Computationally Relaxed Unscented Kalman Filter

József Kuti, *Member, IEEE*, Imre J. Rudas, *Life Fellow, IEEE*,
Huijun Gao, and Péter Galambos, *Member, IEEE*

***Abstract*—Advanced robotics and autonomous vehicles rely on filtering and sensor fusion techniques to a large extent. These mobile applications need to handle the computations onboard at high rates while the computing capacities are limited. Therefore, any improvement that lowers the CPU time of the filtering leads to more accurate control or longer battery operation. This article introduces a generic computational relaxation for the unscented transformation (UT) that is the key operation of the Unscented Kalman filter-based applications. The central idea behind the relaxation is to pull out the linear part of the filtering model and avoid the calculations for the kernel of the nonlinear part. The practical merit of the proposed relaxation is demonstrated through a simultaneous localization and mapping (SLAM) implementation that underpins the superior performance of the algorithm in the practically relevant cases, where the nonlinear dependencies influence only an affine subspace of the image space. The numerical examples show that the computational demand can be mitigated below 50% without decreasing the accuracy of the approximation. The method described in this article is implemented and published as an open-source C++ library *RelaxedUnscentedTransformation* on GitHub.**

***Index Terms*—Computational relaxation, Kalman filter, sensor fusion, unscented Kalman filter (UKF), unscented transformation (UT).**

I. Introduction

THE UNSCENTED transformation (UT) was proposed to approximate the distribution of the results of nonlinear mappings performed on statistical variables that is a crucial problem in Kalman filters [1], [2]. It is applied in remote state estimation with stochastic event-triggered sensor schedule [3], to test multiagent communication schemes [4] and robust, adaptive, and consensus aim are also formulated in the framework, see [5].

The extended Kalman filter (EKF) assumes that the expected value of the resulted distribution equals the mapped expected value and takes into consideration only its local environment via linearization. In contrast, the UT applies so-called sigma points around the expected value according to the distribution of the stochastic variable. By applying the nonlinear mapping on these sigma points, the expected value and the distribution can be estimated in a more accurate way than via the local linearization of EKF. The UT-based Kalman filter is referred to as unscented Kalman filter (UKF) [2].

The methods are applied in distributed filtering [6], consensus filter [7], finite-horizon EKF [8], and the robustness of distributed filters is also analyzed [9].

Different Kalman-filter variants rely on various approximation methods representing a wide range of accuracy and computational complexity. Considering for example the simultaneous localization and mapping (SLAM) applications there are large differences not only in the applied mapping method, but in the prediction techniques as well. The FastSLAM algorithm based on EKF and particle filtering [10] is well known. The uFastSLAM based on UKF and particle filtering [11] showed clearly the better properties of UKF [12], [13]. A purely particle filter (PF)-based application would give the more accurate results but it is hard to manage the necessary computational complexity in realtime applications. Regarding the accuracy and computational cost the UKF is between the EKF and the PF [14].

The main difference of UKF (and EKF) compared to PF is the fact, that the former ones approximate the distribution with a Gaussian covariance ellipsoid described by the covariance matrix. In the standard formulation of UT, the n dimensional ellipsoid is estimated by one sigma point at the expected value and further $2n$ sigma points selected symmetrically around it. There are other methods where more (e.g., $4n + 1$) sigma points are used to achieve a better approximation of the uncertainty [15]. These sigma points are computed via Cholesky-factorization [16] of the covariance matrix. An other approach decreases the number of sigma points to $(n+1)$ [17], [18] via an optimization step but it is computationally more expensive than the Cholesky-factorization. For a systematic overview, see [19].

The principled way of decreasing the computational cost is to avoid the unnecessary operations. This article [20] shows models where either the state or the output update is linear. In these cases, the classical uncertainty propagation of Kalman filters was used in the linear update equation while sigma

Manuscript received October 7, 2021; revised April 8, 2022; accepted May 29, 2022. This work was supported in part by the ÚNKP-20-4 New National Excellence Program of the Ministry for Innovation and Technology from the source of the National Research, Development and Innovation Fund with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2019-1.3.1-KK Funding Scheme under Project 2019-1.3.1-KK-2019-00007. The work of Péter Galambos was supported by the Bolyai Scholarship of the Hungarian Academy of Sciences and the UNKP-21-5 (Bolyai+) New National Excellence Program of the Ministry for Innovation and Technology from the source of the National Research, Development and Innovation Fund. This article was recommended by Associate Editor B. Jiang. *(Corresponding author: Péter Galambos.)*

József Kuti, Imre J. Rudas, and Péter Galambos are with the Antal Bejczy Center for Intelligent Robotics, Óbuda University, 1034 Budapest, Hungary (e-mail: jozsef.kuti@irob.uni-obuda.hu; rudas@uni-obuda.hu; peter.galambos@irob.uni-obuda.hu).

Huijun Gao is with the Research Institute of Intelligent Control and Systems, Harbin Institute of Technology, Harbin 150080, China (e-mail: hjgao@hit.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2022.3181211.

Digital Object Identifier 10.1109/TCYB.2022.3181211

points-based UT in the other one (instead of executing two separate UTs).

Previous works of Kuti and Galambos [21], [22] demonstrated that if the nonlinear function depends only on the first $m < n$ variables in a nonlinear way (n is the number of variables), the transformation can be performed with less sigma points so decreasing the computational demand of the filter. This means a significant improvement for models with a large number of variables if the function does not depend on each of them in a nonlinear way. Real-world numerical examples in [21] and [22] showed, that the cost of UT can be reduced to $\sim 60\%$ through the proposed relaxation.

This article generalizes the previously proposed method alldue to define an arbitrary subset of the variables where the UT will be performed. Furthermore, it provides an opportunity to decrease the dimension of the image of the nonlinear function during the UT according to the range of the mapping. This way, depending on the filtering problem considered, the computational cost of the entire UKF (not only the UT) can be reduced to below 40% as the provided numerical example shows.

This article is structured as follows: The next section briefly discusses the notations used in the followings. After that, Section III shortly describes the former methods which has motivated the new generalized approach described in Section IV and investigated numerically in Section V. Finally, Section VI concludes this article.

II. Notations

$a, b, \ldots$	Scalar values
$\mathbf{a}, \mathbf{b}, \ldots$	Vectors.
$\mathbf{A}, \mathbf{B}, \ldots$	Matrices.
$\mathbf{0}^{a\times b}, \mathbf{I}^{a\times b}$	Zero matrix, identity matrix of size $a \times b$.
$\mathbf{a(i)}$	Reindexed vector with $\mathbf{i}$ index vector, as $\mathbf{a(i)} = \begin{bmatrix} a_{i_1} & a_{i_2} & \cdots \end{bmatrix}^{\mathrm{T}}$.
$\mathbf{A(i}, :)$	Matrix with reindexed rows using the $\mathbf{i}$ index vector, as $\mathbf{A(i}, :) = \begin{bmatrix} A_{i_1,1} & A_{i_1,2} & \cdots \\ A_{i_2,1} & A_{i_2,2} & \cdots \\ \vdots & \vdots & \end{bmatrix}$.
$\mathbf{A}(:, \mathbf{i})$	Matrix with reindexed columns using the $\mathbf{i}$ index vector, as $\mathbf{A(i}, :) = \begin{bmatrix} A_{1,i_1} & A_{1,i_2} & \cdots \\ A_{2,i_1} & A_{2,i_2} & \cdots \\ \vdots & \vdots & \end{bmatrix}$.
$\mathbf{A(i, j)}$	Reindexed matrix with $\mathbf{i}$, $\mathbf{j}$ index vectors, as $\mathbf{A(i, j)} = \begin{bmatrix} A_{i_1,j_1} & A_{i_1,j_2} & \cdots \\ A_{i_2,j_1} & A_{i_2,j_2} & \cdots \\ \vdots & \vdots & \end{bmatrix}$.
$\sqrt{\mathbf{A}}$	Lower triangle Choleski-factorization of matrix $\mathbf{A}$ as $\mathbf{A} = \sqrt{\mathbf{A}}\sqrt{\mathbf{A}}^{\mathrm{T}}$.
$\hat{\mathbf{x}}$	Estimated value of $\mathbf{x}$.
$\tilde{\mathbf{x}}$	Difference of $\mathbf{x}$ from the expected value.
Σ_x	Estimated covariance matrix $\Sigma_x = E(\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T)$.
Σ_{xy}	Estimated cross covariance matrix $\Sigma_{xy} = E(\tilde{\mathbf{x}}\tilde{\mathbf{y}}^T)$.

