A Learning Gaussian Process Approach for Maneuvering Target Tracking and Smoothing

WAQAS AFTAB
LYUDMILA MIHAYLOVA, Senior Member, IEEE
The University of Sheffield, Sheffield,

Model-based approaches for target tracking and smoothing estimate the infinite number of possible target trajectories using a finite set of models. This article proposes a data-driven approach that represents the possible target trajectories using a distribution over an infinite number of functions. Recursive Gaussian process, and derivative-based Gaussian process approaches for target tracking, and smoothing are developed, with online training, and parameter learning. The performance evaluation over two highly maneuvering scenarios, shows that the proposed approach provides 80 and 62% performance improvement in the position, and 49 and 22% in the velocity estimation, respectively, as compared to the best model-based filter.

Manuscript received December 16, 2019; revised April 7, 2020 and July 20, 2020; released for publication July 21, 2020. Date of publication September 2, 2020; date of current version February 9, 2021.

DOI. No. 10.1109/TAES.2020.3021220

Refereeing of this contribution was handled by G. Hendeby.

The work of W. Aftab was supported by the Government of Pakistan and Department of ACSE (University of Sheffield). This work was supported in part by the U.K. Engineering and Physical Sciences Research Council through EP/T013265/1 Project NSF-EPSRC:ShiRAS Towards Safe and Reliable Autonomy in Sensor Driven Systems and in part by the EPSRC for the support through the Bayesian Tracking and Reasoning over Time (BTaRoT) under Grant EP/K021516/1. This work was also supported by the USA National Science Foundation under Grant NSF ECCS 1903466.

Authors' address: The authors are with the Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield S10 2TG, U.K., E-mail: (waqasaftabmalik@gmail.com; L.S.mihaylova@sheffield.ac.uk). *(Corresponding author: Waqas Aftab.)*

I. INTRODUCTION

Multiple target tracking [1], [2] includes state estimation of the targets of interest from a set of noisy measurements and false alarms. Multiple target tracking is part of various automation systems in diverse fields. The application spectrum ranges from micro level, such as human cell trackers [3], to macro level, e.g., tracking of aircrafts [1], [2], [4], [5]. Other applications include oceanography [6], sea surveillance systems [7], autonomous vehicles [8], area security [9], and many more.

Multiple target tracking algorithms are sometimes classified as point, extended, and group target tracking [10], [11]. The sensor generates multiple measurements for an extended target. Comparatively, scarce (in some cases single) measurements for a point target are assumed. The process of tracking of multiple point targets possessing similar kinematics, can be performed as a one group and is called group target tracking [10]. The point target tracking requires estimation of the target kinematics such as position and velocity, attributes and other parameters [5]. The extended target tracking and group target tracking, in addition to the abovementioned parameters of the point objects, estimate the target shape, volume, orientation, and other similar parameters. The approach presented in this article relates to point target tracking.

Two major processes involved in point target tracking are the data association, i.e., measurement to target/track assignment, and the state estimation, which includes target state update using the assigned measurement [5]. The performance of these two processes is interdependent. The output of the data association is considered as an input to the state estimation at the same time step. The output of the state estimation is an input to the data association process of the subsequent time step. Many multiple target tracking approaches solve the data association and the state estimation problems independently such as the global nearest neighbor tracker [1], [2], [5], multiple hypothesis tracker [12], and probabilistic data association filter tracker [1]. Other approaches provide a simultaneous solution, e.g., the multiscan Markov chain Monte Carlo data association tracker [13] and the random finite sets based approaches, such as the probability hypothesis density filter trackers [14]. In all the multiple target tracking approaches, the state estimation process can be treated as an independent process. In this article, the measurement to track assignment is assumed known and the solution to the state estimation problem is provided.

Typically, the estimation problem is formulated within a Bayesian framework. The target dynamics is considered nonlinear in most applications. Sometimes, the measurement process is also considered nonlinear. Different filtering methods [1], [5] have been proposed which depend on the extent of model nonlinearity, the required accuracy and the available processing time. The focus of this article is in real-time estimation only and for the first time, an online data driven approach for point target tracking and fixed-lag smoothing is proposed.

A. Related Work

For linear target dynamics and measurement models, the well-known Kalman filter (KF) [15] provides a recursive optimal solution. For nonlinear models, many filters have been proposed such as an extended KF [16], an unscented KF [17], and an interacting multiple model (IMM) [18] filter. The IMM has been shown to be one of the most cost effective nonlinear filters [19]. The KFs based on different nonlinear target dynamics models have also been proposed for tracking nonlinear kinematics, such as the coordinated turn [2] KF and Singer [20] Kalman Filter (KF). All of the above approaches rely on an underlying target dynamics model. A model, closest to the average target trajectory, is chosen and the variations are captured using the model noise. A highly maneuverable target such as an aircraft can adopt an infinite number of trajectories. Additionally, the model variations are expected to be large. In such cases, the single model-based filters require a high model noise variance to track the different trajectories. A high noise variance degrades the estimation performance and gives inaccurate estimates even when the target trajectory matches the model. The multiple model-based filters are an extension of the single model-based filters. These are again limited in performance as all the possible trajectories are not modeled exactly. A preliminary study on a non-recursive model-free filtering, a Gaussian process motion tracker (GPMT), was proposed in [21]. This is a Gaussian process (GP)-based approach, which predicts and filters the position and higher order position derivatives using historical measurement data. The Gaussian Process Motion Tracker (GPMT), includes training and learning stages, models infinite number of target trajectories. The learning stage proposed for the GPMT, is not recursive. In this article, we propose a Recursive Gaussian Process* Motion Tracker (RGP*MT)[1] for online model-free point target tracking. The state estimation and learning of the GP hyperparameters are performed recursively. The historical data is replaced with an inducing point set representation for the recursive filtering and smoothing.

The GP regression method has been applied to many areas involving time series prediction [22], [23]. Target state filtering and smoothing is also a time series application but the Gaussian Process (GP)-based methods have not been studied widely by the target tracking community. GP methods have been proposed for localization [24], [25]. However, the target trajectory in these works is not considered highly maneuverable. This article focuses on both highly maneuverable and less agile targets. Some GP-based methods have been proposed for extended object tracking [26], [27]. These methods represent the object shape using a GP, whereas the target kinematics estimation is performed using model-based approaches for point target tracking. The approach proposed in this article is for the kinematics estimation of point targets. An overlapping mixture of GP, proposed in [28], provides a solution to the data association problem of point targets. To the best knowledge of the authors, this is the first work on a GP-based model-free approach for online target kinematics filtering and smoothing.

[1]The $\star$ indicates that the RGP*MT is a recursive algorithm, whose hyperparameters are learnt online.

B. Contributions

The key contributions of this article are as follows.

1) A data-driven approach is proposed for online point target tracking. The proposed approach is shown to estimate well the unknown and nonlinear target trajectory using the recursive update of the GP hyperparameters (see Section III-B).
2) A model-free recursive approach is proposed for the estimation of the derivative of a GP. This approach can also be extended in the same way to recursively update the higher order derivatives of the GP. The proposed approach is demonstrated by estimating the velocity of a point target (see Section III-C).
3) A recursive point target smoother is proposed for online position and velocity smoothing (see Section III-D). The proposed smoother can be extended for smoothing of the higher order derivatives of the GP in the same way.
4) The proposed filters and smoothers are shown to be robust to the measurement noise model parameters. This is achieved by augmenting the state vector with the measurement noise variance parameter. This parameter is recursively updated at each time sample (see Section III-E). A simulation-based study is also carried out to demonstrate the robustness of the proposed approaches to the measurement noise variance changes (see Section IV-F).
5) The performance evaluation of the proposed approach is done under challenging scenarios and compared with model-based approaches (see Section IV).

The remaining part of this article is structured as follows. Background knowledge for GP and recursive GP methods is given in Section II. The proposed tracker and smoother are introduced in Section III. The performance evaluation and results are presented in Section IV. Finally, Section V concludes this article.

