Tensor Decompositions for Online Grid-Based Terrain-Aided Navigation

J. Matoušek, J. Krejčí, J. Duník, and R. Zanetti

***Abstract*—This paper presents a practical and scalable grid--based state estimation method for high-dimensional models with invertible linear dynamics and with highly non-linear measurements, such as the nearly constant velocity model with measurements of e.g. altitude, bearing, and/or range. Unlike previous tensor decomposition-based approaches, which have largely remained at the proof-of-concept stage, the proposed method delivers an efficient and practical solution by exploiting decomposable model structurespecifically, block-diagonal dynamics and sparsely coupled measurement dimensions. The algorithm integrates a Lagrangian formulation for the time update and leverages low-rank tensor decompositions to compactly represent and effectively propagate state densities. This enables real-time estimation for models with large state dimension, significantly extending the practical reach of grid-based filters beyond their traditional low-dimensional use. Although demonstrated in the context of terrain-aided navigation, the method is applicable to a wide range of models with decomposable structure. The computational complexity and estimation accuracy depend on the specific structure of the model. All experiments are fully reproducible, with source code provided alongside this paper (GitHub link: https://github.com/pesslovany/Matlab-LagrangianPMF).**

Keywords: state estimation, Bayesian inference, non-linear systems, grid-based filters, Lagrangian filters, CPD, curse of dimensionality, Point-mass filter

I. Introduction

State estimation of discrete-time stochastic dynamical systems from noisy measurements has been a subject of significant research interest for decades. Following the Bayesian framework, a general solution to the state estimation problem is provided by the Bayesian recursive relations (BRRs), which compute the probability density functions (PDFs) of the state conditioned on the available measurements. These conditional PDFs offer a complete probabilistic description of the unobservable state of non-linear or non-Gaussian systems. However, the BRRs are analytically tractable only for a limited class of models, typically those exhibiting linearity. This class of exact Bayesian estimators is represented, for example, by the Kalman filter (KF) for linear and Gaussian models [1], [2]. In all other cases, approximate solutions to the BRRs must be employed. These approximate filtering methods are commonly classified into global and local filters, depending on the validity of their estimates [3], [4].

Local filters are computationally efficient but can diverge under strong nonlinearity or non-Gaussianity. This paper focuses on global filters, which are more robust but traditionally limited by computational complexity. Two main approaches to solving the BRRs globally employ either stochastic or deterministic numerical integration schemes. Particle filters (PFs), also known as Monte Carlo methods [5], use stochastic integration, while grid-based filters (GbFs) employ deterministic numerical integration over discretized state spaces.

The baseline GbF, often referred to as the point-mass filter, was introduced in the 1970s [3] and later applied to navigation applications [6]. Standard GbFs evaluate conditional PDFs at grid points spanning the continuous state space [7], and are generally considered more stable than PFs [8], [9]. However, their major limitation is the exponential growth in computational and memory requirements with increasing state dimensiona manifestation of the curse of dimensionalitywhich makes them impractical for problems above four dimensions, even with the state-of-the-art optimized implementations.

This paper proposes a grid-based filter that scales linearly with state dimension, which is made possible by imposing specific structural assumptions on the model. Namely, we assume that the system dynamics matrix is block-diagonal, the dynamics noise covariance matrix is diagonal, so state variables evolve independently (or in loosely-coupled blocks), and that each measurement dimension depends only on a subset of the state variables. Importantly, the dependency structures in the time update and measurement update need not coincide, and thus the method does not reduce to independent filters; extensions that allow for coupling through non-diagonal (fully coupled) process noise covariance matrices are also under investigation. These assumptions hold in many practical applications, such as terrain-aided navigation, or radar and visual tracking, to name a few.

The core of the proposed solution is the canonical polyadic decomposition (CPD)also known as CANDECOMP or PARAFACwhich represents tensors as sums of rank-one components that are memory efficient compared to the full tensor representation. The entire estimation is performed in this compressed CPD format, i.e., for the rank-one components, and the corresponding full tensors never need to be constructed explicitly. The primary advantage of this approach is that it enables scalable filtering in high-dimensional spaces, while fully exploiting model structure for computational savings and improved accuracy. The proposed method also efficiently solves the main drawback of the CPD, that the tensor rank

J. Matoušek and R. Zanetti are with The University of Texas at Austin, Austin, TX 78712 USA (e-mails: jakub.matousek@austin.utexas.edu, renato@utexas.edu).

J. Krejčí and J. Duník are with the Department of Cybernetics, Faculty of Applied Sciences, University of West Bohemia, Pilsen, Czech Republic (e-mails: {krejci,dunikj}@kky.zcu.cz).

The work of J. Matoušek and R. Zanetti was supported by the Air Force Office of Scientific Research (AFOSR) under award FA9550-23-1-0646. The work of J. Krejčí and J. Duník was supported by the Czech Science Foundation (GACR) under grant GA 25-16919J.

grows over time.

The proposed approach is evaluated on real-world data from a terrain-aided navigation (TAN) scenario and is shown to significantly reduce computational complexity even for 4D state estimation, while maintaining high accuracy.

For completeness, we briefly compare the proposed method to other methods that use tensor decompositions to overcome the curse of dimensionality in estimation as well. Namely:

- **Functional decomposition** [10] uses non-negative tensor factorization to approximate the transient density in a closed region by separating functions of past and future states. While this method was proven effective for mid-dimensional problems, its scalability appear to be challenging.
- **Tensor-train decomposition** [11] requires both likelihood and transition probability tensors being decomposed to tensor trains at runtime, making it computationally intensive and unstable. Current results remain at the proof-of-concept stage.
- **CPD-based methods** [12], [13] provided the initial inspiration for this work but suffer from several practical limitations. Namely, the grid is fixed (non-moving, non-adapting), the advection step is based on inaccurate finite differencesresulting in limited practical usability.

Compared to the mentioned methods that are also based on CPD, the proposed method offers several key advantages:

- The grid moves with the state and the state estimate, which enables efficient solution to the time update using the Lagrangian formulation [14].
- The resolution of the grid can adapt during estimation, which allows solution with constant relative numerical solution accuracy.
- The model structure is fully exploited, leading to higher accuracy and significantly lower computational cost.
- The estimation is formulated in discrete time, which enables direct implementation on digital processors.

The proposed method is implemented using the CPD class and rounding functions from the Tensor Toolbox [15]. All experiments are fully reproducible using real-world data, and the source code is made available at this link[1].

The remainder of the paper is organized as follows. Section II introduces the principles of grid-based Bayesian estimation. Section III presents the canonical polyadic decomposition (CPD) and derives the necessary mathematical operations in the CPD format. Section IV integrates these components to formulate the proposed filter. Section V verifies the method on a real-world terrain-aided navigation dataset. Finally, Section VI concludes the paper and outlines directions for future research.

II. Grid-based Bayesian Estimation

The model dynamics and measurement equation considered in this paper are, respectively,

$$\mathbf{x}_{k+1} = \mathbf{F}_k\mathbf{x}_k + \mathbf{u}_k + \mathbf{w}_k, \tag{1a}$$

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k, \tag{1b}$$

[1]https://github.com/pesslovany/Matlab-LagrangianPMF

where k is the time step, $\mathbf{x}_k \in \mathbb{R}^D$ is the *unknown* state of the system, $\mathbf{u}_k$ is the *known* input, and $\mathbf{z}_k \in \mathbb{R}^{n_z}$ is the measurement. The matrix $\mathbf{F}_k$ is assumed invertible and it describes the state dynamics, and function $\mathbf{h}_k$ defines the relation between the state and measurement. State and measurement noises $\mathbf{w}_k$ and $\mathbf{v}_k$ are unknown, but their PDFs $p_{\mathbf{w}_k}(\mathbf{w}_k)$ and $p_{\mathbf{v}_k}(\mathbf{v}_k)$, respectively, are known, and not necessarily Gaussian. The state noise covariance matrix $\mathbf{Q}_k$ is assumed to be diagonal, a limitation the authors intend to address in future work.