III. Preliminaries

A. Unscented Transformation

The UT considers a continuous nonlinear mapping in general, as

$$\mathbf{y} = f(\mathbf{x}) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is an (approximately) Gaussian stochastic variable with covariance matrix Σ_x. The method approximates the expected value of $\mathbf{y}$, its covariance matrix Σ_y and the cross covariance matrix Σ_{xy}.

The σ points are chosen around the expected value of $\mathbf{x}$, such that the expected values and covariance matrices computed from the σ points with appropriate weighting returns exact values for second-order mappings.

In the Cholesky-factorization-based method, the $(2n + 1)$ sigma points are chosen as

$$\mathcal{X}_0 = \hat{\mathbf{x}}, \quad \mathcal{X}_i = \hat{\mathbf{x}} \pm \sqrt{(n+\lambda)\Sigma_x}, \quad i = 1, \ldots, 2n \tag{2}$$

where λ is a so-called scaling parameter that provides an extra degree of freedom to tune the method.

Then by mapping the sigma points as $\mathcal{Y}_i = f(\mathcal{X}_i)$, the expected value and the covariance matrices can be approximated as

$$\hat{\mathbf{y}} = \sum_{i=0}^{2n} W_i \mathcal{Y}_i, \qquad \Sigma_{yy} = \sum_{i=0}^{2n} W_i^{\mathrm{cov}} (\mathcal{Y}_i - \hat{\mathbf{y}})(\mathcal{Y}_i - \hat{\mathbf{y}})^T$$

$$\Sigma_{xy} = \sum_{i=1}^{2n} W_i^{\mathrm{cov}} (\mathcal{X}_i - \hat{\mathbf{x}})(\mathcal{Y}_i - \hat{\mathbf{y}})^T$$

where

$$W_i^{\mathrm{cov}} = W_i = \frac{1}{2(n+\lambda)}, \quad i = 1, \ldots, 2n$$

$$W_0^{\mathrm{cov}} = W_0 = \frac{\lambda}{n+\lambda}.$$

Earlier studies (e.g., [23]) proposed the scaling parameter to be chosen as

$$\lambda = 3 - n \tag{3}$$

based on the first terms of the Taylor expansion. However, the positive definiteness of the variance matrix is not ensured if $n \geq 4$. In these cases, $\lambda = 0$ shall be chosen.

This article [24] proposed the scaling factor to be selected as

$$\lambda = \alpha^2(\kappa + n) - n, \quad \kappa \to 0, \quad \alpha > 0$$

and a correction term is used in W_0^{cov} as

$$W_0^{\mathrm{cov}} = W_0 + 1 + \beta - \alpha^2$$

where the prior knowledge of the distribution of $\mathbf{x}$ can be taken into account via the β value, that is $\beta = 2$ for Gaussian distributions. Later, Julier [25] proposed a scaled transformation to take the higher order nonlinearities into consideration.

The systematic investigations in [26] and [27] showed that the accuracy of UT can be highly improved by tuning the scaling parameter in each step. A number of studies [27]–[30] proposed online optimization for this purpose that performs the UT much more times in each sampling step. In these methods, the computational demand of UT is crucial.

B. Unscented Transformation With Less Sigma Points

Former studies of Kuti and Galambos [21], [22] has shown, that the number of sigma points can usually be decreased by considering only the variables on which the function depends on in a nonlinear way. In order to separate the linear and nonlinear part of the mapping, the state variables are defined as

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{nl} \\ \mathbf{x}_l \end{bmatrix} \tag{4}$$

and the nonlinear mapping is described as

$$\mathbf{y} = \mathbf{A} \cdot \mathbf{x}_l + \mathbf{f}(\mathbf{x}) \tag{5}$$

where elements of $\mathbf{x}_l$ do not influence the value of $f(\mathbf{x})$.

The UT is performed only on the nonlinear mapping

$$\mathbf{b} = f(\mathbf{x}) \tag{6}$$

exploiting that the last $n - m$ variables are in the null space of the function.

Only the first m columns of the lower triangular Cholesky-factorization

$$\sqrt{\Sigma_x} = \begin{bmatrix} \Delta\mathbf{X}_{nl} & 0 \\ \Delta\mathbf{X}_l & \Delta\mathbf{M} \end{bmatrix}, \quad \Delta\mathbf{X} = \begin{bmatrix} \Delta\mathbf{X}_{nl} \\ \Delta\mathbf{X}_l \end{bmatrix} \tag{7}$$

is computed, where $\Delta\mathbf{X}_{nl} \in \mathbb{R}^{m\times m}$ and $\Delta\mathbf{X}_l \in \mathbb{R}^{(n-m)\times m}$.

Then, it is sufficient to use $2m + 1$ sigma points as

$$\mathcal{X}_0 = \hat{\mathbf{x}}, \quad \mathcal{X}_i = \hat{\mathbf{x}} \pm \sqrt{m+\lambda}\Delta\mathbf{X} \tag{8}$$

because the others would be equivalent to $\mathcal{X}_0$ after the mapping.

The sigma points must be mapped as $\mathcal{B}_i = f(\mathcal{X}_i)$, then $\hat{\mathbf{b}}$, Σ_b, and Σ_{xb} can be computed as

$$\begin{aligned} \hat{\mathbf{b}} &= \sum_{i=0}^{2m} W_i \mathcal{B}_i \\ \Sigma_b &= \sum_{i=0}^{2m} W_i^{\text{cov}} \left(\mathcal{B}_i - \hat{\mathbf{b}}\right)\left(\mathcal{B}_i - \hat{\mathbf{b}}\right)^T \\ \Sigma_{xb} &= \sum_{i=1}^{2m} W_i^{\text{cov}} \left(\mathcal{X}_i - \hat{\mathbf{x}}\right)\left(\mathcal{B}_i - \hat{\mathbf{b}}\right)^T \end{aligned} \tag{9}$$

where

$$\begin{aligned} W_i^{\text{cov}} &= W_i = \frac{1}{2(m+\lambda)}, \quad i = 1\ldots 2m \\ W_0 &= \frac{\lambda}{m+\lambda}, \quad W_0^{\text{cov}} = W_0 + \left(1 + \beta - \alpha^2\right). \end{aligned} \tag{10}$$

This way, the estimated value of the mapping is give as

$$\hat{\mathbf{y}} = \hat{\mathbf{b}} + \mathbf{A}\hat{\mathbf{x}}_l. \tag{11}$$

Furthermore, the covariance matrices can be computed as

$$\begin{aligned} \Sigma_y &= \Sigma_b + \mathbf{A}\Sigma_{x_l}\mathbf{A}^T + \text{Sym}\left(\Sigma_{bx_l}\mathbf{A}^T\right) \\ \Sigma_{xy} &= \Sigma_{xb} + \Sigma_{xx_l}\mathbf{A}^T \end{aligned} \tag{12}$$

where $\Sigma_x = \left[\Sigma_{xx_{nl}}\ \Sigma_{xx_l}\right]$ and $\Sigma_{xx_l} = \begin{bmatrix} \Sigma_{x_{nl}x_l} \\ \Sigma_{x_l} \end{bmatrix}$.