II. BACKGROUND KNOWLEDGE

A. GP Regression

A GP is a stochastic process defining a distribution over possible functions that fit a set of points. A GP is a nonparametric method [23] which can have different probability distributions to fit data to the unknown functions. Most commonly, the squared exponential and Gaussian kernels are used. The GP is defined by two functions, a mean and a covariance kernel. It is assumed that any finite realizations of the GP are mutually independent normally distributed with the given mean and covariance kernel. The parameters of the mean and the covariance kernel are called *hyperparameters*. In a regression problem, the function/model selection is done using the hyperparameters and the given data also called *training* data. The hyperparameters are determined using the training data and the process is called *learning*.

Suppose a one-dimensional (1-D) output r is nonlinearly dependent on a 1-D input u. The nonlinear mapping and the observation model are given as

$$r = f(u), \quad f(u) \sim GP(m(u), k(u, u')) \tag{1}$$

$$\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{u}) + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}) \tag{2}$$

where f is a nonlinear function, u and u' are either the training or the testing input data, $GP(m(u), k(u, u'))$ represents a GP with a mean m and a covariance kernel k, $\boldsymbol{z}$ is the observation vector, $\boldsymbol{f}(\boldsymbol{u})$ represents the true function values at input vector $\boldsymbol{u}$, $\boldsymbol{\epsilon}$ denotes the measurement noise vector whose elements are independent and identically distributed (i.i.d.) Gaussian random variables with variance σ^2 and $\boldsymbol{I}$ denotes an identity matrix. The n-dimensional identity matrix is written as $\boldsymbol{I}_n$. However, often we will not indicate its dimension.

Given an n-dimensional training data set, $\underline{\mathcal{D}}_n = \{(\underline{z}_1, \underline{u}_1), (\underline{z}_2, \underline{u}_2), \ldots, (\underline{z}_n, \underline{u}_n)\}$, the GP regression equations at new input locations are

$$\mathbb{E}[\boldsymbol{f}(\boldsymbol{u}^\star)] = \boldsymbol{m}(\boldsymbol{u}^\star) + \boldsymbol{K}_{\boldsymbol{u}^\star \underline{\boldsymbol{u}}}(\boldsymbol{K}_{\underline{\boldsymbol{u}}\underline{\boldsymbol{u}}} + \sigma^2 \boldsymbol{I}_n)^{-1}(\underline{\boldsymbol{z}} - \boldsymbol{m}(\underline{\boldsymbol{u}})) \tag{3}$$

$$\mathbb{C}[\boldsymbol{f}(\boldsymbol{u}^\star)] = \boldsymbol{K}_{\boldsymbol{u}^\star \boldsymbol{u}^\star} - \boldsymbol{K}_{\boldsymbol{u}^\star \underline{\boldsymbol{u}}}(\boldsymbol{K}_{\underline{\boldsymbol{u}}\underline{\boldsymbol{u}}} + \sigma^2 \boldsymbol{I}_n)^{-1} \boldsymbol{K}_{\underline{\boldsymbol{u}}\boldsymbol{u}^\star} \tag{4}$$

where $\mathbb{E}[\cdot]$ denotes the mathematical expectation operator, $\boldsymbol{f}(\boldsymbol{u}^\star) = [f(u_1^\star), f(u_2^\star), \ldots, f(u_l^\star)]^T$ represents the function prediction vector at the new input vector $\boldsymbol{u}^\star = [u_1^\star, \ldots, u_l^\star]^T$, $\mathbb{E}[\boldsymbol{f}(\boldsymbol{u}^\star)]$ and $\mathbb{C}[\boldsymbol{f}(\boldsymbol{u}^\star)]$ represent, respectively, the predicted mean and the predicted covariance, $\underline{\boldsymbol{z}} = [\underline{z}_1, \underline{z}_2, \ldots, \underline{z}_n]^T$ and $\underline{\boldsymbol{u}} = [\underline{u}_1, \underline{u}_2, \ldots, \underline{u}_n]^T$ are the n-dimensional measurement and input training data vectors, respectively, $\boldsymbol{m}(\underline{\boldsymbol{u}}) = [m(\underline{u}_1), m(\underline{u}_2), \ldots, m(\underline{u}_n)]^T$ represents the n-dimensional GP mean vector, $\boldsymbol{K}_{\boldsymbol{pq}}$ denotes the GP covariance matrix between the input vectors $\boldsymbol{p}$ and $\boldsymbol{q}$, $\cdot^{-1}$ represents the matrix inverse. A GP covariance matrix between a j-dimensional vector $\boldsymbol{u}$ and the l-dimensional vector $\boldsymbol{u}^\star$ is given as

$$\boldsymbol{K}_{\boldsymbol{u}\boldsymbol{u}^\star} = \begin{bmatrix} k(u_1, u_1^\star) & k(u_1, u_2^\star) & \cdots & k(u_1, u_l^\star) \\ k(u_2, u_1^\star) & k(u_2, u_2^\star) & \cdots & k(u_2, u_l^\star) \\ \vdots & \vdots & \ddots & \vdots \\ k(u_j, u_1^\star) & k(u_j, u_2^\star) & \cdots & k(u_j, u_l^\star) \end{bmatrix}. \tag{5}$$

The above GP formulation is applicable to multiple dimensional inputs and outputs in a similar way.

The derivatives of the mean function and covariance kernel are required and are defined as

$$\boldsymbol{\delta}(\boldsymbol{m}(\boldsymbol{u})) = \begin{bmatrix} \left.\frac{\partial m(u)}{\partial u}\right|_{u=u_1} & \cdots & \left.\frac{\partial m(u)}{\partial u}\right|_{u=u_j} \end{bmatrix}^T \tag{6}$$

$$\boldsymbol{\Delta}_{\boldsymbol{u}}(\boldsymbol{K}_{\boldsymbol{u}\boldsymbol{u}^\star}) = \begin{bmatrix} \left.\frac{\partial k(u,u_1^\star)}{\partial u}\right|_{u=u_1} & \cdots & \left.\frac{\partial k(u,u_l^\star)}{\partial u}\right|_{u=u_1} \\ \vdots & \ddots & \vdots \\ \left.\frac{\partial k(u,u_1^\star)}{\partial u}\right|_{u=u_j} & \cdots & \left.\frac{\partial k(u,u_l^\star)}{\partial u}\right|_{u=u_j} \end{bmatrix} \tag{7}$$

$$\boldsymbol{\Delta}^2_{\boldsymbol{u}\boldsymbol{u}^\star}(\boldsymbol{K}_{\boldsymbol{u}\boldsymbol{u}^\star}) = \begin{bmatrix} \left.\frac{\partial^2 k(u,u^\star)}{\partial u \partial u^\star}\right|_{\substack{u=u_1 \\ u^\star=u_1^\star}} & \cdots & \left.\frac{\partial^2 k(u,u^\star)}{\partial u \partial u^\star}\right|_{\substack{u=u_1 \\ u^\star=u_l^\star}} \\ \vdots & \ddots & \vdots \\ \left.\frac{\partial^2 k(u,u^\star)}{\partial u \partial u^\star}\right|_{\substack{u=u_j \\ u^\star=u_1^\star}} & \cdots & \left.\frac{\partial^2 k(u,u_l^\star)}{\partial^{2\sim} u \partial u^\star}\right|_{\substack{u=u_j \\ u^\star=u_l^\star}} \end{bmatrix} \tag{8}$$

where $\boldsymbol{\delta m}(\boldsymbol{u})$ denotes the partial derivative of the mean function, $\boldsymbol{\Delta}_{\boldsymbol{u}}(\boldsymbol{K}_{\boldsymbol{u}\boldsymbol{u}^\star})$ and $\boldsymbol{\Delta}^2_{\boldsymbol{u}\boldsymbol{u}^\star}(\boldsymbol{K}_{\boldsymbol{u}\boldsymbol{u}^\star})$ represent, respectively, the first- and second-order derivative of the covariance matrix, and $\cdot\Big|_{\boldsymbol{\alpha}}$ denotes the variable that is evaluated at $\boldsymbol{\alpha}$.