A. Bayesian Recursive Relations

Following the state-space formulation, the filtering task, can be formalised as an estimation of the state $\mathbf{x}_k$ based on the available measurements $\mathbf{z}^k = [\mathbf{z}_0, \mathbf{z}_1, \dots, \mathbf{z}_k]$, inputs $\mathbf{u}_k$, and the state-space model (1). The *Bayesian* estimation infers the PDF of the state conditioned on available measurements.

The filtering and one-step predictive conditional[2] PDFs are recursively calculated by the Bayes' rule and the Chapman-Kolmogorov equation (CKE), which form the Bayesian recursive relations (BRRs), as

$$p(\mathbf{x}_k|\mathbf{z}^k) \propto p(\mathbf{x}_k|\mathbf{z}^{k-1})p(\mathbf{z}_k|\mathbf{x}_k), \tag{2}$$

$$p(\mathbf{x}_{k+1}|\mathbf{z}^k) = \int p(\mathbf{x}_{k+1}|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}^k)d\mathbf{x}_k, \tag{3}$$

respectively, where $\propto$ denotes equality up to a normalizing constant and where

- $p(\mathbf{x}_k|\mathbf{z}^k)$ is the sought posterior (filtering) PDF at $\mathbf{x}_k$,
- $p(\mathbf{x}_{k+1}|\mathbf{z}^k)$ is the prior (predictive) PDF at $\mathbf{x}_{k+1}$,
- $p(\mathbf{z}_k|\mathbf{x}_k) = p_{\mathbf{v}_k}(\mathbf{z}_k - \mathbf{h}_k(\mathbf{x}_k))$ and $p(\mathbf{x}_{k+1}|\mathbf{x}_k) = p_{\mathbf{w}_k}(\mathbf{x}_{k+1} - \mathbf{F}_k\mathbf{x}_k - \mathbf{u}_k)$ are the measurement likelihood and state transition PDFs obtained from (1), respectively.

B. Point - Mass Density

The grid-based solution to the CKE (3) starts with an approximation of the *known* PDF $p_{\mathbf{x}_k}(\mathbf{x}_k)$[3] by a *piece-wise constant* point-mass density (PMD) [16]. The PMD is defined around the set of N grid points $\Xi_k = \{\boldsymbol{\xi}_{k,i}\}_{i=1}^N,\ \boldsymbol{\xi}_{k,i} \in \mathbb{R}^D$, as

$$\overline{p}(\mathbf{x}_k; \Xi_k) \triangleq \sum_{i=1}^{N} P_k^{(i)} S\{\mathbf{x}_k; \boldsymbol{\xi}_{k,i}\}, \tag{4}$$

where $N = \prod_{j=1}^{D} N_j$, with N_j being a user-defined number[4] of grid points in the j-th dimension of the state-space, $P_k^{(i)} \propto p_{\mathbf{x}_k}(\boldsymbol{\xi}_{k,i})$ is a normalised value of the PDF $p(\mathbf{x}_k)$ evaluated at the i-th grid point $\boldsymbol{\xi}_{k,i}$ further called *weight*, and $S\{\mathbf{x}_k; \boldsymbol{\xi}_{k,i}\}$ is an indicator function that equals to 1 if $\mathbf{x}_k$ is in the neighbourhood of the i-th point $\boldsymbol{\xi}_{k,i}$. That is the weight is constant in the neighbourhood of the i-th point. In this paper, an equidistant grid is assumed, i.e., grid where each grid point is associated with the same vector of cell dimension

[2]Shorthand notation of the conditional PDF $p(\mathbf{x}_k|\mathbf{z}^l) = p(\mathbf{x}_k|\mathbf{z}^l; \mathbf{u}^{l-1})$ is used throughout the paper.

[3]The lower index RV notation will be dropped for notation simplicity, where possible.

[4]N_j is usually time independent to achieve constant computational complexity.

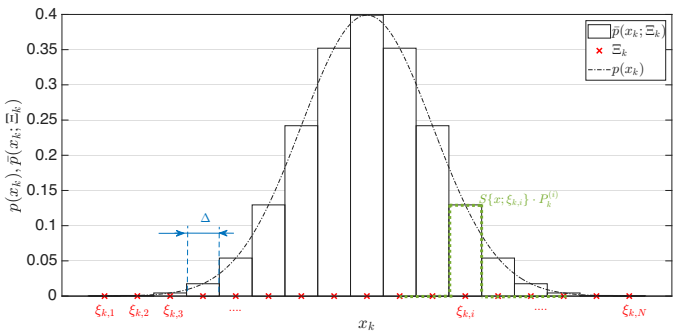

Fig. 1: Points-mass density.

sizes $\boldsymbol{\Delta}_k \in \mathbb{R}^D$ of the same volume δ_k, $\forall i$, as illustrated in Fig. 1. The grid boundaries are assumed to be aligned with the state-space axes, thus the grid can be represented as the Cartesian product

$$\boldsymbol{\Xi}_k = \Xi_k^1 \times ... \times \Xi_k^D \,, \tag{5}$$

requiring storage of only $\sum_{j=1}^{D} N_j$ values, where $\Xi_k^j \subset \mathbb{R}^{N_j}$, $j \in \{1, \ldots, D\}$ is a one-dimensional (equidistant) grid. While the grid was defined as a set, it is implicitly assumed that there is an ordering on the set so that indices of points are readily available at any stage of the algorithm.

The weights evaluated in the grid can be stored in a full tensor whose *order* is that of the state-space dimension D, i.e., $P_k \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_D}$. The *modes* of a tensor refer to one of its dimensions[5] (axes) and are further indexed with $i_1, \ldots i_D$, leading to $P_k^{(i_1, i_2, \ldots, i_D)}$. The order within $(i_1, \ldots, i_D)$ is often irrelevant, in which case we use a single "linear" proxy index i instead, leading to $P_k^{(i)}$, i.e.,

$$i \Leftrightarrow \underbrace{(i_1, ..., i_D)}_{(i_j)_{j=1}^D} \in \underbrace{\{1, \ldots, N_1\} \times \cdots \times \{1, \ldots, N_D\}}_{\mathcal{I}} \,. \tag{6}$$

With this notation, the normalisation of weights can be conveniently written as $P_k^{(i)} = \frac{P_k^{(i)}}{\delta_k \sum_i P_k^{(i)}}$.

C. Lagrangian Grid-based Solution

The Lagrangian grid-based filter (LGbF) working with the tensors in the full format is a state-of-the-art method for efficient state estimation in models with linear dynamics and non-linear measurements[6], such as (1), see [14]. The approach proposed in this paper builds upon this method.