C. Time Update in UKF

If the standard UT is applied, the system model is usually defined as

$$\mathbf{x}_k = f_k(\mathbf{x}_{k-1}) + \mathbf{w}_k \tag{13}$$

$$\mathbf{y}_k = g_k(\mathbf{x}_k) + \mathbf{v}_k \tag{14}$$

or it can depend on the disturbance and noise signals in a nonlinear way and the $\mathbf{x}_0$ value is considered an initial value.

For the relaxed UT, the linear dependencies are pulled out as

$$\mathbf{x}_k = \mathbf{A}_k\mathbf{x}_{k-1,l} + f_k(\mathbf{x}_{k-1}) + \mathbf{w}_k \tag{15}$$

$$\mathbf{y}_k = \mathbf{C}_k\mathbf{x}_{k,l} + g_k(\mathbf{x}_k) + \mathbf{v}_k. \tag{16}$$

It must be mentioned here, that there are two ways to apply the UT on a system model.

1) The naive way is to use the UT with the Cholesky-factorization twice.
   a) First, to determine $\hat{\mathbf{x}}_k$, $\Sigma_{xx,k}$ based on $\hat{\mathbf{x}}_{k-1}$, $\Sigma_{xx,k-1}$, $\hat{\mathbf{w}}_k$, and $\Sigma_{ww,k}$.
   b) And again to compute $\hat{\mathbf{y}}_k$, $\Sigma_{yy,k}$, $\Sigma_{yx,k}$ based on $\hat{\mathbf{x}}_k$, $\Sigma_{xx,k}$, $\hat{\mathbf{v}}_k$, and $\Sigma_{vv,k}$.
2) Or by using the mapped sigma points of the first step as sigma points $\mathcal{X}_i$ of the second step in order to spare one Cholesky-factorization. In this case, the $\mathbf{x}_{nl}$ variable set must be the same in (15) and (16).

D. Adaptive UKF

This article [28] showed the importance of the scaling factor (λ), and Straka *et al.* [27], Dunik *et al.* [28], Scardua and da Cruz [29], and Bahraini *et al.* [30] proposed optimization methods to adapt its value according to the current measurements. The methods represent a tradeoff between accuracy and computational cost, so the capabilities of the available hardware are a key constrain.

For example, the minimizing normalized measurement prediction error squared (MNMPES) method of [27] defines the scaling parameter to be applied as

$$\lambda^{*\text{MNMPES}} = \arg\min_{\lambda}\left[\tilde{\mathbf{z}}(\lambda)^T\Sigma_{zz}(\lambda)^{-1}\tilde{\mathbf{z}}(\lambda)\right] \tag{17}$$

where $\tilde{\mathbf{z}}(\lambda) = \hat{\mathbf{z}}(\lambda) - \mathbf{z}_{\text{measured}}$ and the λ argument denotes a'priori estimations derived with λ scaling parameter. The minimization can be a grid search where the grid density must be set according to the capacities of the computing hardware.

IV. Generalized Relaxed Unscented Transformation

The section first discusses the key pillars of decreasing the computational demand, then the generalized algorithm is constructed that unifies the building blocks.

A. Less Sigma Points by Considering Only Subset of Variables

As a generalization of the idea discussed in [21], assume that the mapping is given as

$$\mathbf{y} = \mathbf{A} \cdot \mathbf{x}(\mathbf{i}_l) + \mathbf{f}(\mathbf{x})$$

where the indices of variables with linear dependence are collected into vector $\mathbf{i}_l$ and the nonlinear ones into $\mathbf{i}_{nl}$.

In this case, the method of Section III-B can be used as the followings.

1) Denote the result of the nonlinear part as $\mathbf{b} = \mathbf{f}(\mathbf{x})$. To derive such a situation as in Section III-B, the Choleski-factorization of rearranged Σ_x must be computed as

$$\Delta\mathbf{A} = \sqrt{\Sigma_x\big(\big[\mathbf{i}_{nl}\ \bar{\mathbf{i}}_{nl}\big], \big[\mathbf{i}_{nl}\ \bar{\mathbf{i}}_{nl}\big]\big)} \tag{18}$$

and then $\Delta\mathbf{X}$ is constructed from its first m columns, by rearranging the rows as

$$\Delta\mathbf{X}\big(\big[\mathbf{i}_{nl}\ \bar{\mathbf{i}}_{nl}\big], :\big) = \Delta\mathbf{A}\big(:, \big[1\ \ldots\ m\big]\big) \tag{19}$$

where index vector $\bar{\mathbf{i}}_{nl}$ is constructed from the complementer set of indices in $\mathbf{i}_{nl}$.

2) Then, the reduced number of sigma points $\mathcal{X}_0, \ldots, \mathcal{X}_{2m}$ can be computed as in (8) using $\Delta\mathbf{X}$ derived in the latest step.

3) Then $\hat{\mathbf{b}}$, Σ_b, and Σ_{xb} can be derived from these sigma points as in (9).

4) Then $\hat{\mathbf{y}}$, Σ_y, and Σ_{xy} can be computed as

$$\begin{aligned}\hat{\mathbf{y}} &= \mathbf{A}\cdot\hat{\mathbf{x}}(\mathbf{i}_l) + \hat{\mathbf{b}}\\ \Sigma_y &= \Sigma_b + \mathbf{A}\Sigma_x(\mathbf{i}_l, \mathbf{i}_l)\mathbf{A}^T + \mathrm{Sym}(\mathbf{A}\Sigma_{xb}(\mathbf{i}_l, :))\\ \Sigma_{xy} &= \Sigma_{xb} + \Sigma_x(:, \mathbf{i}_l)\mathbf{A}^T.\end{aligned} \tag{20}$$

B. Less Sigma Points by Considering Subspace of the Domain

Consider again the mapping in the following form:

$$\mathbf{y} = \mathbf{A}\cdot\mathbf{x}(\mathbf{i}_l) + \mathbf{f}(\mathbf{x})$$

where the indices of variables with linear dependence are collected into vector $\mathbf{i}_l$ and the nonlinear part depends on the linear combinations of the variables. Describe these values as $(\mathbf{m}_1{\cdot}\mathbf{x}(\mathbf{i}_1))$, $(\mathbf{m}_2{\cdot}\mathbf{x}(\mathbf{i}_2)), \ldots, (\mathbf{m}_K{\cdot}\mathbf{x}(\mathbf{i}_K))$, where $\mathbf{m}_k$ contains the weights for values with indices $\mathbf{i}_k$ and K denotes the number of combinations.

In this case, the function depends on values of $\mathbf{M}\mathbf{x}$, where

$$\mathbf{M}(k, \mathbf{i}_k) = \mathbf{m}_k, \quad k = 1, \ldots, K$$

denote its rank by m.