B. Recursive GP Regression

GP regression is a powerful estimation method with high-computational training and learning stages. The computations scale as $\mathcal{O}(n^3)$ for the $n-$dimensional training data. The learning requires solving a nonconvex optimization problem. Typically, an analytical solution to this optimization problem is not available. Most of the existing numerical solutions are not applicable in real time. In [29], an online approach for GP regression has been proposed. Two methods have been proposed in [29] by assuming bounded input domain. The first one addresses the computational complexity of the training process and is termed as *online regression* or recursive GP (RGP). The second method, in addition to the training, provides online learning and is called *online learning* or recursive GP* (RGP*). The input domain is sampled at discrete points and the input vector representing the sparse grid points is called basis vector $\boldsymbol{u}$. The mean and the covariance at the sparse grid points form the sparse GP representation. The sparse GP representation is updated using the measurements received at each time sample.

C. RGP With Online Regression

This section summarizes the RGP with online regression proposed in [29]. Consider the GP model of (1) and (2). The unknown function f is represented by a multivariate Gaussian distributed vector of N inducing points $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{u}) = [f(u_1), \ldots, f(u_N)]^T$ with an initial distribution $p_0(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{\mu}_0^f, \boldsymbol{C}_0^f)$. At each time sample, the inducing point distribution is updated using the corresponding measurements and the prior distribution, assuming known hyperparameters. The required conditional distribution $p(\boldsymbol{f}|\boldsymbol{z}_{1:k})$ cannot be updated using the GP regression (3) and (4). First, because, the GP regression is a prediction method whereas a measurement update is required. Second, the GP regression requires complete training data whereas a recursive update of a prior distribution is required. The conditional distribution is expanded as follows for the recursive update:

$$p(\boldsymbol{f}|z_{1:k}) = \int c \cdot p(z_k|\boldsymbol{f}, \bar{\boldsymbol{f}}) \cdot p(\bar{\boldsymbol{f}}|\boldsymbol{f}) \cdot p(\boldsymbol{f}|z_{1:k-1}) d\bar{\boldsymbol{f}} \tag{9}$$

where $\bar{\boldsymbol{f}} = \boldsymbol{f}(\boldsymbol{u}_k^{\bar{f}})$ denotes the functional evaluation at the measurement input vector $\boldsymbol{u}_k^{\bar{f}}$ and c is a normalization constant. The product $p(\bar{\boldsymbol{f}}|\boldsymbol{f}) \cdot p(\boldsymbol{f}|z_{1:k-1})$ represents the predictive distribution. The RGP recursion equations are

Prediction :

$$\boldsymbol{J}_k = \boldsymbol{K}_{\boldsymbol{u}_k^{\bar{f}}\boldsymbol{u}} \boldsymbol{K}_{\boldsymbol{uu}}^{-1} \tag{10}$$

$$\boldsymbol{B}_k = \boldsymbol{K}_{\boldsymbol{u}_k^{\bar{f}}\boldsymbol{u}_k^{\bar{f}}} - \boldsymbol{J}_k \boldsymbol{K}_{\boldsymbol{uu}_k^{\bar{f}}} \tag{11}$$

$$\tilde{\boldsymbol{\mu}}_k^{\bar{f}} = \boldsymbol{m}(\boldsymbol{u}_k^{\bar{f}}) + \boldsymbol{J}_k \left(\hat{\boldsymbol{\mu}}_{k-1}^{f} - \boldsymbol{m}(\boldsymbol{u})\right) \tag{12}$$

$$\tilde{\boldsymbol{C}}_k^{\bar{f}} = \boldsymbol{B}_k + \boldsymbol{J}_k \hat{\boldsymbol{C}}_{k-1}^{f} \boldsymbol{J}_k^T \tag{13}$$

Estimation

$$\boldsymbol{M}_k = \tilde{\boldsymbol{C}}_k^{\bar{f}} (\tilde{\boldsymbol{C}}_k^{\bar{f}} + \sigma^2 \boldsymbol{I})^{-1} \tag{14}$$

$$\hat{\boldsymbol{\mu}}_k^{\bar{f}} = \tilde{\boldsymbol{\mu}}_k^{\bar{f}} + \boldsymbol{M}_k \left(z_k - \tilde{\boldsymbol{\mu}}_k^{\bar{f}}\right) \tag{15}$$

$$\hat{\boldsymbol{C}}_k^{\bar{f}} = \tilde{\boldsymbol{C}}_k^{\bar{f}} - \boldsymbol{M}_k \tilde{\boldsymbol{C}}_k^{\bar{f}} \tag{16}$$

$$\boldsymbol{C}_k^{f\bar{f}} = \hat{\boldsymbol{C}}_{k-1}^{f} \boldsymbol{J}_k^T \left(\tilde{\boldsymbol{C}}_k^{\bar{f}}\right)^{-1} \hat{\boldsymbol{C}}_k^{\bar{f}} \tag{17}$$

$$\boldsymbol{G}_k = \hat{\boldsymbol{C}}_{k-1}^{f} \boldsymbol{J}_k^T \left(\tilde{\boldsymbol{C}}_k^{\bar{f}} + \sigma^2 \boldsymbol{I}\right)^{-1} \tag{18}$$

$$\hat{\boldsymbol{\mu}}_k^{f} = \hat{\boldsymbol{\mu}}_{k-1}^{f} + \boldsymbol{G}_k \left(z_k - \tilde{\boldsymbol{\mu}}_k^{\bar{f}}\right) \tag{19}$$

$$\hat{\boldsymbol{C}}_k^{f} = \hat{\boldsymbol{C}}_{k-1}^{f} - \boldsymbol{G}_k \boldsymbol{J}_k \hat{\boldsymbol{C}}_{k-1}^{f} \tag{20}$$

where $\boldsymbol{u}$ denotes the input basis vector, $\boldsymbol{u}_k^{\bar{f}}$ represents the input vector corresponding to the kth measurement vector z_k, $\boldsymbol{J}_k$ and $\boldsymbol{B}_k$ are matrices derived from the GP regression (3) and (4), respectively, $\tilde{}$ and $\hat{}$ represent, respectively, the predicted and filtered variables, $\boldsymbol{\mu}^f$ and $\boldsymbol{C}^f$ are, respectively, the sparse GP mean and covariance of the modeled nonlinear function f at $\boldsymbol{u}$, $\boldsymbol{G}_k$ is the gain matrix, σ^2 is the measurement noise variance hyperparameter, $\boldsymbol{\mu}^{\bar{f}}$ and $\boldsymbol{C}^{\bar{f}}$ represent, respectively, the estimated mean and covariance of the unknown function evaluated at the measured location vector $\boldsymbol{u}_k^{\bar{f}}$, $\boldsymbol{C}_k^{f\bar{f}}$ is the cross-covariance between $\boldsymbol{f}$ and $\bar{\boldsymbol{f}}$, and $\boldsymbol{M}_k$ is the Kalman gain matrix.

To summarize, the inducing points distribution at $k-1$ is used to find the predictive distribution of $\bar{\boldsymbol{f}}$ in (10)–(13). The prediction step utilizes the GP regression of (3) and (4). The predicted distribution is updated using the measurement vector z_k to give the posterior distribution using a KF [15] in (14)–(16). The product of Gaussians is evaluated for the estimated distribution of the inducing points in (18)–(20).

D. RGP With Online Learning

This section summarizes the RGP with online learning proposed in [29]. Consider the GP described by (1) and (2). The hyperparameters are assumed unknown and learned online within the state space framework

$$\begin{bmatrix} \boldsymbol{f}_{k-1} \\ \boldsymbol{\theta}_{k-1} \\ \bar{\boldsymbol{f}}_k \end{bmatrix} = \boldsymbol{A}_k \begin{bmatrix} \boldsymbol{f}_{k-1} \\ \boldsymbol{\theta}_{k-1} \end{bmatrix} + \boldsymbol{v}_k \tag{21}$$

$$\boldsymbol{y}_k = \boldsymbol{A}_k \boldsymbol{x}_{k-1} + \boldsymbol{v}_k \tag{22}$$

$$\boldsymbol{A}_k = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I} \\ \boldsymbol{J}_k & \boldsymbol{O} \end{bmatrix}, \quad \boldsymbol{v}_k \sim \mathcal{N}(\boldsymbol{\mu}_k^{\boldsymbol{v}}, \boldsymbol{C}_k^{\boldsymbol{v}}) \tag{23}$$