1) Measurement Update: The measurement update step, given by (2), applies Bayes rule to incorporate the latest measurement information into the conditional PMD, which, in the case of point-mass densities, reduces to a simple weight update

$$P_{k|k}^{(i)} \propto \underbrace{p_{\mathbf{v}_k}\left(\mathbf{z}_k - \mathbf{h}_k(\boldsymbol{\xi}_{k,i})\right)}_{P_{\mathbf{z}_k|\mathbf{x}_k}^{(i)}} P_{k|k-1}^{(i)}, \tag{7}$$

where $P_{\mathbf{z}_k|\mathbf{x}_k}^{(i)}$ are the likelihood weights for a fixed $\mathbf{z}_k$, $P_{k|k}^{(i)}$ are the posterior (filtering) weights, and $P_{k|k-1}^{(i)}$ are the prior (prediction) weights, $\forall i \in \mathcal{I}$ (6). The measurement update can directly be given for tensors of the PMD weights and tensors of likelihood values as

$$P_{k|k} \propto P_{\mathbf{z}_k|\mathbf{x}_k} \odot P_{k|k-1}, \tag{8}$$

where $\odot$ is the Hadamard (element-wise) product.

2) Time Update: The time-update is performed in two steps. The first step solves the deterministic part of the time-update $\mathbf{x}_{k+1}^{\text{adv}} = \mathbf{F}_k \mathbf{x}_k$ using the advection (grid movement)

$$\boldsymbol{\xi}_{k+1,i} = \mathbf{F}_k \boldsymbol{\xi}_{k,i}, \ \forall i \,, \tag{9}$$

followed by solving the stochastic part of the time-update described by model $\mathbf{x}_{k+1} = \mathbf{x}_{k+1}^{\text{adv}} + \mathbf{w}_k$. For this model the CKE (3) is reduced to a convolution (PDF of a sum of two random variables is a convolution of their PDFs). Considering the respective densities in the form of the PMDs and using the convolution theorem, the PMD evolution is given by

$$P_{k+1|k} = \mathcal{F}^{-1}(\mathcal{F}(P_{k|k}) \odot \mathcal{F}(W_k)), \tag{10}$$

where $\mathcal{F}$ is the Fourier transform (FT), $\mathcal{F}^{-1}$ is the inverse FT, and the elements of $W_k \in \mathbb{R}^{N_1 \times \ldots \times N_D}$ are given by pointwise evaluation of the state noise PDF $p_{\mathbf{w}_k}(\boldsymbol{\xi}_{k+1}^{\text{mean}} - \boldsymbol{\xi}_{k+1,i})$ with $\boldsymbol{\xi}_{k+1}^{\text{mean}}$ being the sample mean over all points in $\boldsymbol{\Xi}_{k+1}$. If N_j is set odd $\forall j$, then $\boldsymbol{\xi}_{k+1}^{\text{mean}}$ is one of the grid points (the "centre") of $\boldsymbol{\Xi}_{k+1}$, which further simplifies the calculations.

Although the grid $\boldsymbol{\xi}_{k+1}$ is formally defined by the flow (9), the grid design remains fully flexible. In particular, interpolation from $\boldsymbol{\xi}_k$ to an alternative grid with appropriate support may be performed prior to applying the flow [14]. This allows the grid to be designed to incorporate measurement information while ensuring that the effect of process noise is properly accounted for during the recursion, as neither of these effects is captured by the flow itself.

In this paper, the LGbF is derived in the computationally and memory efficient CPD format of the tensors. In the following sections, the CPD background is introduced first, and then, the CPD-based LGbF is derived.

III. Canonical Polyadic Decomposition

The CPD [18] enables the representation of high-order tensors using only a set of factor matrices. Most importantly, the storage and computational complexities of the standard operations on tensors in CPD format scale linearly with the number of state dimension (i.e., the order of the tensor), in contrast to the exponential complexity associated with operations on full-format tensors.

The rank-R CPD approximation of a tensor $P \in \mathbb{R}^{N_1 \times \cdots \times N_D}$ is given by a series of outer products as

$$P \approx \sum_{r=1}^{R} \lambda_P^{(r)} \mathbf{p}_1^{(:,r)} \circ \mathbf{p}_2^{(:,r)} \circ \cdots \circ \mathbf{p}_D^{(:,r)}, \tag{11}$$

where $\circ$ denotes an outer product, $\lambda_P^{(r)}$ is the weight of the r-th rank component, and $\mathbf{p}_j^{(:,r)}$ denotes the r-th column (loading vector) of the j-th factor matrix. That is, there are D matrices

[5]Considering the matrix, the first mode indexes *rows* while the second *columns*.

[6]Paper on how to extend the Lagrangian approach to any invertible dynamics is under review [17].

$\mathbf{p}_j \in \mathbb{R}^{N_j \times R},\ j \in \{1, \ldots, D\}$. The rank R, which is the user-defined parameter, controls the trade-off between the accuracy and storage complexity of the CPD representation.

The memory-intensive quantities that must be stored during state estimation by the LGbF are the prior and posterior weights $P_{k+1|k}$ and $P_{k|k}$, respectively. The full-format tensors are assumed to be too large to construct explicitly and therefore remain represented by the low-rank elements of the CPD form throughout the entire estimation process.

The proposed algorithm relies on the following operations:

- Hadamard (element-wise) product in CPD format.
- CPD decomposition of special cases of tensors.

These operations are presented in the following subsections.

A. Hadamard Product for the CPD Format

Let two tensors A and B in the CPD form be considered

$$A = \sum_{r=1}^{R_A} \lambda_A^{(r)}\, \mathbf{a}_1^{(:,r)} \circ \mathbf{a}_2^{(:,r)} \circ \cdots \circ \mathbf{a}_D^{(:,r)}, \tag{12a}$$

$$B = \sum_{s=1}^{R_B} \lambda_B^{(s)}\, \mathbf{b}_1^{(:,s)} \circ \mathbf{b}_2^{(:,s)} \circ \cdots \circ \mathbf{b}_D^{(:,s)}. \tag{12b}$$

That is, the individual elements of A and B are

$$A^{(i_1,\ldots,i_D)} = \sum_{r=1}^{R_A} \lambda_A^{(r)} \prod_{k=1}^{D} \mathbf{a}_k^{(i_k,r)}, \tag{13a}$$

$$B^{(i_1,\ldots,i_D)} = \sum_{s=1}^{R_B} \lambda_B^{(s)} \prod_{k=1}^{D} \mathbf{b}_k^{(i_k,s)}. \tag{13b}$$

Lemma 1: The Hadamard product $C = A \odot B$ is given by

$$C = \sum_{r=1}^{R_A}\sum_{s=1}^{R_B} \left(\lambda_A^{(r)}\lambda_B^{(s)}\right) \left(\mathbf{a}_1^{(:,r)} \odot \mathbf{b}_1^{(:,s)}\right) \circ \cdots \circ \left(\mathbf{a}_D^{(:,r)} \odot \mathbf{b}_D^{(:,s)}\right). \tag{14}$$

Proof: Evaluating the product $C = A \odot B$ at the index $(i_1, \ldots, i_D)$ reads

$$\begin{aligned} C^{(i_1,\ldots,i_D)} &= \left(\sum_{r=1}^{R_A} \lambda_A^{(r)} \prod_{k=1}^{D} \mathbf{a}_k^{(i_k,r)}\right)\left(\sum_{s=1}^{R_B} \lambda_B^{(s)} \prod_{k=1}^{D} \mathbf{b}_k^{(i_k,s)}\right) \\ &= \sum_{r=1}^{R_A}\sum_{s=1}^{R_B} \lambda_A^{(r)}\lambda_B^{(s)} \prod_{k=1}^{D} \left(\mathbf{a}_k^{(i_k,r)}\, \mathbf{b}_k^{(i_k,s)}\right), \end{aligned} \tag{15}$$

whose tensor notation yields (14). The ranks R_a and R_b for which the decompositions are exact are data dependent and bounded by $\prod_{i=1}^{D} N_i$. □

It can be seen that the Hadamard product of two CPD tensors results in a new CPD tensor with rank equal to product of their ranks, i.e., $R_A R_B$. The cost of computing each new component involves D Hadamard products of vectors of length N_j, costing $\mathcal{O}(\sum_{j=1}^{D} N_j)$ repeated for each pair (r, s) across $R_A \times R_B$ terms. Hence, the total computational complexity is

$$\mathcal{O}\left(R_A R_B \sum_{i=j}^{D} N_j\right), \tag{16}$$

This operation thus scales linearly with the tensor order (state dimension) D.