Example 1: Considering a mapping, for example

$$f(\mathbf{x}) = \sin(x_1 + 0.1x_3) - \cos(0.5x_2 + x_3)$$

two linear combinations of the variables can be seen in nonlinear functions. In the first case, the index vector is $\mathbf{i}_1 = \begin{bmatrix}1 & 3\end{bmatrix}$ and the corresponding weight vector is $\mathbf{m}_1 = \begin{bmatrix}1 & 0.1\end{bmatrix}$. In the second linear combination, the index vector is $\mathbf{i}_2 = \begin{bmatrix}2 & 3\end{bmatrix}$ and the weight vector is $\mathbf{m}_2 = \begin{bmatrix}0.5 & 1\end{bmatrix}$. This way, the matrix $\mathbf{M}$ can be written in this case as

$$\mathbf{M} = \begin{bmatrix}1 & 0 & 0.1\\ 0 & 0.5 & 1\end{bmatrix}.$$

Denote the matrix constructed from an orthonormal basis of the rowspace of $\mathbf{M}$ by $\mathbf{Q}_1 \in \mathbb{R}^{m\times n}$ and the matrix constructed from an orthonormal basis of nullspace of $\mathbf{M}$ by $\mathbf{Q}_2 \in \mathbb{R}^{(n-m)\times n}$. The matrix $\mathbf{Q}$ is constructed of them as

$$\mathbf{Q} = \begin{bmatrix}\mathbf{Q}_1\\ \mathbf{Q}_2\end{bmatrix} \tag{21}$$

and can be easily computed as RQ factorization of $\mathbf{M}$.

In this case, the method of Section III-B can be used as the followings.

1) The Choleski-factorization of rearranged Σ_x must be computed as

$$\Delta\mathbf{A} = \sqrt{\mathbf{Q}\Sigma_x\mathbf{Q}^T} \tag{22}$$

and the $\Delta\mathbf{X}$ is constructed from its first m columns, by rearranging the rows as

$$\Delta\mathbf{X} = \mathbf{Q}^T\Delta\mathbf{A}\big(:, \big[1\ \ldots\ m\big]\big). \tag{23}$$

2) Then, the reduced number of sigma points $\mathcal{X}_0, \ldots, \mathcal{X}_{2m}$ can be computed as in (8) using $\Delta\mathbf{X}$ derived in the latest step.

3) Then $\hat{\mathbf{b}}$, Σ_b, and Σ_{xb} can be derived from these sigma points as in (9).

4) Then $\hat{\mathbf{y}}$, Σ_y, and Σ_{xy} can be computed as in (20).

To minimize the matrix $\mathbf{Q}$ related computations, it is proposed to be chosen to as sparse as possible.

C. Reduced Size Output of the Nonlinear Function

For the sake of simplicity, consider a purely nonlinear mapping $\mathbb{R}^n \to \mathbb{R}^a$ denoted as $\mathbf{y} = f(\mathbf{x})$. Denote the dimension of the image of f by $b \leq a$.

In this case, the function can be written as

$$\mathbf{y} = \mathbf{b}(\mathbf{g}), \quad \mathbf{b} = \begin{bmatrix}\mathbf{b}_0\\ \mathbf{F}\cdot\mathbf{b}_0\end{bmatrix}, \quad \mathbf{b}_0 = f_0(\mathbf{x}) \tag{24}$$

where $\mathbf{g} \in \mathbb{N}^b$ is an index vector, $\mathbf{F} \in \mathbb{R}^{a-b\times b}$ is a coefficient matrix and $f_0 : \mathbb{R}^n \to \mathbb{R}^b$.

In this case, the computational cost of UT can be decreased via the following method.

1) Perform the UT as in Section III-A on function f_0 to compute $\hat{\mathbf{b}}_0$, Σ_{b_0}, and Σ_{xb_0}.

2) Compute the $\mathbf{b}$ related quantities as

$$\begin{aligned}\hat{\mathbf{b}} &= \begin{bmatrix}\hat{\mathbf{b}}_0\\ \mathbf{F}\hat{\mathbf{b}}_0\end{bmatrix}, \qquad \Sigma_{xb} = \begin{bmatrix}\Sigma_{xb_0} & \Sigma_{xb_0}\mathbf{F}^T\end{bmatrix}\\ \Sigma_b &= \begin{bmatrix}\Sigma_{b_0} & \Sigma_{b_0}\mathbf{F}^T\\ \mathbf{F}\Sigma_{b_0} & \mathbf{F}\Sigma_{b_0}\mathbf{F}^T\end{bmatrix}.\end{aligned} \tag{25}$$

3) Perform reindexing with $\mathbf{g}$ if it is needed

$$\hat{\mathbf{y}} = \hat{\mathbf{b}}(\mathbf{g}), \quad \Sigma_y = \Sigma_b(\mathbf{g}, \mathbf{g}), \quad \Sigma_{xy} = \Sigma_{xb}(:, \mathbf{g}). \tag{26}$$

D. New Relaxed UT Algorithm

The proposed algorithm is based on Section IV-A and output reduction of Section IV-C and optionally allows to use the method of Section IV-B and reindexing of Section IV-C because of their higher computational demand.

The mapping in this case must be defined as

$$\mathbf{y} = \mathbf{A}\cdot\mathbf{x}(\mathbf{i}_l) + \begin{cases}\mathbf{b}(\mathbf{g}), & \text{if reindexing is used}\\ \mathbf{b}, & \text{otherwise}\end{cases} \tag{27}$$

where

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{F} \cdot \mathbf{b}_0 \end{bmatrix}, \quad \mathbf{b}_0 = f_0(\mathbf{x})$$

and the f_0 function depends on the variables with indices $\mathbf{i}_{nl}$, furthermore, if exact subspace is considered denote the combinations $k = 1, \ldots, K$ of variables with indices $\mathbf{i}_k$ and weights $\mathbf{m}_k$ of the other nonlinear dependencies.

1) Initialization: If exact subspace is considered then RQ factorization-based offline computation is needed. The matrix $\mathbf{M}$ matrix must be constructed as

$$\underbrace{\begin{bmatrix} \mathbf{I}^{n\times n}(\mathbf{i}_{nl}, :) \\ \mathbf{M}' \end{bmatrix}}_{\mathbf{M}} \mathbf{x}$$

where $\mathbf{M}' \in \mathbb{R}^{K\times n}$, $\mathbf{M}'(k, \mathbf{i}_k) = \mathbf{m}_k$ for all $k = 1, \ldots, K$ and the other values are zeros. Then, m is the rank of matrix $\mathbf{M}$ and the matrix $\mathbf{Q}$ can be computed from its RQ factorization.

Example 2: For example, if the mapping is

$$f(\mathbf{x}) = \sin(x_1 + 0.1x_3) - \cos(0.5x_1) + x_3$$

the standalone nonlinear dependencies are $\mathbf{i}_{nl} = \begin{bmatrix}1\end{bmatrix}$ and the index vector of the linear combination is $\mathbf{i}_1 = \begin{bmatrix}1 & 3\end{bmatrix}$ and the corresponding weight vector is $\mathbf{m}_1 = \begin{bmatrix}1 & 0.1\end{bmatrix}$. The $\mathbf{M}'$ matrix contains these values as

$$\mathbf{M}' = \begin{bmatrix}1 & 0 & 0.1\end{bmatrix}.$$

2) Online Computation: The online computation of $\mathbf{y}$ related quantities from $\hat{\mathbf{x}}$ and Σ_x is as the followings.