where $\boldsymbol{x}_k$ denotes the state vector, $\boldsymbol{y}_k$ is an augmented vector composed of the state vector and the function prediction at the measured locations, $\boldsymbol{\theta}_k = [\boldsymbol{\eta}_k, \sigma_k]^T$ represents the hyperparameters vector, $\boldsymbol{\eta}_k$ denotes the hyperparameters vector of the GP mean and covariance functions, $\boldsymbol{f}$ and $\bar{\boldsymbol{f}}$ are, respectively, the function evaluation at the inducing points and measured locations, $\boldsymbol{A}_k$ is the state update matrix, $\boldsymbol{O}$ denotes a zero vector/matrix of appropriate dimensions, $\boldsymbol{v}_k$ is the additive Gaussian noise vector with independent and identically distributed (i.i.d.) elements and with the following parameters:

$$\boldsymbol{\mu}_k^{\boldsymbol{v}} = \begin{bmatrix} \boldsymbol{O} \\ \boldsymbol{O} \\ \boldsymbol{b}_k \end{bmatrix}, \quad \boldsymbol{C}_k^{\boldsymbol{v}} = \begin{bmatrix} \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{B}_k \end{bmatrix} \tag{24}$$

where $\boldsymbol{b}_k = \boldsymbol{m}(\boldsymbol{u}_k^{\bar{f}}) - \boldsymbol{J}_k \boldsymbol{m}(\boldsymbol{u})$ and $\boldsymbol{J}_k$ and $\boldsymbol{B}_k$ are given by (10) and (11), respectively. The RGP* recursion begins with s sigma points selection around the mean hyperparameters vector $\boldsymbol{\mu}_{k-1}^{\boldsymbol{\theta}}$. In [29], the unscented transform [30] is used for sigma points selection. Given an ith sigma point $\boldsymbol{\theta}_k^i$ with weight $\boldsymbol{w}_k^i$, the predicted state vector and covariance matrix corresponding to the ith sigma point are in the form

$$\boldsymbol{\mu}_k^{\boldsymbol{y}_i} = \boldsymbol{A}_k\Big|_{\boldsymbol{\theta}_k^i} \begin{bmatrix} \boldsymbol{\mu}_{k-1}^{f} + \boldsymbol{S}_k \left(\boldsymbol{\theta}_k^i - \boldsymbol{\mu}_{k-1}^{\boldsymbol{\theta}}\right) \\ \boldsymbol{\theta}_k^i \end{bmatrix} + \boldsymbol{\mu}_k^{\boldsymbol{v}}\Big|_{\boldsymbol{\theta}_k^i} \tag{25}$$

$$\boldsymbol{C}_k^{\boldsymbol{y}_i} = \boldsymbol{A}_k\Big|_{\boldsymbol{\theta}_k^i} \begin{bmatrix} \boldsymbol{C}_{k-1}^{f} - \boldsymbol{S}_k \boldsymbol{C}_{k-1}^{\boldsymbol{\theta} f} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} \end{bmatrix} \boldsymbol{A}_k^T\Big|_{\boldsymbol{\theta}_k^i} + \boldsymbol{C}_k^{\boldsymbol{v}}\Big|_{\boldsymbol{\theta}_k^i} \tag{26}$$

$$\boldsymbol{S}_k = \boldsymbol{C}_{k-1}^{f\boldsymbol{\theta}} \left(\boldsymbol{C}_{k-1}^{\boldsymbol{\theta}}\right)^{-1} \tag{27}$$

where $\boldsymbol{\mu}_k^{\boldsymbol{y}_i}$ and $\boldsymbol{C}_k^{\boldsymbol{y}_i}$ represent, respectively, the state mean and the state error covariance corresponding to the ith sigma point, $\boldsymbol{C}_{k-1}^{\boldsymbol{\theta}}$ represents the covariance of the hyperparameters vector and $\boldsymbol{C}_{k-1}^{\boldsymbol{\theta} f}$ and $\boldsymbol{C}_{k-1}^{f\boldsymbol{\theta}}$ are the cross-covariance matrices between the hyperparameters vector and the sparse GP representation $\boldsymbol{f}$. The combined prediction variables are calculated next, as follows:

$$\boldsymbol{\mu}_k^{\boldsymbol{y}} = \sum_{i=1}^{s} \boldsymbol{w}_k^i \boldsymbol{\mu}_k^{\boldsymbol{y}_i} \tag{28}$$

$$\boldsymbol{C}_k^{\boldsymbol{y}} = \sum_{i=1}^{s} \boldsymbol{w}_k^i ((\boldsymbol{\mu}_k^{\boldsymbol{y}_i} - \boldsymbol{\mu}_k^{\boldsymbol{y}})(\boldsymbol{\mu}_k^{\boldsymbol{y}_i} - \boldsymbol{\mu}_k^{\boldsymbol{y}})^T + \boldsymbol{C}_k^{\boldsymbol{y}_i}) \tag{29}$$

where $\boldsymbol{\mu}_k^{\boldsymbol{y}}$ and $\boldsymbol{C}_k^{\boldsymbol{y}}$ denote, respectively, the mean and covariance of the augmented vector $\boldsymbol{y}$. The state vector $\boldsymbol{x}_k$

is decomposed in to an observed, $\boldsymbol{o}_k = [\sigma_k, (\bar{\boldsymbol{f}}_k)^T]^T$, and unobserved vector, $\boldsymbol{u}_k = [(\boldsymbol{f}_k)^T, \boldsymbol{\eta}_k^T]^T$. The mean vector and covariance matrix corresponding to this decomposition are

$$\boldsymbol{\mu}_k^{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{\mu}_{k-1}^{\boldsymbol{u}} \\ \tilde{\boldsymbol{\mu}}_k^{\boldsymbol{o}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu}_{k-1}^{f} \\ \boldsymbol{\mu}_{k-1}^{\eta} \\ \mu_{k-1}^{\sigma} \\ \tilde{\boldsymbol{\mu}}_k^{\bar{f}} \end{bmatrix} \tag{30}$$

$$\boldsymbol{C}_k^{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{C}_{k-1}^{\boldsymbol{u}} & \tilde{\boldsymbol{C}}_k^{\boldsymbol{uo}} \\ \tilde{\boldsymbol{C}}_k^{\boldsymbol{ou}} & \tilde{\boldsymbol{C}}_k^{\boldsymbol{o}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{C}_{k-1}^{f} & \boldsymbol{C}_{k-1}^{f\eta} & \boldsymbol{C}_{k-1}^{f\sigma} & \tilde{\boldsymbol{C}}_k^{f\bar{f}} \\ \boldsymbol{C}_{k-1}^{\eta f} & \boldsymbol{C}_{k-1}^{\eta} & \boldsymbol{C}_{k-1}^{\eta\sigma} & \tilde{\boldsymbol{C}}_k^{\eta\bar{f}} \\ \boldsymbol{C}_{k-1}^{\sigma f} & \boldsymbol{C}_{k-1}^{\sigma\eta} & C_{k-1}^{\sigma} & \tilde{\boldsymbol{C}}_k^{\sigma\bar{f}} \\ \tilde{\boldsymbol{C}}_k^{\bar{f}f} & \tilde{\boldsymbol{C}}_k^{\bar{f}\eta} & \tilde{\boldsymbol{C}}_k^{\bar{f}\sigma} & \tilde{\boldsymbol{C}}_k^{\bar{f}} \end{bmatrix} \tag{31}$$

where $\boldsymbol{\mu}$ and μ represent, respectively, the mean of a vector and a scalar, $\boldsymbol{C}^{\boldsymbol{a}}$ is the covariance matrix of vector $\boldsymbol{a}$, $\boldsymbol{C}^{\boldsymbol{ab}}$ is the cross-covariance between vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, $\boldsymbol{C}^{ab}$ and $\boldsymbol{C}^{ba}$ denotes the covariance between scalar a and vector b and C^a is the variance of the scalar variable a. The measurement model (2) is reformulated in the form

$$z_k^m = f(u_k^m) + \sigma\beta_k^m, \quad \beta \sim \mathcal{N}(0, 1) \tag{32}$$

to make σ explicit. In (32), β is assumed to be a random variable sampled from the standard Gaussian distribution (which is with a zero mean and covariance equal to 1) scaled by σ, z_k^m and u_k^m are, respectively, the mth measurement of $\boldsymbol{z}_k$ and its corresponding input location, σ is the noise variance (hyperparameter). The β noise is assumed uncorrelated with σ. The mean and the covariances corresponding to the model (32) are given as