B. Special Cases of Decomposition

During proposed CPD based state estimation routine, certain special cases of tensors need to be decomposed into CPD format. This subsection presents these special cases and their corresponding efficient CPD decompositions.

1) Tensor of Initial Condition Weights: It is assumed that the initial density for $\mathbf{x}_0 = \left[x_0^{(1)}, \ldots, x_0^{(D)}\right]^{\mathrm{T}}$ can be approximated by a weighted sum of G Gaussian components with diagonal covariance matrices,

$$p(\mathbf{x}_0) \approx \hat{p}(\mathbf{x}_0) = \sum_{g=1}^{G} w_g \prod_{d=1}^{D} \mathcal{N}\left(x_0^{(d)}; \mu_{0,g}^{(d)}, \left(\sigma_{0,g}^{(d,d)}\right)^2\right), \tag{17}$$

where $w_g \geq 0$ and $\sum_{g=1}^{G} w_g = 1$ are the mixture weights, and $\mu_{0,g}^{(d)}$ with $\sigma_{0,g}^{(d,d)}$ are the mean and standard deviation corresponding to $x_0^{(d)}$, respectively. When evaluated on the axis-aligned grid $\boldsymbol{\Xi}_0$, each Gaussian component factorizes along dimensions and therefore yields a rank-one CPD tensor. The initial PMD weights tensor is thus represented as a CPD,

$$P_{0|-1} = \sum_{g=1}^{G} w_g\, \mathbf{p}_1^{(:,g)} \circ \mathbf{p}_2^{(:,g)} \circ \cdots \circ \mathbf{p}_D^{(:,g)}, \tag{18}$$

where $\mathbf{p}_d^{(:,g)}$ is a loading vector corresponding to the g-th Gaussian in (17).

2) Tensor with Some Invariant Modes: A tensor with invariant modes is a tensor whose values do not change along some of its modes, e.g., a matrix whose rows (or columns) are all the same.

Let us consider a tensor $T \in \mathbb{R}^{N_1 \times \cdots \times N_D}$ that varies in only $d < D$ modes. Such a tensor can be fully described by a smaller tensor of unique values, denoted by $M \in \mathbb{R}^{N_1 \times \cdots \times N_d}$. Suppose that M is known, and the goal is to efficiently construct T in the CPD format. We focus on three scenarios:

- For $d = 1$, M is a vector and can be directly used as a loading vector in the CPD at the l-th position, where l is the index of the variant mode. Yielding a rank 1 CPD tensor
$$T = I_{N_1} \circ \cdots \circ I_{N_{l-1}} \circ M^{(:,r)} \circ I_{N_{l+1}} \circ \cdots \circ I_{N_D}, \tag{19}$$
where I_{N_i} is a vector of ones of dimension N_i.
- For $d = 2$, M is a matrix. The singular value decomposition (SVD) is a special case of CPD. Therefore, firstly an SVD of M is calculated
$$M = \mathbf{U}\mathbf{S}\mathbf{V}^T = \sum_{r=1}^{R} \mathbf{S}^{(r,r)}\mathbf{U}^{(:,r)} \circ \mathbf{V}^{(:,r)}, \tag{20}$$

and then the CPD is formed by placing the SVD factors at the appropriate positions (given by variant modes indices) as

$$T = \sum_{r=1}^{R} \mathbf{S}^{(r,r)} \; I_{N_1} \circ \cdots \circ \mathbf{U}^{(:,r)} \circ \cdots \circ \mathbf{V}^{(:,r)} \circ \cdots \circ I_{N_D} . \tag{21}$$

The rank R of the SVD decomposition may be truncated based on a user-defined parameter that determines the proportion of the singular values magnitude to be preserved (e.g. 99.99 %).

- For $d \geq 3$ there is no shortcut and the tensor M has to be decomposed directly by CPD routine, such as *cp_als* routine that is part of the MATLAB© Tensor Toolbox [15] that was used for the implementation of the proposed method. The rank of the CPD decomposition is set-up based on user defined maximal rank of all CPD decompositions during the estimation. The resulting tensor T can be composed of M analogously as for $d = 1$ or $d = 2$.

IV. Grid-based Filtering in CPD Format

This section outlines derivation of the LGbF in CPD format, which employs the weights tensor CPD format representation in the context of high-dimensional non-linear filtering. For the proposed method to be efficient, the model (1) must be such that there are subsets of state and measurement variables exhibiting conditional independency. For convenience, the proposed method is demonstrated on a particular model often utilized in terrain-aided navigation (TAN). The model comes with a public GitHub repository containing a real dataset and several baseline estimators for comparison, enabling validation of the proposed methods performance. The proposed method can be generalized for any model of the form (1) with the said property with little effort.

In the TAN model, it is assumed that the sought state $\mathbf{x}_k$ with $D = 4$ (i.e. state estimation dimension $D = 4$) contains the vehicle *horizontal* position $\mathbf{p}_k^{\mathrm{W}} = [p_k^{x,\mathrm{W}}, p_k^{y,\mathrm{W}}]^T$ [m] and velocity $\mathbf{v}_k^{\mathrm{W}} = [v_k^{x,\mathrm{W}}, v_k^{y,\mathrm{W}}]^T$ [m s^{-1}] in a *world* (W) frame aligned with the geographic north, i.e., $\mathbf{x}_k = [(\mathbf{p}_k^{\mathrm{W}})^T, (\mathbf{v}_k^{\mathrm{W}})^T]^T$. The measurement $\mathbf{z}_k$ with $n_z = 3$ is given by the barometric altimeter, which reads the vehicle altitude above the mean sea level (MSL) $\hbar_k^{\mathrm{MSL}}$ [m], and the odometer, which provides the vehicle velocity in the *body* (B) frame $\mathbf{v}_k^{\mathrm{B}} = [v_k^{x,\mathrm{B}}, v_k^{y,\mathrm{B}}]^T$ [m s^{-1}]. We assume that the heading of the vehicle is aligned with $\mathbf{v}_k^{\mathrm{B}}$; consequently, the W and B frames are rotated by the heading angle ψ_k (angle between $\mathbf{v}_k^{\mathrm{B}}$ and $\mathbf{v}_k^{\mathrm{W}}$) and the respective direction cosine matrix (DCM) is $\mathbf{C}_k = \begin{bmatrix} \cos(\psi_k) & -\sin(\psi_k) \\ \sin(\psi_k) & \cos(\psi_k) \end{bmatrix}$. The model (1) thus reads