1) Compute $\Delta\mathbf{X}$ for sigma points as in (22) and (23) if exact subspace is considered, otherwise as in (18) and (19).
2) Compute the sigma points as in (8).
3) Compute $\mathbf{b}_0$ related properties as in (9).
4) Compute $\mathbf{b}$ related values as in (25).
5) If reindexing is used, compute $\mathbf{y}$ related values as

$$\begin{aligned} \hat{\mathbf{y}} &= \hat{\mathbf{b}}(\mathbf{g}) + \mathbf{A}\hat{\mathbf{x}}(\mathbf{i}_l) \\ \Sigma_y &= \Sigma_b(\mathbf{g}, \mathbf{g}) + \mathbf{A}\Sigma_x(\mathbf{i}_l, \mathbf{i}_l)\mathbf{A}^T + \mathrm{Sym}(\mathbf{A}\Sigma_{xb}(\mathbf{i}_l, \mathbf{g})) \\ \Sigma_{xy} &= \Sigma_{xb}(:, \mathbf{g}) + \Sigma_x(:, \mathbf{i}_l)\mathbf{A}^T \end{aligned} \tag{28}$$

otherwise as in (20).

Remark 1: Although the number of sigma points can be decreased by considering the exact subspace, it needs an initialization with RQ factorization. If the method must be always reinitialized for the current configuration, the cost of the factorization can overwhelm the benefit of sparing sigma points. In this case, describing these nonlinear dependencies via the index vector $\mathbf{i}_{nl}$ can be better.

V. Numerical Example

In this section, the accuracy and the computational benefits of the proposed approach is investigated and compared to the standard UT-based method. Although the latest researches in the field of SLAM consider problems like multiagent SLAM triggered by loop closures [31], here the centralized SLAM model is considered as a representative example (as in [30]). The example demonstrates how this well-known model is rewritten into the form of (27) and how the proposed method performs according to the complexity of the model, that is determined by the number of registered and currently seen landmarks in these models.

The examples were implemented in modern C++ using the Eigen3 [32] library. The experiments were executed on MS Windows 10 OS and Intel i7 9750 2.6-GHz processor.

A. Modeling

The kinematic model of a planar mobile robot with differential drive is described as follows:

$$\begin{bmatrix} x_k \\ y_k \\ \varphi_k \end{bmatrix} = \begin{bmatrix} x_{k-1} + v_k\cos(\varphi_{k-1})dT_k \\ y_{k-1} + v_k\sin(\varphi_{k-1})dT_k \\ \varphi_{k-1} + \omega_k dT_k \end{bmatrix} \tag{29}$$

where x_k, y_k, and φ_k denote the position and the orientation of the robot at the kth iteration, v_k and ω_k are the linear and angular velocity, and dT_k is the elapsed time since the latest iteration.

Stationary landmarks are considered, so the model of the (i)th landmark ($i = 1, \ldots, N$, where N denotes the number of already registered landmarks) is

$$\begin{bmatrix} x_k^{(i)} \\ x_k^{(i)} \end{bmatrix} = \begin{bmatrix} x_{k-1}^{(i)} \\ x_{k-1}^{(i)} \end{bmatrix}. \tag{30}$$

The output (distance r and angle θ) of laser range finder of the (i)th landmark can be described as follows:

$$z_k^{(i)} = \begin{bmatrix} r_k^{(i)} \\ \theta_k^{(i)} \end{bmatrix} = \begin{bmatrix} \sqrt{\left(x_k^{(i)} - x_k\right)^2 + \left(y_k^{(i)} - y_k\right)^2} + v_{r_k} \\ \arctan\left(\frac{y_k^{(i)} - y_k}{x_k^{(i)} - x_k}\right) - \varphi_k + v_{\theta_k} \end{bmatrix} \tag{31}$$

where v_r and v_θ denote measurement noises.

B. UT of the State Update Function

The nonlinear model can be written as

$$\underbrace{\begin{bmatrix} x_k \\ y_k \\ \varphi_k \\ x_k^{(1)} \\ y_k^{(1)} \\ \vdots \\ x_k^{(N)} \\ y_k^{(N)} \end{bmatrix}}_{=\mathbf{x}_k} = \underbrace{\begin{bmatrix} x_{k-1} + v_k\cos(\varphi_{k-1})dT_k \\ y_{k-1} + v_k\sin(\varphi_{k-1})dT_k \\ \varphi_{k-1} + \omega_k dT_k \\ x_{k-1}^{(1)} \\ x_{k-1}^{(1)} \\ \vdots \\ x_{k-1}^{(N)} \\ x_{k-1}^{(N)} \end{bmatrix}}_{=f\left(\begin{bmatrix} v_k \\ \omega_k \\ \mathbf{x}_{k-1} \end{bmatrix}\right)} \tag{32}$$

where the odometry data and the latest state constitutes the input vector

$$\begin{bmatrix} v_k \\ \omega_k \\ \mathbf{x}_{k-1} \end{bmatrix} \in \mathbb{R}^{2N+5} \tag{33}$$

the new state vector is the resulted output and dT_k is an exactly known parameter.

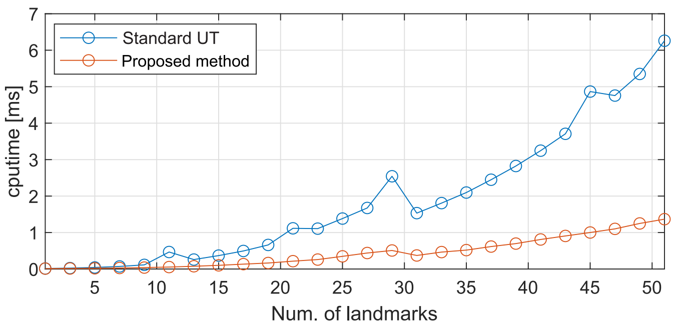

Fig. 1. CPU time of SLAM state update computation as function of number of registered landmarks (N).

In this case, the state update law for N landmarks can be written as

$$\mathbf{x}_k = \mathbf{b} + \mathbf{A} \cdot \mathbf{x}(\mathbf{i}_l) \tag{34}$$

where the core nonlinear function $\mathbf{b} = f_0(\mathbf{x})$ is

$$f_0(\mathbf{x}) = \begin{bmatrix} v_k \cos(\varphi_{k-1})dT_k \\ v_k \sin(\varphi_{k-1})dT_k \\ \varphi_{k-1} \end{bmatrix} \tag{35}$$

the nonlinear variables are v_k and φ_{k-1}, so the corresponding indices are $\mathbf{i}_{nl} = \begin{bmatrix} 1 & 5 \end{bmatrix}$, and there is not need of mixed groups. The matrix $\mathbf{F}$ contains only zeros in this case $\mathbf{F} = \mathbf{0}^{2N+2\times 3}$. The linear indices are the other ones, so the index vector is $\mathbf{i}_l = \begin{bmatrix} 2 & 3 & 4 & 6 & 7 & \cdots & 2N+5 \end{bmatrix}$ and the coefficient matrix can be written as

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & \\ 0 & 0 & 1 & \mathbf{0}^{3\times 2N} \\ dT_k & 0 & 0 & \\ & \mathbf{0}^{2N\times 3} & & \mathbf{I}^{2N\times 2N} \end{bmatrix}. \tag{36}$$

Because the nonlinear part depends only on two variables, 5 sigma point is enough in the new UT but the standard UT uses $2N+1$ sigma points.