$$\boldsymbol{\mu}_k^{\boldsymbol{z}} = \tilde{\boldsymbol{\mu}}_k^{\bar{f}} \tag{33}$$

$$\boldsymbol{C}_k^{\boldsymbol{z}} = \tilde{\boldsymbol{C}}_k^{\bar{f}} + \mathbb{V}[\sigma] + \mathbb{E}[\sigma]^2 = \tilde{\boldsymbol{C}}_k^{\bar{f}} + C_{k-1}^{\sigma} + (\mu_{k-1}^{\sigma})^2 \tag{34}$$

$$\boldsymbol{C}_k^{\boldsymbol{oz}} = \mathbb{C}[\boldsymbol{o}_k, \boldsymbol{z}_k] = \mathbb{C}[\boldsymbol{o}_k, \bar{\boldsymbol{f}}] \tag{35}$$

where $\mathbb{C}[\cdot]$ and $\mathbb{V}[\cdot]$ represent, respectively, a covariance and variance function, $\boldsymbol{\mu}_k^{\boldsymbol{z}}$ and $\boldsymbol{C}_k^{\boldsymbol{z}}$ denote, respectively, the mean and covariance of the predicted measurement vector and $\boldsymbol{C}_k^{\boldsymbol{oz}}$ is the cross-covariance between the observed state and the predicted measurement. It is assumed that $\boldsymbol{o}$ and $\bar{\boldsymbol{f}}$ are jointly Gaussian. The KF equations used to update the observed and unobserved states are

$$\hat{\boldsymbol{\mu}}_k^{\boldsymbol{o}} = \tilde{\boldsymbol{\mu}}_k^{\boldsymbol{o}} + \boldsymbol{E}_k(\boldsymbol{z}_k - \boldsymbol{\mu}_k^{\boldsymbol{z}}) \tag{36}$$

$$\hat{\boldsymbol{C}}_k^{\boldsymbol{o}} = \tilde{\boldsymbol{C}}_k^{\boldsymbol{o}} - \boldsymbol{E}_k\boldsymbol{C}_k^{\boldsymbol{z}}\boldsymbol{E}_k^T \tag{37}$$

$$\boldsymbol{\mu}_k^{\boldsymbol{u}} = \boldsymbol{\mu}_{k-1}^{\boldsymbol{u}} + \boldsymbol{L}_k(\hat{\boldsymbol{\mu}}_k^{\boldsymbol{o}} - \tilde{\boldsymbol{\mu}}_k^{\boldsymbol{o}}) \tag{38}$$

$$\boldsymbol{C}_k^{\boldsymbol{u}} = \boldsymbol{C}_{k-1}^{\boldsymbol{u}} + \boldsymbol{L}_k(\hat{\boldsymbol{C}}_k^{\boldsymbol{o}} - \tilde{\boldsymbol{C}}_k^{\boldsymbol{o}})\boldsymbol{L}_k^T \tag{39}$$

where $\boldsymbol{E}_k = \boldsymbol{C}_k^{\boldsymbol{oz}}(\boldsymbol{C}_k^{\boldsymbol{z}})^{-1}$ and $\boldsymbol{L}_k = \tilde{\boldsymbol{C}}_k^{\boldsymbol{uo}}(\tilde{\boldsymbol{C}}_k^{\boldsymbol{o}})^{-1}$ are the Kalman gain matrices for the observed and unobserved state updates, respectively, and $\tilde{\boldsymbol{C}}_k^{\boldsymbol{uo}}$ is the predicted cross-covariance matrix between the unobserved and observed states and is a submatrix of $\boldsymbol{C}_k^{\boldsymbol{y}}$. The state vector and the covariance matrix are updated as given below

$$\boldsymbol{\mu}_k^{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{\mu}_k^{\boldsymbol{u}} \\ \boldsymbol{h}^T\hat{\boldsymbol{\mu}}_k^{\boldsymbol{o}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu}_k^{f} \\ \boldsymbol{\mu}_k^{\theta} \end{bmatrix} \tag{40}$$

$$\boldsymbol{C}_k^{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{C}_k^{\boldsymbol{u}} & \boldsymbol{L}_k\hat{\boldsymbol{C}}_k^{\boldsymbol{o}}\boldsymbol{h} \\ \boldsymbol{h}^T\hat{\boldsymbol{C}}_k^{\boldsymbol{o}}\boldsymbol{L}_k^T & \boldsymbol{h}^T\hat{\boldsymbol{C}}_k^{\boldsymbol{o}}\boldsymbol{h} \end{bmatrix} \tag{41}$$

where $\boldsymbol{h}^T = [1, \boldsymbol{O}_{N_k}]$, $\boldsymbol{O}_{N_k}$ represents a $1 \times N_k$-dimensional zero vector, and N_k is the number of measurements received at k.

III. PROPOSED DATA-DRIVEN RECURSIVE TRACKING APPROACH

A GPMT was proposed by us in [21]. The GPMT deals with 2-D (x and y) data. It can be extended to any number of dimensions in a similar way. The GPMT assumes that the target motion, in the x and y directions, is mutually uncorrelated. However, both coordinates are correlated in time. Furthermore, it is assumed that the motion correlation in time is limited to the recent past and is uncorrelated with distant past. The filter has a sliding window of data and as a result, the computational complexity of the GP training is reduced for online processing. In target tracking, the higher order position derivatives are also estimated. The first-order GPMT (FO-GPMT[2]) was proposed in [21] for the velocity estimation. The same approach can be extended for estimating other higher order derivatives. *Importance of Learning*. A fixed set of hyperparameters allows tracking of a limited set of target trajectories. The hyperparameters need to be learned for tracking a relatively wide set of target trajectories. In [21], a maximum likelihood (ML) based approach was adopted for learning. The computational cost of this approach [21] is high and the GPMT processing is not in realtime. A recursive GPMT (RGP*MT) is proposed in this article to reduce the computational time of the learning process. As a result, the RGP*MT provides realtime target state estimation.

The GPMT is summarized next followed by the proposed approach.

A. GP Motion Tracker

The GPMT is a GP-based approach for position prediction and filtering of nonlinear target dynamics. It is based on the following assumptions.

1) Cross-coordinates coupling is weak enough to be ignored.
2) Coordinate autocorrelation is available in time.
3) The temporal correlation with input points in distant past is weak.
4) The measurement noise variance is the same for all the measurements.

The cross-coordinate coupling is ignored according to 1. The coordinates are coupled when a target maneuvers [31].

[2]The prefix FO is removed from hereon for brevity.

The effect of ignoring this coupling in a model-free approach such as GPMT has not been studied yet. The coordinate coupling can be included by extending the proposed approach using a coupled GP [32]. The GPMT can estimate any number of output dimensions. For tracking of x and y coordinates, a 2D-GPMT is proposed in [21] and summarized in this section. The GPMT formulation for x-coordinate is presented below, since tracking of the y and other coordinates is done in the same way. The x-coordinate is represented using a GP

$$x = f^x(t), \quad f^x \sim \mathrm{GP}^x(0, k^x(t, t')) \tag{42}$$

$$\boldsymbol{z}^x_{\boldsymbol{t}} = \boldsymbol{f}^x(\boldsymbol{t}) + \boldsymbol{\epsilon}^x_{\boldsymbol{t}}, \quad \boldsymbol{\epsilon}^x_{\boldsymbol{t}} \sim \mathcal{N}(0, \sigma^2_x \boldsymbol{I}) \tag{43}$$

where x represents the horizontal Cartesian coordinate or the output (position) variables, f^x denotes the unknown nonlinear function, GP denotes the GP representation, $\boldsymbol{z}^x_{\boldsymbol{t}}$ represents the measurement vector consisting of samples corresponding to time vector $\boldsymbol{t}$, $\boldsymbol{\epsilon}^x_{\boldsymbol{t}}$ denotes an additive Gaussian noise vector whose elements are i.i.d. with variance denoted by σ^2_x. In the proposed approach the squared exponential covariance kernel [23] is adopted.