$$\mathbf{x}_{k+1} = \begin{bmatrix} \mathbf{p}_k^{\mathrm{W}} \\ \mathbf{v}_k^{\mathrm{W}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_k + \mathbf{w}_k, \tag{22a}$$

$$\mathbf{z}_k = \begin{bmatrix} \hbar_k^{\mathrm{MSL}} \\ \mathbf{v}_k^{\mathrm{B}} \end{bmatrix} = \begin{bmatrix} \mathrm{terMap}(\mathbf{p}_k^{\mathrm{W}}) \\ \mathbf{C}_k \mathbf{v}_k^{\mathrm{W}} \end{bmatrix} + \mathbf{v}_k, \tag{22b}$$

where $\mathrm{terMap}(\cdot)$ is the Digital Terrain Model of the Czech Republic of the 5th generation (DMR 5G) provided by the Czech State Administration of Land Surveying and Cadastre under license CC BY 4.0. The "true" state was measured by GNSS receiver EVK-7 u-blox 7 Evaluation Kit for the *reference purposes*. The values of $\hbar_k^{\mathrm{MSL}}$ were measured by MicroStrain 3DM-CV5-AHRS, and $\mathbf{v}_k^{\mathrm{B}}$ were simulated by perturbing the GNSS data with noise.

In this paper, the measurement noise $\mathbf{v}_k$ and dynamics noise $\mathbf{w}_k$ are zero-mean *normally* distributed random variables with the covariance matrices $\mathbf{R}_k$ and $\mathbf{Q}_k$, respectively, that are assumed to be block-diagonal such that we can write

$$p(\mathbf{x}_{k+1}|\mathbf{x}_k) = p\Big(p_{k+1}^{x,\mathrm{W}}, v_{k+1}^{x,\mathrm{W}} \Big| p_k^{x,\mathrm{W}}, v_k^{x,\mathrm{W}}\Big) \times p\Big(p_{k+1}^{y,\mathrm{W}}, v_{k+1}^{y,\mathrm{W}} \Big| p_k^{y,\mathrm{W}}, v_k^{y,\mathrm{W}}\Big) \;, \tag{23a}$$

$$p(\mathbf{z}_k|\mathbf{x}_k) = p\Big(\hbar_k^{\mathrm{MSL}} \Big| \mathbf{p}_k^{\mathrm{W}}\Big) \cdot p\big(\mathbf{v}_k^{\mathrm{B}} \big| \mathbf{v}_k^{\mathrm{W}}\big) \;, \tag{23b}$$

where terms on each right-hand side involve disjoint state variables from the given $\mathbf{x}_k$. Such independent structure is essential for the efficiency of the proposed method.

A. Measurement Update

The likelihood function $p(\mathbf{z}_k|\mathbf{x}_k)$ (23b) in the tensor notation is

$$P_{\mathbf{z}_k|\mathbf{x}_k} = P_{\mathbf{z}_k^{(1)}|\mathbf{p}_k^{\mathrm{W}}} \odot P_{\mathbf{z}_k^{(2:3)}|\mathbf{v}_k^{\mathrm{W}}} \,. \tag{24}$$

Each of the likelihood tensors $P_{\mathbf{z}_k^{(1)}|\mathbf{p}_k^{\mathrm{W}}}$ and $P_{\mathbf{z}_k^{(2:3)}|\mathbf{v}_k^{\mathrm{W}}}$ varies only along two state dimensions. That is, the unique likelihood values for each likelihood tensor are given by matrices with the elements

$$\mathbf{P}^{(i_1,i_2)}_{\mathbf{z}_k^{(1)}|\mathbf{p}_k^{\mathrm{W}}} = p^{(1)}_{\mathbf{v}_k}\Big(\mathbf{z}_k^{(1)} - \mathrm{terMap}\big(\boldsymbol{\xi}^{(1:2)}_{k,(i_1,i_2,1,1)}\big)\Big) \,, \tag{25a}$$

$$\mathbf{P}^{(i_3,i_4)}_{\mathbf{z}_k^{(2:3)}|\mathbf{v}_k^{\mathrm{W}}} = p^{(2:3)}_{\mathbf{v}_k}\Big(\mathbf{z}_k^{(2:3)} - \mathbf{C}_k \boldsymbol{\xi}^{(3:4)}_{k,(1,1,i_3,i_4)}\Big) \,, \tag{25b}$$

where $p^{(1)}_{\mathbf{v}_k}$ is the one-dimensional noise PDF for the first dimension of the measurement and $p^{(2:3)}_{\mathbf{v}_k}$ is the two-dimensional noise PDF for the second and third measurement dimension. The tensors $P_{\mathbf{z}_k^{(1)}|\mathbf{p}_k^{\mathrm{W}}}$ and $P_{\mathbf{z}_k^{(2:3)}|\mathbf{v}_k^{\mathrm{W}}}$ that have four independent indices can efficiently be constructed from $\mathbf{P}_{\mathbf{z}_k^{(1)}|\mathbf{p}_k^{\mathrm{W}}}$ (25a) and $\mathbf{P}_{\mathbf{z}_k^{(2:3)}|\mathbf{v}_k^{\mathrm{W}}}$ (25b) respectively, using (21), without the need to calculate likelihood for all combinations of four indices.

Likelihoods and posterior weights are now both represented in CPD format. Therefore, the measurement update is performed using (8) and (15) iteratively with

$$P_{k|k} = P_{\mathbf{z}_k^{(1)}|\mathbf{p}_k^{\mathrm{W}}} \odot P_{\mathbf{z}_k^{(2:3)}|\mathbf{v}_k^{\mathrm{W}}} \odot P_{k|k-1} \,. \tag{26}$$

Since the resulting rank is the product of the ranks of all participating tensors, the resulting CPD tensor must be rank-reduced using the *cp_als* routine from the Tensor Toolbox [15].

B. Time Update

The proposed method adopts the Lagrangian form of the time update [14], consisting of two steps: advection and diffusion. In the remainder of this section, these steps are adapted to the CPD format by exploiting the independence structure in the dynamics equations (22a), particularly (23a).

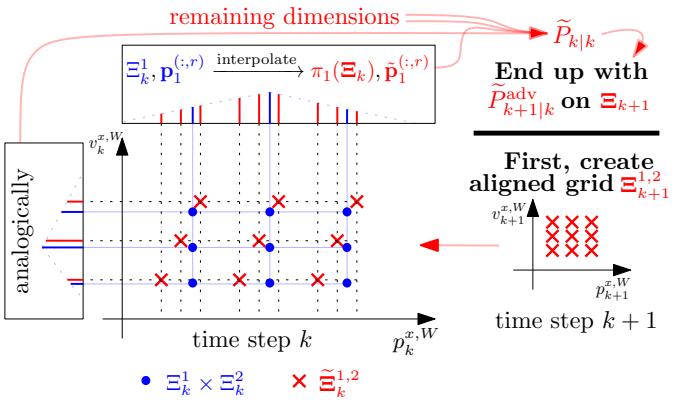

Fig. 2: An axis-aligned grid is transformed back in time (×), towards which we interpolate from the posterior grid (•).