Fig. 1 shows the average time needed for computing one state update with a given number of registered landmarks. The preemptibility settings of the OS cause a few outlying peaks but the trend is clear: the standard UT shows third-order dependency on the number of landmarks but using the proposed method, the linear component is dominant. If the number of landmarks is 51, the proposed method does need only 20.6% of the CPU time of the standard method.

It shows well the effectiveness of the method for the case if the nonlinear part of the mapping is quite small.

C. UT of the Output Update Function

Assume that from the N registered landmarks the ones with indices $\mathbf{h} \in \mathbb{N}^{N_{\text{active}}}$ was detected in the latest sampling time. Then, the output of the system is

$$\mathbf{z}_k = \begin{bmatrix} r_k^{(h_1)} & \theta_k^{(h_1)} & \dots & r_k^{(h_{N_\text{active}})} & \theta_k^{(h_{N_\text{active}})} \end{bmatrix}^T.$$

The output characteristics depends on φ_k in a linear way, so $\mathbf{i}_l = 3$ and the coefficient matrix is

$$\mathbf{A} = \begin{bmatrix} 0 & -1 & 0 & -1 & \cdots \end{bmatrix}^T \in \mathbb{R}^{2N_\text{active}}. \tag{37}$$

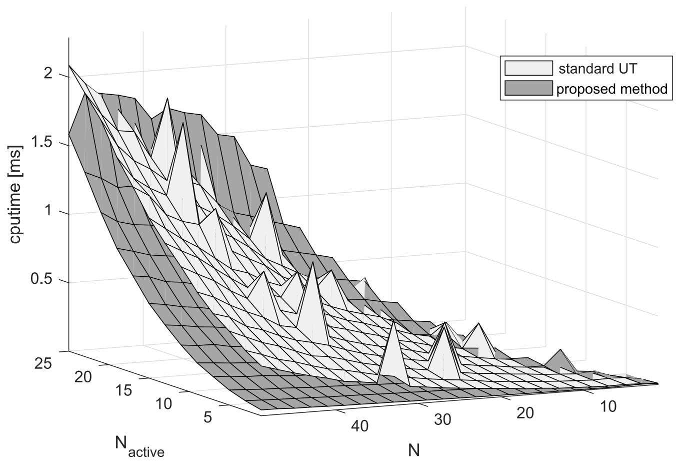

Fig. 2. CPU time of SLAM output update computation as function of number of registered landmarks (N) and currently seen landmarks (N_{active}).

The nonlinear mapping is

$$f_0(\mathbf{x}) = \begin{bmatrix} \sqrt{\left(x_k^{(h_1)} - x_k\right)^2 + \left(y_k^{(h_1)} - y_k\right)^2} \\ \operatorname{atan2}\left(y_k^{(h_1)} - y_k, x_k^{(h_1)} - x_k\right) \\ \sqrt{\left(x_k^{(h_2)} - x_k\right)^2 + \left(y_k^{(h_2)} - y_k\right)^2} \\ \operatorname{atan2}\left(y_k^{(h_2)} - y_k, x_k^{(h_2)} - x_k\right) \\ \vdots \end{bmatrix} \tag{38}$$

that depends on linear combinations $(x_k^{(i)} - x_k)$, $(y_k^{(i)} - y_k)$ for all $i \in \mathbf{h}$ in a nonlinear way.

If exact subspace is applied with initialization, the settings of the method is applied, he variables of the mixed groups are $\mathbf{i}_{2n} = \begin{bmatrix} 1 & 1+2h_n \end{bmatrix}$, $\mathbf{m}_{2n} = \begin{bmatrix} -1 & 1 \end{bmatrix}$ and $\mathbf{i}_{2n+1} = \begin{bmatrix} 2 & 2+2h_n \end{bmatrix}$, $\mathbf{m}_{2n+1} = \begin{bmatrix} -1 & 1 \end{bmatrix}$ for $n = 1, \dots, N_\text{active}$. Reindexing with $\mathbf{g}$ is not needed, $\mathbf{F}$ is empty as $\mathbf{F} \in \mathbb{R}^{0\times 2N_\text{active}}$. In this case, the proposed method uses only $4N_\text{active}+1$ sigma points (the standard UT method applies $4N+7$ ones), but the RQ factorization of the initialization is needed for the different measurement configurations described by $\mathbf{h}$.

Remark 2: The image of $\operatorname{atan2}(y, x)$ function is $(-\pi, \pi]$ and it is not continuous, that can causes problems if one or more sigma points are mapped to $\sim \pi$ and the other ones to $\sim -\pi$. By offsetting the values with 2π appropriately, the shape of the uncertainty ellipsoid must be restored before substituting their values into (9).

The output characteristics (38) is a quite nonlinear formula. The computational complexity depends on the number of registered landmarks (N) and on the number of seen landmarks in the given moment (N_active). The results are presented in Fig. 2. As the number of registered landmarks is increasing and the number of currently seen landmarks is small, the proposed method has huge benefits considering the standard UT. As the number of active landmarks is increased the computational complexity of the proposed method can be larger than the standard UT.

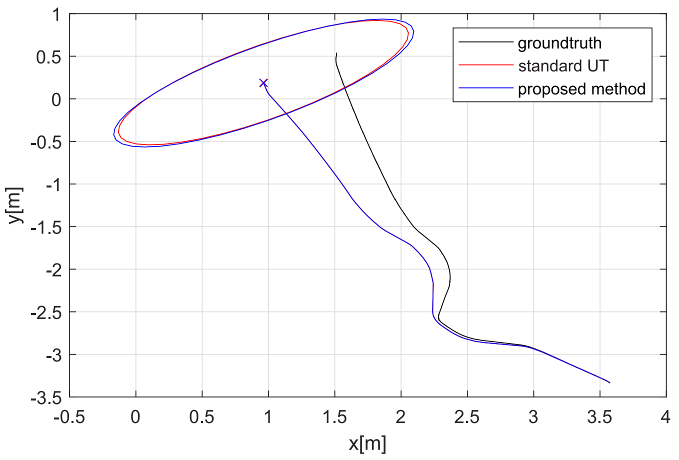

Fig. 3. Odometry-based prediction of the path via standard UT and the proposed method.

This case shows well, that if the considered mapping depends on all of the variables in a nonlinear way, the standard UT is a better choice, but in other cases, the proposed method can crucially decrease the computational demand.

If exact subspace is not used, the indices of nonlinear dependencies

$$\mathbf{i}_{nl} = \begin{bmatrix} 1 & 2 & 2+2h_1 & 3+2h_1 & 2+2h_2 & 3+2h_2 \\ & & \cdots & 2+2h_{N_{\text{active}}} & 3+2h_{N_{\text{active}}} \end{bmatrix}.$$

Reindexing with $\mathbf{g}$ is not needed, $\mathbf{F}$ is empty as $\mathbf{F} \in \mathbb{R}^{0\times 2N_{\text{active}}}$. In this case, the proposed method uses only $4N_{\text{active}}+3$ sigma points (the standard UT method applies $4N+7$ ones). For practical reasons, this setting will be used in the following simulations.

D. Numerical Simulations

There is always some difference in the approximation of the proposed and the standard UT method. To check its effect to the performance, and the difference of computational times in practical scenarios, the UTIAS multirobot cooperative localization and mapping (MRCLAM) dataset [33] is considered. The first 80[s] of the trajectory of robot *n*. 1 contains 4220 odometry input and registers 11 landmarks. During the motion 142 measurements of the LRF sensor are logged, $1, \ldots, 5$ landmarks are seen at the same time.