The measurements are stored and used for the GP training. Training data older than the d most recent measurement samples, also called depth of the tracker, are assumed uncorrelated and removed. The position prediction and update equations for all output dimensions are similar. The prediction and filtering of the x coordinate, using the GP regression (3) and (4) can be performed with the sequence of equations

$$\tilde{\mu}^x_k = \boldsymbol{K}_{\bar{t}\boldsymbol{t}_k}\bar{\boldsymbol{K}}^{-1}_{\boldsymbol{t}_k}\boldsymbol{z}^x_{\boldsymbol{t}_k} \tag{44}$$

$$(\tilde{\phi}^x_k)^2 = \boldsymbol{K}_{\bar{t}\bar{t}} - \boldsymbol{K}_{\bar{t}\boldsymbol{t}_k}\bar{\boldsymbol{K}}^{-1}_{\boldsymbol{t}_k}\boldsymbol{K}^T_{\bar{t}\boldsymbol{t}_k} \tag{45}$$

$$\hat{\mu}^x_k = \boldsymbol{K}_{\bar{t}\boldsymbol{t}'_k}\bar{\boldsymbol{K}}^{-1}_{\boldsymbol{t}'_k}\boldsymbol{z}^x_{\boldsymbol{t}'_k} \tag{46}$$

$$(\hat{\phi}^x_k)^2 = \boldsymbol{K}_{\bar{t}\bar{t}} - \boldsymbol{K}_{k\boldsymbol{t}'_k}\bar{\boldsymbol{K}}^{-1}_{\boldsymbol{t}'_k}\boldsymbol{K}^T_{\bar{t}\boldsymbol{t}'_k} \tag{47}$$

where $\bar{\boldsymbol{K}}_{\boldsymbol{a}} = [\boldsymbol{K}_{\boldsymbol{aa}} + \sigma^2_x \boldsymbol{I}_d]$, $\bar{t} = k$, $\boldsymbol{t}_k = [k-d, k-d+1, \ldots, k-1]^T$, $\boldsymbol{t}'_k = [k-d+1, k-d+2, \ldots, k]^T$, μ^x and $(\phi^x)^2$ denote, respectively, the position mean and variance. The GPMT prediction is explained in Fig. 1.

The derivative of a GP is also a GP [23]. The existence of a Derivative of a Gaussian Process (DGP) depends on the differentiability of the covariance kernel. A squared exponential kernel is infinitely differentiable and as many position derivatives can be predicted and filtered. Based on the works of [33] and [34], a firstorder GPMT (FO-GPMT) was proposed in [21] and is in the form

$$\tilde{\mu}^{\dot{x}}_k = \boldsymbol{\Delta}_{\bar{t}}(\boldsymbol{K}_{\bar{t}\boldsymbol{t}_k})\bar{\boldsymbol{K}}^{-1}_{\boldsymbol{t}_k}\boldsymbol{z}^x_{\boldsymbol{t}_k} \tag{48}$$

$$(\tilde{\phi}^{\dot{x}}_k)^2 = \boldsymbol{\Delta}^2_{\bar{t}\bar{t}}(\boldsymbol{K}_{\bar{t}\bar{t}}) - \boldsymbol{\Delta}_{\bar{t}}(\boldsymbol{K}_{\bar{t}\boldsymbol{t}_k})\bar{\boldsymbol{K}}^{-1}_{\boldsymbol{t}_k}[\boldsymbol{\Delta}_{\bar{t}}(\boldsymbol{K}_{\bar{t}\boldsymbol{t}_k})]^T \tag{49}$$

$$\hat{\mu}^{\dot{x}}_k = \boldsymbol{\Delta}_{\bar{t}}(\boldsymbol{K}_{\bar{t}\boldsymbol{t}'_k})\bar{\boldsymbol{K}}^{-1}_{\boldsymbol{t}'_k}\boldsymbol{z}^x_{\boldsymbol{t}'_k} \tag{50}$$

$$(\hat{\phi}^{\dot{x}}_k)^2 = \boldsymbol{\Delta}^2_{\bar{t}\bar{t}}(\boldsymbol{K}_{\bar{t}\bar{t}}) - \boldsymbol{\Delta}_{\bar{t}}(\boldsymbol{K}_{\bar{t}\boldsymbol{t}'_k})\bar{\boldsymbol{K}}^{-1}_{\boldsymbol{t}'_k}[\boldsymbol{\Delta}_{\bar{t}}(\boldsymbol{K}_{\bar{t}\boldsymbol{t}'_k})]^T \tag{51}$$

where $\mu^{\dot{x}}$ and $(\phi^{\dot{x}})^2$ denote, respectively, the velocity mean and the variance.

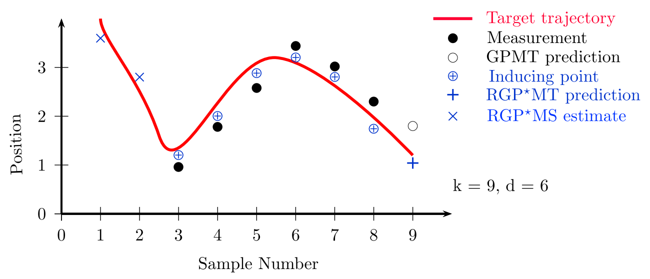

Fig. 1. This figure illustrates the GPMT, RGP*MT, and Recursive Gaussian Process* Motion Smoother (RGP*MS) approaches for output at $k = 9$ using a sliding window width $d = 6$. The target trajectory (solid red line) is sampled at 8 points and the prediction is done for $k = 9$. The GPMT uses $d = 6$ measurement samples (black dots), $3 \leq k \leq 8$, as training data. It predicts the trajectory at 9th sample shown as black empty circle. The RGP*MT uses $d = 6$ inducing points (blue encircled plus) to predict the trajectory shown as blue plus. The RGP*MT filtering at $k = 9$, updates the inducing points based on the new measurement sample. The RGP*MS estimates or the smoothed estimate (blue cross) are the inducing points that are no more part of the training data. After processing of the 9th sample, the training data at $k = 3$ is removed and the data of sample $k = 9$ is added. The removed inducing point becomes the RGP*MS smoothed estimate.

B. Recursive Gaussian Process Motion Tracker (RGP*MT)

The RGP*MT is an online state estimation approach for point target tracking. Consider again (42) and we will present the derivations for the x coordinate. Similar derivations hold true for the y coordinate and other states. The GPMT assumptions 1, 2, 3, and 4 are also valid for the RGP*MT. It is further assumed that the function values at the d most recent measurement samples are represented by a set of d inducing points. This GP representation is different from the sparse GP of [29]. A bounded input domain is assumed in [29], whereas the input dimension for the proposed approach is right unbounded. Using 3, the input location of the inducing points are chosen to be same as that of the d most recent measurements and the input locations of distant past are ignored.

The target is assumed to follow an unknown nonlinear trajectory from time $t = 0$ to $t = t_{\max}$. The input domain is sampled using a fixed equidistant grid also called inducing points. Unlike [29] where multiple measurements are received at random input locations, the measurements are received in a sequence starting at $t = 0$ and going toward $t_{\max}$. Moreover, a single measurement is received per sample. In order to keep the training computational complexity low and using three, a smaller set of inducing points (near the measured location) are considered for the state prediction and filtering. In this way, the inducing points represent the nonlinear function in the near vicinity of the measured location. This reduced set of inducing points is then used for the functional prediction and filtering at the measured location. The processing time of the training was reduced in the same way in GPMT [21]. However, the locations of the inducing points can be chosen different from the measured locations. This is not possible in the GPMT [21]. Another advantage is that the location of the inducing points can be learned at each measurement sample for optimal representation of the slice (of width $d+1$) of the underlying nonlinear function. As a result, the number of inducing points can be reduced when the target motion is uniform. Conversely, when it is exhibiting sharp maneuvers, the inducing points can be increased. The whole process also employs online learning of the hyperparameters which makes the approach highly adaptive.