1) Advection Solution: To solve the advection the grid points could be simply flowed according to (9). However, this would result in a grid that is not aligned with the state-space axes at time $k+1$, rendering further computations inefficient and impractical. This issue can be solved by:

- Designing an axes-aligned predictive grid
$$\Xi_{k+1} = \Xi_{k+1}^1 \times \cdots \times \Xi_{k+1}^4 \,. \tag{27}$$
covering the part of the state space determined by the first two moments given by the Kalman filter (KF) prediction that is readily available for the considered model with linear dynamics.
- Transforming the grid Ξ_{k+1} backward in time to yield
$$\widetilde{\Xi}_k := \mathbf{F}_k^{-1}\Xi_{k+1} \triangleq \{\mathbf{F}_k^{-1}\boldsymbol{\xi}_{k+1,j};\ \boldsymbol{\xi}_{k+1,j} \in \Xi_{k+1}\} \tag{28}$$
so that the flow (9) holds, while introducing a structural limitation: the grid $\widetilde{\Xi}_k$ (28) cannot be represented as a Cartesian product and therefore cannot be stored dimension-wise; this limitation is resolved by the construction introduced below.
- Interpolating the weights $P_{k|k}$ from the grid Ξ_k to weights $\widetilde{P}_{k|k}$ on the interpolation grid $\widetilde{\Xi}_k$ which we denote by
$$\Xi_k, P_{k|k} \xrightarrow{\text{interpolation}} \widetilde{\Xi}_k, \widetilde{P}_{k|k} \,, \tag{29}$$
in a way suitable for the CPD format.

The first point is straightforward. The second and third points are treated together in detail further. Illustration is given in Fig. 2 for the joint treatment of $p_k^{x,\mathrm{W}}$ and $v_k^{x,\mathrm{W}}$ dimensions.

Noting that the weights we are interpolating from are stored in the CPD format as
$$P_{k|k} \approx \sum_{r=1}^{R} \lambda^{(r)} \mathbf{p}_1^{(:,r)} \circ \mathbf{p}_2^{(:,r)} \circ \mathbf{p}_3^{(:,r)} \circ \mathbf{p}_4^{(:,r)} \,, \tag{30}$$
we can interpolate the loading vectors $\mathbf{p}_d^{(:,r)}$ for each rank r along each axis (mode) i_d, $d \in \{1,2,3,4\}$ separately. To represent the interpolated weights $\widetilde{P}_{k|k}$ in the CPD format without the need to work with the full tensor, we can exploit the independency (23a). It follows that two tensors of order two, i.e., matrices in our case, must suffice to describe $\widetilde{P}_{k|k}$, and thus the interpolation in x and y directions of the W-frame can be dealt with individually. For advection, the key property is that the dynamics matrix $\mathbf{F}_k$ from (22a) is (after proper re-ordering of state elements) block diagonal
$$\begin{bmatrix} p_{k+1}^{x,\mathrm{W}} \\ v_{k+1}^{x,\mathrm{W}} \\ p_{k+1}^{y,\mathrm{W}} \\ v_{k+1}^{y,\mathrm{W}} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_k^{x,\mathrm{W}} \\ v_k^{x,\mathrm{W}} \\ p_k^{y,\mathrm{W}} \\ v_k^{y,\mathrm{W}} \end{bmatrix} . \tag{31}$$
To solve the interpolation in the x direction[7] of the W-frame, first consider the two-dimensional part of the interpolation grid $\widetilde{\Xi}_k$ (28) for $p_k^{x,\mathrm{W}}$ and $v_k^{x,\mathrm{W}}$ only, i.e., after the appropriate re-ordering,
$$\widetilde{\Xi}_k^{1,2} := \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}^{-1} \left(\Xi_{k+1}^1 \times \Xi_{k+1}^2\right) \,, \tag{32}$$
which is not W-frame (state) axes-aligned as illustrated in Fig. 3 using (×). Notice that the set $\pi_1\big(\widetilde{\Xi}_k^{1,2}\big)$ denoted by (■) of projected points[8] on the first axis contain $\big|\pi_1\big(\widetilde{\Xi}_k^{1,2}\big)\big| = N_1 \cdot N_2$ number of position coordinates, while the set $\pi_2\big(\widetilde{\Xi}_k^{1,2}\big)$ denoted as (●) contains only N_2 velocity coordinates since $\left[\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right]^{-1} = \left[\begin{smallmatrix} 1 & -1 \\ 0 & 1 \end{smallmatrix}\right]$ in (32) contains zero. The loading vectors $\mathbf{p}_1^{(:,r)}$, $\mathbf{p}_2^{(:,r)}$ (30) can now be interpolated from the old grid to new interpolating loading vectors on the interpolation grids as
$$\Xi_k^1,\ \mathbf{p}_1^{(:,r)} \xrightarrow{\text{interpolate}} \pi_1\big(\widetilde{\Xi}_k^{1,2}\big),\ \tilde{\mathbf{p}}_1^{(:,r)}, \quad \forall r \,, \tag{33a}$$
$$\Xi_k^2,\ \mathbf{p}_2^{(:,r)} \xrightarrow{\text{interpolate}} \pi_2\big(\widetilde{\Xi}_k^{1,2}\big),\ \tilde{\mathbf{p}}_2^{(:,r)}, \quad \forall r \,, \tag{33b}$$
which both are simple one-dimensional interpolations. Using $\tilde{\mathbf{p}}_1^{(:,r)}$ and $\tilde{\mathbf{p}}_2^{(:,r)}$ directly as loading vectors, one could potentially construct a large tensor $\tilde{\mathbf{p}}_1^{(:,r)} \circ \tilde{\mathbf{p}}_2^{(:,r)} \in \mathbb{R}^{(N_1 \cdot N_2) \times N_2}$ on the grid $\pi_1\big(\widetilde{\Xi}_k^{1,2}\big) \times \pi_2\big(\widetilde{\Xi}_k^{1,2}\big)$ as illustrated in Fig 3 with (○). However, to solve the advection in a Lagrangian way [14], we are interested only in those entries in $\tilde{\mathbf{p}}_1^{(:,r)} \circ \tilde{\mathbf{p}}_2^{(:,r)}$, $\forall r$, that correspond to the desired subgrid $\widetilde{\Xi}_k^{1,2} \subset \Big(\pi_1\big(\widetilde{\Xi}_k^{1,2}\big) \times \pi_2\big(\widetilde{\Xi}_k^{1,2}\big)\Big)$, see points in Fig. 3 denoted by (×) and (○). In practice, the corresponding matrices (on the mentioned subgrid) denoted as $\widetilde{\mathbf{P}}_{1,2}^r \in \mathbb{R}^{N_1 \times N_2}$ can be constructed $\forall r$ by back-projecting the points $\pi_1\big(\widetilde{\Xi}_k^{1,2}\big)$ into the two dimensions. The back-projection can be realized by a mapping $\mathfrak{m}$ of one-dimensional indices i_x of points in $\pi_1\big(\widetilde{\Xi}_k^{1,2}\big)$ to two-dimensional indices (i_1, i_2) of points in $\widetilde{\Xi}_k^{1,2}$: for the i_xs-th point of $\pi_1\big(\widetilde{\Xi}_k^{1,2}\big)$ there exists an index $(i_1, i_2) = \mathfrak{m}(i_x)$ such that
$$\widetilde{\mathbf{P}}_{1,2}^{r,(i_1,i_2)} = \tilde{\mathbf{p}}_1^{(i_x,r)} \cdot \tilde{\mathbf{p}}_2^{(i_2,r)} \,, \ \forall r \,. \tag{34}$$
Unfortunately, the matrix $\widetilde{\mathbf{P}}_{1,2}^r$ regardless of r cannot[9] be written as an outer product of some vectors $\mathbf{a} \in \mathbb{R}^{N_1}$ and

[7] The interpolation in the y world direction is analogous.