First, the only odometry-based prediction was performed using the standard UT and the proposed method without registering the landmarks. The results are shown in Fig. 3.

In this case, the differences between the standard UT and the proposed method are quite small. The computational demand of the simulation based on the standard UT is much smaller (25[ms] against 71[ms]), because the model is almost totally nonlinear in this case.

In the next use case, the landmarks are registered after their first detection and are considered during the prediction but filtering is not performed. This case shows the benefit of the new method. It reduces the computational time from 234[ms] to 81[ms]. Fig. 4 shows how accurately the results of the

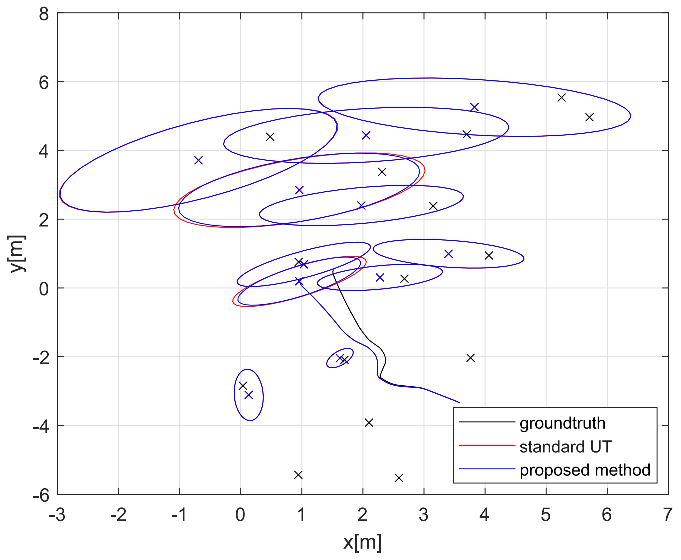

Fig. 4. Odometry-based prediction of the path and initialization of landmark poses based on the first measured values via standard UT and the proposed method.

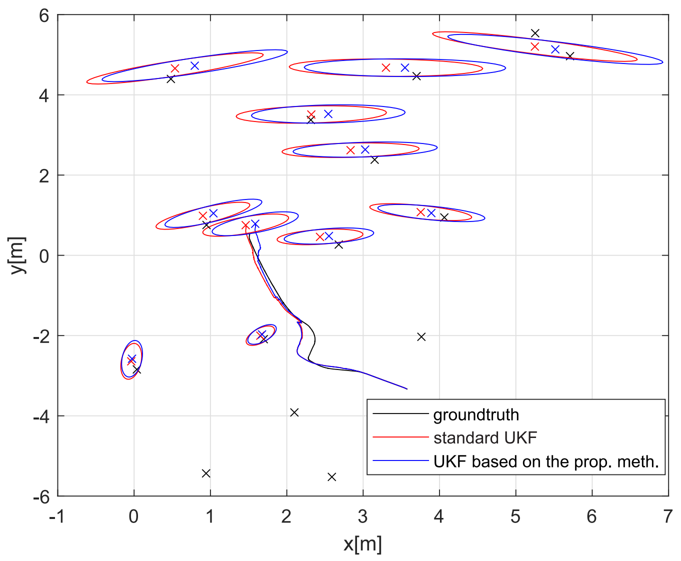

Fig. 5. Result of unscented Kalman filtering based on standard UT and the proposed method.

standard UT and the proposed method cover each other. It is interesting that the standard UT results in a slightly different robot pose if the landmarks are taken into account in the model, but the proposed method results in exactly the same results as in Fig. 3.

The results of Kalman filtering are depicted in Fig. 5. In this case, larger differences appear in the results of the methods, but the computational demands do not change essentially. With the standard UT, it needs 236[ms], with the proposed method it is 84[ms]. The odometry is updated much more times than measurement received, so the state update is much more dominant than output computation and Kalman filtering.

Finally, the AUKF method of [27] with criteria MNMPES was applied, and the scaling factor was optimized on $\lambda \in$

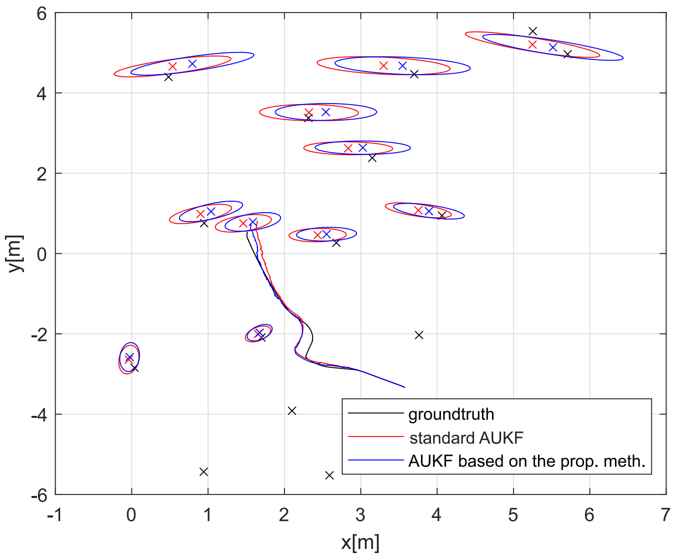

Fig. 6. Result of adaptive unscented Kalman filtering MNMPES of [27] based on standard UT and the proposed method.

TABLE I
CPU Times of Simulations on MRCLAM 1 Datasets Based on the Standard UT and the Proposed Method in Milliseconds

	prediction (only robot)	prediction (with landmarks)	UKF	AUKF [27]
standard UT	25	234	236	490
proposed method	71	81	84	194

$[0, 50]$ in 100 gridpoints. The result is shown in Fig. 6: the adaptive method decreases the uncertainty of the estimations compared to the results in Fig. 5. In this case, the computational time of the filtering using the standard UT was increased to 490[ms], but with the proposed method it is only 194[ms].

Remark 3: In computation of $\tilde{\mathbf{z}}(\lambda)$ in (17), it must be ensured, that the computed difference of variables $\hat{\theta}(\lambda), \theta_{\text{measured}} \in (-\pi, \pi]$ will be in the domain $(-\pi, \pi]$ via an appropriate offset.

The computational times of the simulations are collected in Table I. If there is not a large linear part in the mapping the standard UT performs better (see the prediction without landmark positions) but in the other cases, the relaxed UT method decreases the computational demand to 34%–40%.

VI. Conclusion

A systematic technique was introduced to largely increase the computational efficiency of the filtering methods that uses the UT. The presented algorithm exploits the partial linearity of the dynamic models and reducing the dimensions on which the Cholesky-factorization have to be performed through the UT. Along with detailed derivation, the proposed relaxation were elaborated and validated via real-world numerical examples showing that the computational demand can be decreased as much as 60% without influencing the accuracy. In practice, implementing the proposed approach is generally more demanding in terms of mathematical investigation efforts than the standard UT. Furthermore, the increased complexity of reindexing and subspace transformation, as two optional elements of the approach, may not always decrease the computational cost, see Remark 1. An open-source C++ implementation of the methods is available in our public repository [34].

References

[1] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, “A new approach for filtering nonlinear systems,” in *Proc. Amer. Control Conf.*, vol. 3, 1995, pp. 1628–1632.

[2] S. J. Julier and J. K. Uhlmann, “Unscented filtering and nonlinear estimation,” *Proc. IEEE*, vol. 92, no. 3, pp. 401–422, Mar. 2004.

