampere University). , Finland. His scientific interests include mathematical and statistical modelling, systems theory, and applications in positioning, computational finance, and mechanics.