[8] The projection π_j onto the j-th axis is defined as $\pi_j(\Xi) = \cup_{\boldsymbol{\xi} \in \Xi} \{\mathbf{e}_j^T \boldsymbol{\xi}\}$, where $\mathbf{e}_j = [0, \ldots, 0, 1, 0, \ldots, 0]^T$ has 1 on the j-th entry.

[9] The reason is that i_x on the right-hand side of (34) depends on i_2 via $i_x = \mathfrak{m}^{-1}(i_1, i_2)$, i.e., both product terms on the right-hand side of (34) depends on i_2.

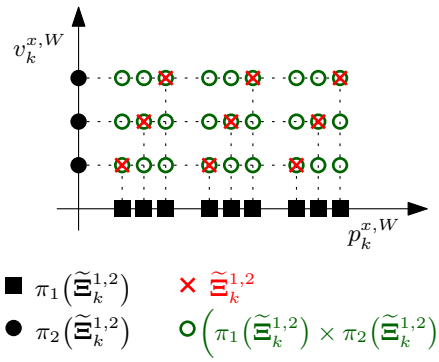

Fig. 3: Illustration of different grids and their projections for x world direction at time step k.

$\mathbf{b} \in \mathbb{R}^{N_2}$ constructed from the loading vectors $\widetilde{\mathbf{p}}_1^{(i_x,r)}$ and $\widetilde{\mathbf{p}}_2^{(i_2,r)}$ by omitting some of its entries, respectively. However, the SVD (20) is readily available for the matrices $\widetilde{\mathbf{P}}_{1,2}^r \; \forall r$ to yield the CPD format

$$\widetilde{\mathbf{P}}_{1,2}^r = \sum_{r_x=1}^{R_x^r} \mathbf{S}_x^{r,(r_x,r_x)} \cdot \mathbf{U}_x^{r,(:,r_x)} \circ \mathbf{V}_x^{r,(:,r_x)}, \tag{35}$$

where R_x^r is the rank of the SVD and $\mathbf{U}_x \in \mathbb{R}^{N_1 \times R_x}$ and $\mathbf{V}_x \in \mathbb{R}^{N_2 \times R_x}$ are the SVD factor matrices.

Repeating the above process for the remaining direction y yields the matrices $\widetilde{\mathbf{P}}_{3,4}^r$, $\forall r$, with the decompositions

$$\widetilde{\mathbf{P}}_{3,4}^r = \sum_{r_y=1}^{R_y^r} \mathbf{S}_y^{r,(r_y,r_y)} \cdot \mathbf{U}_y^{r,(:,r_y)} \circ \mathbf{V}_y^{r,(:,r_y)}, \tag{36}$$

where R_y^r is the rank of the SVD and $\mathbf{U}_y \in \mathbb{R}^{N_1 \times R_y}$ and $\mathbf{V}_y \in \mathbb{R}^{N_2 \times R_y}$ are the SVD factor matrices. The desired tensor $\widetilde{P}_{k|k}$ on the entire 4-dimensional grid $\widetilde{\mathbf{\Xi}}_k$ in the CPD format can be constructed from the above decompositions using (21). Keeping the ordering of state dimensions as above, we have

$$\begin{aligned}\widetilde{P}_{k|k} &= \sum_{r=1}^{R} \sum_{r_x=1}^{R_x^r} \sum_{r_y=1}^{R_y^r} \lambda^r \cdot \mathbf{S}_x^{r,(r_x,r_x)} \cdot \mathbf{S}_y^{r,(r_y,r_y)} \times \\ &\qquad \mathbf{U}_x^{r,(:,r_x)} \circ \mathbf{V}_x^{r,(:,r_x)} \circ \mathbf{U}_y^{r,(:,r_y)} \circ \mathbf{V}_y^{r,(:,r_y)} \\ &= \sum_{\rho=1}^{\mathcal{R}} \widetilde{\lambda}^{(\rho)} \cdot \widetilde{\mathbf{q}}_1^{(:,\rho)} \circ \widetilde{\mathbf{q}}_2^{(:,\rho)} \circ \widetilde{\mathbf{q}}_3^{(:,\rho)} \circ \widetilde{\mathbf{q}}_4^{(:,\rho)}, \end{aligned} \tag{37}$$

where the indices r, r_x, r_y were mapped onto ρ and

$$\widetilde{\lambda}^{(r,r_x,r_y)} = \lambda^r \cdot \mathbf{S}_x^{r,(r_x,r_x)} \cdot \mathbf{S}_y^{r,(r_y,r_y)}, \tag{38a}$$

$$\widetilde{\mathbf{q}}_1^{(:,(r,r_x,r_y))} = \mathbf{U}_x^{r,(:,r_x)}, \tag{38b}$$

$$\widetilde{\mathbf{q}}_2^{(:,(r,r_x,r_y))} = \mathbf{V}_x^{r,(:,r_x)}, \tag{38c}$$

$$\widetilde{\mathbf{q}}_3^{(:,(r,r_x,r_y))} = \mathbf{U}_y^{r,(:,r_y)}, \tag{38d}$$

$$\widetilde{\mathbf{q}}_4^{(:,(r,r_x,r_y))} = \mathbf{V}_y^{r,(:,r_y)}. \tag{38e}$$

The resulting tensor $\widetilde{P}_{k|k}$ has rank $\mathcal{R} = \sum_{r=1}^{R} R_x^r R_y^r$, where R is the rank of the original posterior $P_{k|k}$ (30). Since advection conserves the weights, it follows that the solution to the advection is given by the weights tensor

$$P_{k+1|k}^{\text{adv}} = \widetilde{P}_{k|k} \tag{39}$$

on the grid $\mathbf{\Xi}_{k+1}$ (27), which is axes aligned.

Remark: It can be observed that the CPD rank grows rapidly during the advection process. Therefore, rank reduction is required at the end of advection, which is performed using the *cp_als* routine from the Tensor Toolbox [15].

2) Diffusion Solution: Recalling that $\mathbf{Q}$ is assumed to be diagonal, the solution to the diffusion can be done by simple convolution of 1D Gaussian kernels (given by the dynamics noise) for each diagonal element of $\mathbf{Q}$ with the appropriate loading vectors. Therefore the loading vectors of diffusion solution $\mathbf{p}_j^{\text{dif},(:,r)}$ are calculated from the advection solution loading vectors $\mathbf{p}_j^{\text{adv},(:,r)}$ as[10]

$$\mathbf{p}_j^{\text{dif},(:,r)} = \mathbf{p}_j^{\text{adv},(:,r)} * W_{k,j}, \; \forall r \tag{40}$$

where $*$ denotes convolution, and $W_{k,j} \in \mathbb{R}^{N_j}$ defined as

$$W_{k,j}^{(i_j)} = \mathcal{N}\big((i_j - \bar{i}_j) \cdot \mathbf{\Delta}_{k+1}^{(j)};\; 0, \mathbf{Q}^{(j,j)}\big), \; \forall r \tag{41}$$

are the weights of PMD of Gaussian kernel for j-th state dimension given by the dynamics noise, where $\bar{i}_j = \lceil N_j/2 \rceil$ is the index of the middle grid point, this is the decomposed representation equivalent to W_k in (10).