[3] L. Li, D. Yu, Y. Xia, and H. Yang, “Remote nonlinear state estimation with stochastic event-triggered sensor schedule,” *IEEE Trans. Cybern.*, vol. 49, no. 3, pp. 734–745, Mar. 2019.

[4] M. C. Fowler, T. C. Clancy, and R. K. Williams, “Intelligent knowledge distribution: Constrained-action POMDPs for resource-aware Multiagent communication,” *IEEE Trans. Cybern.*, vol. 52, no. 4, pp. 2004–2017, Apr. 2022.

[5] W. Li, G. Wei, F. Han, and Y. Liu, “Weighted average consensus-based unscented Kalman filtering,” *IEEE Trans. Cybern.*, vol. 46, no. 2, pp. 558–567, Feb. 2016.

[6] Y. Zhang, B. Chen, L. Yu, and D. W. C. Ho, “Distributed Kalman filtering for interconnected dynamic systems,” *IEEE Trans. Cybern.*, early access, Jun. 16, 2021, doi: 10.1109/TCYB.2021.3072198.

[7] B. Lian, Y. Wan, Y. Zhang, M. Liu, F. L. Lewis, and T. Chai, “Distributed Kalman consensus filter for estimation with moving targets,” *IEEE Trans. Cybern.*, early access, Nov. 11, 2020, doi: 10.1109/TCYB.2020.3029007.

[8] P. Duan, Z. Duan, Y. Lv, and G. Chen, “Distributed finite-horizon extended Kalman filtering for uncertain nonlinear systems,” *IEEE Trans. Cybern.*, vol. 51, no. 2, pp. 512–520, Feb. 2021.

[9] B. Lian, F. L. Lewis, G. A. Hewer, K. Estabridis, and T. Chai, “Robustness analysis of distributed Kalman filter for estimation in sensor networks,” *IEEE Trans. Cybern.*, early access, Jun. 18, 2021, doi: 10.1109/TCYB.2021.3082157.

[10] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges,” in *Proc. IJCAI*, 2003, pp. 1151–1156.

[11] C. Kim, R. Sakthivel, and W. K. Chung, “Unscented FastSLAM: A robust and efficient solution to the SLAM problem,” *IEEE Trans. Robot.*, vol. 24, no. 4, pp. 808–820, Aug. 2008.

[12] Z. Kurt-Yavuz and S. Yavuz, “A comparison of EKF, UKF, FastSLAM2. 0, and UKF-based FastSLAM algorithms,” in *Proc. IEEE 16th Int. Conf. Intell. Eng. Syst. (INES)*, 2012, pp. 37–43.

[13] A. Giannitrapani, N. Ceccarelli, F. Scortecci, and A. Garulli, “Comparison of EKF and UKF for spacecraft localization via angle measurements,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 47, no. 1, pp. 75–84, Jan. 2011.

[14] S. Konatowski, P. Kaniewski, and J. Matuszewski, “Comparison of estimation accuracy of EKF, UKF and PF filters,” *Annu. Navig.*, vol. 23, no. 1, pp. 69–87, 2016.

[15] R. Radhakrishnan, A. Yadav, P. Date, and S. Bhaumik, “A new method for generating sigma points and weights for nonlinear filtering,” *IEEE Control Syst. Lett.*, vol. 2, no. 3, pp. 519–524, Jul. 2018.

[16] N. J. Higham, “Cholesky factorization,” *Wiley Interdiscipl. Rev. Comput. Stat.*, vol. 1, no. 2, pp. 251–254, 2009.

[17] S. J. Julier, “The spherical simplex unscented transformation,” in *Proc. Amer. Control Conf.*, vol. 3, 2003, pp. 2430–2434.

[18] S. J. Julier and J. K. Uhlmann, “Reduced sigma point filters for the propagation of means and covariances through nonlinear transformations,” in *Proc. Amer. Control Conf.*, vol. 2, 2002, pp. 887–892.

[19] H. M. T. Menegaz, J. Y. Ishihara, G. A. Borges, and A. N. Vargas, “A systematization of the unscented Kalman filter theory,” *IEEE Trans. Autom. Control*, vol. 60, no. 10, pp. 2583–2598, Oct. 2015.

[20] M. Briers, S. R. Maskell, and R. Wright, “A Rao-Blackwellised unscented Kalman filter,” in *Proc. 6th Int. Conf. Inf. Fusion (ISIF)*, 2003, pp. 55–61.

[21] J. Kuti and P. Galambos, "Decreasing the computational demand of unscented Kalman filter based methods," in *Proc. IEEE 15th Int. Symp. Appl. Comput. Intell. Inform. (SACI)*, 2021, pp. 181–186.

[22] J. Kuti and P. Galambos, "Computational analysis of relaxed unscented transformation in terms of necessary floating point operations," in *Proc. IEEE 25th IEEE Int. Conf. Intell. Eng. Syst. (INES)*, 2021, pp. 55–60.

[23] S. Julier, J. Uhlmann, and H. F. Durrant-Whyte, "Technical notes and correspondence," *IEEE Trans. Autom. Control*, vol. 45, no. 3, p. 477, Mar. 2000.

[24] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proc. IEEE Adapt. Syst. Signal Process. Commun. Control Symp.*, 2000, pp. 153–158.

[25] S. J. Julier, "The scaled unscented transformation," in *Proc. Amer. Control Conf.*, vol. 6, 2002, pp. 4555–4559.

[26] O. Straka, J. Dunik, and M. Simandl, "Scaling parameter in unscented transform: Analysis and specification," in *Proc. Amer. Control Conf. (ACC)*, 2012, pp. 5550–5555.

[27] O. Straka, J. Duník, and M. Šimandl, "Unscented Kalman filter with advanced adaptation of scaling parameter," *Automatica*, vol. 50, no. 10, pp. 2657–2664, 2014.

[28] J. Dunik, M. Simandl, and O. Straka, "Unscented Kalman filter: Aspects and adaptive setting of scaling parameter," *IEEE Trans. Autom. Control*, vol. 57, no. 9, pp. 2411–2416, Sep. 2012.

[29] L. A. Scardua and J. J. da Cruz, "Adaptively tuning the scaling parameter of the unscented Kalman filter," in *Proc. 11th Portuguese Conf. Autom. Control*, 2015, pp. 429–438.

[30] M. S. Bahraini, M. Bozorg, and A. B. Rad, "A new adaptive UKF algorithm to improve the accuracy of SLAM," *Int. J. Robot.*, vol. 5, no. 1, pp. 35–46, 2019.

[31] M. U. M. Bhutta, M. Kuse, R. Fan, Y. Liu, and M. Liu, "Loop-box: Multiagent direct SLAM triggered by single loop closure for large-scale mapping," *IEEE Trans. Cybern.*, early access, Nov. 6, 2020, doi: 10.1109/TCYB.2020.3027307.

[32] G. Guennebaud *et al.* "The Eigen 3 C++ Library." Jan. 2022. [Online]. Available: https://eigen.tuxfamily.org

[33] K. Y. Leung, Y. Halpern, T. D. Barfoot, and H. H. Liu, "The UTIAS multi-robot cooperative localization and mapping dataset," *Int. J. Robot. Res.*, vol. 30, no. 8, pp. 969–974, 2011.

[34] J. Kuti. "C++ Library for Relaxed Unscented Transformation." 2021. [Online]. Available: https://github.com/ABC-iRobotics/RelaxedUnscentedTransformation.git