The inducing points are updated at each time step. Consider the prior inducing points set locations at k are represented by the d-dimensional prior input vector $\boldsymbol{t}_k = [k-d, k-d+1, \ldots, k-1]^T$. The inducing points store an estimate of the nonlinear function f^x in the vector $\boldsymbol{f}_k^x = \boldsymbol{f}^x(\boldsymbol{t}_k)$. The prior inducing points are the training data for the function prediction at $\boldsymbol{t}_k$ and k. The measurement z_k^x, received at k, updates the set of inducing points and the function prediction at the measured locations. The update of the function at the measured location k is the position estimate of the target. The position estimate at k, evaluated using the inducing point set and the measurement, is included in the inducing point set. The oldest, in terms of time, inducing point is removed and a new point is added to the inducing point. As a result, the total number of inducing points is kept fixed. The new point is the function estimate at k denoted as $\bar{f}_k^x = f^x(k)$. The recursion starts at $k = d+1$ sample. The initial distribution of the inducing points set and the hyperparameters vector is determined using the first d measurement samples, in the form

$$p(\boldsymbol{f}_d^x) = \mathcal{N}(\boldsymbol{\mu}_d^{\boldsymbol{f}^x}, \boldsymbol{C}_d^{\boldsymbol{f}^x} = \boldsymbol{K}_{\boldsymbol{t}_d\boldsymbol{t}_d} + (\sigma_d^x)^2\boldsymbol{I}_d) \tag{52}$$

$$p(\boldsymbol{\theta}_d^x) = \mathcal{N}(\boldsymbol{\mu}_d^{\boldsymbol{\theta}^x}, \boldsymbol{C}_d^{\boldsymbol{\theta}^x}) \tag{53}$$

where $p(\cdot)$ is the probability density function, $\boldsymbol{\mu}_d^{\boldsymbol{f}^x}$ and $\boldsymbol{C}_d^{\boldsymbol{f}^x}$ represent, respectively, the mean and covariance of the initial inducing point set and $\boldsymbol{\mu}_d^{\boldsymbol{\theta}^x}$ and $\boldsymbol{C}_d^{\boldsymbol{\theta}^x}$ denote, respectively, the mean and covariance of the initial hyperparameters vector. The prior covariance $\boldsymbol{C}_d^{\boldsymbol{f}^x}$ differs from that proposed in [29] since the inducing points are initialized and recursively learned from noisy measurements. However, the inducing points are not learned exactly due to the measurement noise and the uncertainty in the inducing points is represented with the following assumptions.

1) The inducing points noise is modeled as an additive random noise vector whose elements are i.i.d. with variance $(\sigma_d^x)^2$.
2) The variance of the inducing point is of the order of the measurement noise variance.

As a result of the above assumptions, (10) is represented as

$$\boldsymbol{J}_k' = \boldsymbol{K}_{\boldsymbol{u}_k^{\bar{f}}\boldsymbol{u}}(\boldsymbol{K}_{\boldsymbol{u}\boldsymbol{u}} + (\sigma_k^x)^2\boldsymbol{I}_d)^{-1}. \tag{54}$$

The initial value of the mean vector $\boldsymbol{\mu}_d^{\boldsymbol{\theta}^x}$ is found with the ML estimation based hyperparameters optimization with the first d measurements. The inducing point vector and the hyperparameters vector are combined to define the state vector $\boldsymbol{x}^x = [(\boldsymbol{f}^x)^T, (\boldsymbol{\theta}^x)^T]^T$ and $\boldsymbol{\theta}^x = [\sigma^x, (\boldsymbol{\eta}^x)^T]^T$. The state-space model is given by (21) and (22). For recursive update of the inducing points, the prior at k is assumed to be $p(\boldsymbol{x}_{k-1}^x|z_{1:k-1}^x) = \mathcal{N}(\hat{\boldsymbol{\mu}}_{k-1}^{\boldsymbol{x}^x}, \hat{\boldsymbol{C}}_{k-1}^{\boldsymbol{x}^x})$ and that given the joint distribution of the inducing points at $\boldsymbol{t}_k$ and k, the likelihood of the new measurement is independent of the previous measurements. The posterior $p(\boldsymbol{x}_k^x|z_{1:k}^x)$ can be updated using the decomposition [35] of the state into observable $\boldsymbol{o}_k^x = [\sigma_k^x, (\bar{\boldsymbol{f}}_k^x)^T]^T$ and unobservable $\boldsymbol{u}_k^x = [(\boldsymbol{f}_k^x)^T, (\boldsymbol{\eta}_k^x)^T]^T$ parts

$$p(\boldsymbol{x}_k^x|z_{1:k}^x) = \int p(\boldsymbol{u}_k^x|\boldsymbol{o}_k^x)p(\boldsymbol{o}_k^x|z_{1:k})d\bar{f}_k^x. \tag{55}$$

Inference is done using the RGP* described in Section II-D. The sigma points are chosen using the constrained unscented transform [36], which is different from [29]. The constraints are applied on the hyperparameters to remain positive at all times. The RGP* recursion is adopted under the following equivalent notations:

$$\boldsymbol{u} = \boldsymbol{t}_k,\ \ \boldsymbol{u}_k^{\bar{\boldsymbol{f}}} = k,\ \ \hat{\boldsymbol{x}}_k = \boldsymbol{x}_k^x,\ \ \bar{\boldsymbol{f}}_k = \bar{f}_k^x \tag{56}$$

$$z_k = z_k^x,\ \ \boldsymbol{m}(k) = 0,\ \ \boldsymbol{m}(\boldsymbol{t}_k) = \boldsymbol{O}_d, \boldsymbol{J}_k = \boldsymbol{J}_k' \tag{57}$$

to give the RGP* motion tracker (RGP*MT). Using (56) and (57) and the prior distribution of $\boldsymbol{x}^x$, the posterior is estimated using (21)–(41). The mean $\hat{\mu}_k^{\bar{f}^x}$ and variance $\hat{C}_k^{\bar{f}^x}$ of the function at k can be found after modifying the update (40) and (41) and the new expressions are

$$\hat{\mu}_k^{\bar{f}^x} = \boldsymbol{h}'^T\hat{\boldsymbol{\mu}}_k^{\boldsymbol{o}},\quad \boldsymbol{h}'^T = \boldsymbol{1}_{N_k+1} - \boldsymbol{h}^T \tag{58}$$

$$\hat{C}_k^{\bar{f}^x} = \boldsymbol{h}'^T\hat{\boldsymbol{C}}_k^{\boldsymbol{o}}\boldsymbol{h}' \tag{59}$$

$$\boldsymbol{C}_k^{\boldsymbol{f}^x\bar{f}^x} = \boldsymbol{h}'^T\hat{\boldsymbol{C}}_k^{\boldsymbol{o}}\boldsymbol{L}_k^T \tag{60}$$

where $\boldsymbol{C}_k^{\boldsymbol{f}^x\bar{f}^x}$ represents the cross-covariance vector between the inducing points set and the estimate at the new input location and $\boldsymbol{1}_a$ is a $1\times a$ unit vector. Let the posterior mean and covariance from (40) and (41) are

$$\mathring{\mu}_k^{\boldsymbol{f}^x} = \boldsymbol{\mu}_k^{\boldsymbol{f}},\quad \mathring{C}_k^{\boldsymbol{f}^x} = \boldsymbol{C}_k^{\boldsymbol{f}},\quad \mathring{\boldsymbol{C}}_k^{\boldsymbol{f}^x\bar{f}^x} = \boldsymbol{C}_k^{\boldsymbol{f}\bar{\boldsymbol{f}}}. \tag{61}$$

Then, the posterior distribution of the inducing points can be determined by this posterior mean and covariance given in (61). Let the elements of the updated inducing point set, the covariance matrix and the cross-covariance vector be represented as $\mathring{\boldsymbol{\mu}}_k^{\boldsymbol{f}_i^x}$, $\mathring{\boldsymbol{C}}_k^{\boldsymbol{f}_{i,i}^x}$, and $\mathring{\boldsymbol{C}}_k^{\boldsymbol{f}_i^x\bar{f}^x}$, respectively, where $i = \{1, 2, \ldots, d\}$. The most recent inducing point is denoted with the index d. The following operations are performed on the updated inducing points set to get the modified set and to also keep the number of elements constant

$$\hat{\boldsymbol{\mu}}_k^{\boldsymbol{f}_i^x} = \begin{cases} \mathring{\boldsymbol{\mu}}_k^{\boldsymbol{f}_{i+1}^x} & \text{if } i = 1, 2, \ldots, d-1 \\ \hat{\mu}_k^{\bar{f}^x} & if i = d \end{cases} \tag{62}$$

$$\hat{\boldsymbol{C}}_k^{\boldsymbol{f}_{i,j}^x} = \begin{cases} \mathring{\boldsymbol{C}}_k^{\boldsymbol{f}_{i+1,j+1}^x} \text{if } i,\ j{=}\{1, \ldots, d-1\} \\ \mathring{\boldsymbol{C}}_k^{\boldsymbol{f}_{i+1}^x\bar{f}^x} & \text{if } i = \{1, \ldots, d-1\}, j = d \\ \mathring{\boldsymbol{C}}_k^{\boldsymbol{f}_{j+1}^x\bar{f}^x} & \text{if } i = d, j = \{1, \ldots, d-1\} \\ \hat{C}_k^{\bar{f}^x} & \text{if } i = j = d. \end{cases} \tag{63}$$