V. Verification on Terrain-Aided Navigation

This section presents results for the terrain-aided navigation scenario introduced earlier. A total of 20 Monte-Carlo simulations were run, comparing the following filters:

- Lagrangian GbF (LGbF) with $N = 51 \times 51 \times 41 \times 41 \approx 4{,}400{,}000$ [14],
- LGbF with spectral differentiation (LGbFs) with the same N [19],
- Bootstrap PF (PFb), with the same number of particles as points in the GbFs [5],
- Unscented Kalman filter (UKF) [20],
- Proposed CPD-based LGbF (LGbF CPD) with $N_j = 101$, that is $N \approx 100{,}000{,}000$.

Note that the standard GbF is not part of the repository[11] and and it would take hours to days for one step to be computed. Also note that the Rao-Blackwellised PF [21] available in the repository is not included in the comparison, as it was unstable for this setup with forced diagonal process noise covariance matrix, likely due to a model mismatch.

The filters are evaluated using the root mean square error (RMSE) for both position and velocity, along with the average computational time per time step, measured on a MacBook Air M1. The results are presented in Table I. While the UKF is the fastest, it fails to accurately estimate either the position or the velocity. The proposed method achieves the highest position estimation accuracy while also maintaining high velocity estimation accuracy. Importantly, it also exhibits the lowest computational complexity, making it one of the least computationally demanding global methods developed to date. Moreover, it can be executed at least ten times per second, enabling real-time online estimation.

[10]Convolution theorem would not lead to a significantly lower computational complexity as the convolution is only running on N_j points in one dimension.

[11]https://github.com/pesslovany/Matlab-LagrangianPMF

Remark: Readers are encouraged to experiment with the published code; however, it should be noted that the proposed method may suffer from stability issues and negative PMD weights when used with a different set of user-defined parameters (i.e. ranks of different decompositions throughout the algorithm) than those selected for this verification. Addressing this limitation remains an open topic for future research.

TABLE I: Position and velocity RMSE, and average computational time per step.

Method	RMSE [m] Position	RMSE [ms^{-1}] Velocity	Time [s]
LGbF	15.23	0.76	3.88
Spect LGbF	17.00	0.84	0.97
PF bootstrap	14.76	0.82	0.68
UKF	290.97	4.78	0.0008
LGbF CPD (Proposed)	14.13	0.80	0.06

VI. Summary and Future Work

In this paper, we introduced a tensor decomposition-based grid estimation method that dramatically extends the applicability of grid-based filters. By leveraging the structure of the nearly constant velocity model and the terrain-aided navigation measurement equation, the method achieves remarkable computational efficiencyyet it remains general and can be adapted to arbitrary models with invertible dynamics, with complexity and accuracy dictated by the model structure. Most notably, the proposed approach scales linearly with the state dimension, breaking the long-standing curse of dimensionality that has limited grid-based filtering to low-dimensional problems. This advancement opens the door to applying grid-based filters in previously intractable scenarios enabling a new class of high-accuracy solutions across a wide range of applications.

In future work, we aim to develop an algorithm that eliminates the requirement for a diagonalized state noise covariance matrix, enabling greater flexibility and applicability in more general settings. We also plan to remove the assumption of linear dynamics, allowing for the estimation of models with arbitrary invertible dynamics. Furthermore, we intend to demonstrate the proposed method on higher-dimensional estimation problems to rigorously validate its linear computational scaling with respect to state dimension. Finally, there remain unresolved issues related to negative PMD weights and stability when user-defined parameters are not carefully tuned. Addressing these challenges will also be an important focus of future research.

References

[1] M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Practice with MATLAB*, 4th ed. Wiley, 2015.

[2] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley-Interscience, 2006.

[3] H. W. Sorenson, "On the development of practical nonlinear filters," *Information Sciences*, vol. 7, pp. 230–270, 1974.

[4] M. Šimandl and J. Duník, "Derivative-free estimation methods: New results and performance analysis," *Automatica*, vol. 45, no. 7, pp. 1749–1757, 2009.

[5] A. Doucet, N. De Freitas, and N. Gordon, Eds., *Sequential Monte Carlo Methods in Practice*. Springer, 2001, (Ed. Doucet A., de Freitas N., and Gordon N.).

[6] N. Bergman, "Recursive bayesian estimation: Navigation and tracking applications," Ph.D. dissertation, Linköping University, Sweden, 1999.

[7] M. Šimandl, J. Královec, and T. Söderström, "Anticipative grid design in point-mass approach to nonlinear state estimation," *IEEE Transactions on Automatic Control*, vol. 47, no. 4, 2002.

[8] K. B. Ånonsen and O. K. Hagen, "An analysis of real-time terrain aided navigation results from a HUGIN AUV," in *2010 Marine Technology Society (MTS) and the Oceanic Engineering Society of the IEEE Conference (MTS/IEEE OCEANS)*, Seattle, WA, USA, 2010, pp. 1–9.

[9] M. Teng, D. Shuoshuo, L. Ye, and F. Jiajia, "A review of terrain aided navigation for underwater vehicles," *Ocean Engineering*, vol. 281, p. 114779, 2023.

[10] P. Tichavský, O. Straka, and J. Duník, "Grid-based bayesian filters with functional decomposition of transient density," *IEEE Transactions on Signal Processing*, vol. 71, pp. 92–104, 2023.

[11] J. Matoušek, M. Brandner, J. Duník, and I. Punčochář, "Tensor train discrete grid-based filters: Breaking the curse of dimensionality," *IFAC-PapersOnLine*, vol. 58, no. 15, pp. 19–24, 2024, 20th IFAC Symposium on System Identification SYSID 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896324012771

[12] J. Gehlen, M. Ulmke, J. Springer, F. Govaers, and W. Koch, "Tensor decomposition based bearing-only target tracking - an analysis based on real data," in *2024 27th International Conference on Information Fusion (FUSION)*, 2024, pp. 1–6.

[13] F. Govaers, "On statistics based prediction of decomposed 16 tensor probability density functions," in *2023 IEEE Symposium Sensor Data Fusion and International Conference on Multisensor Fusion and Integration (SDF-MFI)*, 2023, pp. 1–6.

[14] J. Matoušek, J. Duník, and O. Straka, "Lagrangian grid-based filters with application to terrain-aided navigation," *IEEE Signal Processing Magazine*, 2025.

[15] B. W. Bader, T. G. Kolda *et al.*, "Tensor Toolbox for MATLAB, version 3.6," http://www.tensortoolbox.org, 2023, latest release, September 28, 2023.

[16] J. Matoušek, J. Duník, and M. Brandner, "Design of efficient point-mass filter with illustration in terrain aided navigation," in *26th International Conference on Information Fusion (FUSION)*, Charleston, USA, 2023, 2023.

[17] J. Dunik, J. Krejčí, J. Matoušek, M. Brandner, and Y. Choe, "Lagrangian grid-based estimation of nonlinear systems with invertible dynamics," *in Proceedings of the 23rd IFAC World Congress (IFAC WC 2026)*, Jul. 2026, under review.

[18] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sapm192761164

[19] J. Matoušek, J. Duník, F. Govaers, and J. Gehlen, "Diffusion in lagrangian grid-based predictors," in *Proceedings of the 2025 Fusion Conference*, Rio de Janeiro, Brazil, 2025.

[20] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *IEEE Proceedings*, vol. 92, no. 3, pp. 401–421, 2004.

[21] Y. Choe, J. W. Song, and C. G. Park, "Lightweight marginalized particle filtering with enhanced consistency for terrain referenced navigation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 3, pp. 2493–2504, 2022.