In (62) and (63), the oldest inducing point is removed and a new inducing point is added, using the estimates and the corresponding cross-covariance matrix of the current time step. The state vector $\boldsymbol{x}$ defined in (21) and (22) is built using the new inducing point set and the updated hyperparameters vector. The steps from (21) to (41) are repeated for the estimates at the next sample. The RGP⋆MT recursion is explained in Fig. 1.

C. Recursive Derivative of Gaussian Process⋆ Motion Tracker (RDGP⋆MT)

The recursion for a DGP has not been proposed in the literature. In this section, a recursive estimation of the first-order derivative of the x position is proposed. In a similar way, the higher order derivatives of x and the time derivatives for other coordinates can be derived. This formulation is generic and can be applied to any derivative of a GP regression. Consider the GP model defined in (42). The DGP is given as

$$\frac{dx}{dt} = \frac{df^x(t)}{dt} = \dot{f}^x(t). \tag{64}$$

A derivative of a GP is also a GP [23]. As described in Section III-B, the input domain is sampled using a fixed grid of inducing points. The location of the inducing points is chosen the same as for the RGP⋆MT. Let the input vector of the inducing points is $\boldsymbol{t}_k$, then the local estimates of the DGP are stored in the following inducing points:

$$\dot{\boldsymbol{f}}^x = \dot{\boldsymbol{f}}^x(\boldsymbol{t}). \tag{65}$$

Consider the following initial distribution of the DGP:

$$p_d(\dot{\boldsymbol{f}}^x) = \mathcal{N}(\boldsymbol{\mu}_d^{\dot{\boldsymbol{f}}^x}, \boldsymbol{C}_d^{\dot{\boldsymbol{f}}^x} = \boldsymbol{K}_{\boldsymbol{t}_d \boldsymbol{t}_d} + (\sigma_d^{\dot{x}})^2 \boldsymbol{I}_d). \tag{66}$$

The posterior distribution is updated recursively using the Bayes law and the Chapman–Kolmogrov equations given as

$$p(\dot{\boldsymbol{f}}_k^x | \dot{\boldsymbol{z}}_{1:k}^x) = \frac{\int p(\dot{z}_k^x | \bar{\dot{f}}_k^x, \dot{\boldsymbol{f}}_k^x) p(\bar{\dot{f}}_k^x, \dot{\boldsymbol{f}}_k^x | \dot{\boldsymbol{z}}_{1:k-1}^x) d\bar{\dot{f}}_k^x}{c_k} \tag{67}$$

where $\dot{z}$ is the measurement, $\bar{\dot{f}}_k^x$ represents the function evaluation at the measured location, and c_k is the normalization constant. It is assumed that the hyperparameters have been learned during the position filtering step, i.e., RGP⋆MT. Hence, the posterior update is independent of the hyperparameters. The DGP inducing points and the velocity inference and filtering at time k is done recursively following the RGP formulation given in Section II-B and the following equivalence:

$$\boldsymbol{u} = \boldsymbol{t}_k, \;\; \boldsymbol{u}_k^{\bar{f}} = k, \;\; \boldsymbol{f}_k = \dot{\boldsymbol{f}}_k^x, \;\; \bar{\boldsymbol{f}}_k = \dot{\bar{f}}_k^x, \;\; \dot{m}(k) = 0$$
$$\dot{\boldsymbol{m}}(\boldsymbol{t}) = \boldsymbol{O}_d, \;\; \dot{z}_k^x = \frac{z_k^x - z_{k-1}^x}{T}, \;\; (\dot{\sigma}_k^x)^2 = \frac{2(\sigma_k^x)^2}{T}$$

where T is the sampling time and $\dot{z}_k^x$ is a pseudomeasurement with measurement noise variance $(\dot{\sigma}_k^x)^2$. Using the RGP recursion given in Section II-C, the filtered inducing points set and the derivative of the function at k are

$$\mathring{\boldsymbol{\mu}}_k^{\dot{\boldsymbol{f}}^x} = \hat{\boldsymbol{\mu}}_k^{\boldsymbol{f}}, \;\; \mathring{\boldsymbol{C}}_k^{\dot{\boldsymbol{f}}^x} = \hat{\boldsymbol{C}}_k^{\boldsymbol{f}}, \;\; \mathring{\boldsymbol{C}}_k^{\dot{\boldsymbol{f}}^x \dot{\bar{f}}^x} = \boldsymbol{C}_k^{\boldsymbol{f}\bar{\boldsymbol{f}}}$$
$$\hat{\mu}_k^{\dot{\bar{f}}^x} = \hat{\boldsymbol{\mu}}_k^{\bar{f}}, \;\; \hat{C}_k^{\dot{\bar{f}}^x} = \hat{\boldsymbol{C}}_k^{\bar{f}}.$$

At the end of the recursion, the inducing points are updated, similarly to (62) and (63), as follows:

$$\hat{\boldsymbol{\mu}}_k^{\dot{\boldsymbol{f}}_i^x} = \begin{cases} \mathring{\boldsymbol{\mu}}_k^{\dot{\boldsymbol{f}}_{i+1}^x} & \text{if } i = 1, 2, \ldots, d-1 \\ \hat{\mu}_k^{\dot{\bar{f}}^x} & \text{if } i = d \end{cases} \tag{68}$$

$$\hat{\boldsymbol{C}}_k^{\dot{\boldsymbol{f}}_{i,j}^x} = \begin{cases} \mathring{\boldsymbol{C}}_k^{\dot{\boldsymbol{f}}_{i+1,j+1}^x} & \text{if } i, j{=}\{1, \ldots, d-1\} \\ \mathring{\boldsymbol{C}}_k^{\dot{\boldsymbol{f}}_{i+1}^x \dot{\bar{f}}^x} & \text{if } i = \{1, \ldots, d-1\}, j = d \\ \mathring{\boldsymbol{C}}_k^{\dot{\boldsymbol{f}}_{j+1}^x \dot{\bar{f}}^x} & \text{if } i = d, j = \{1, \ldots, d-1\} \\ \hat{C}_k^{\dot{\bar{f}}^x} & \text{if } i = j = d. \end{cases} \tag{69}$$

An alternative way of determining the derivatives (also used in Section IV) is from the inducing points and the learned hyperparameters of the RGP⋆MT. The inducing points and derivatives can be represented by a multivariate Gaussian distribution [34]

$$\begin{bmatrix} \boldsymbol{f}_k^x \\ \dot{\bar{f}}_k^x \end{bmatrix} = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{m}(\boldsymbol{t}_k) \\ \delta(m(\bar{t})) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}_{\boldsymbol{t}_k \boldsymbol{t}_k} + (\sigma_k^x)^2 \boldsymbol{I}_d & \boldsymbol{\Delta}_{\bar{t}}(\boldsymbol{K}_{\boldsymbol{t}_k \bar{t}}) \\ \boldsymbol{\Delta}_{\bar{t}}(\boldsymbol{K}_{\bar{t}\boldsymbol{t}_k}) & \Delta_{\bar{t}\bar{t}}^2(K_{\bar{t}\bar{t}}) \end{bmatrix} \right). \tag{70}$$

The predicted and filtered variables of the first-order derivative process given the inducing points are

$$\tilde{\mu}_k^{\dot{\bar{f}}^x} = \delta(m(\bar{t})) + \boldsymbol{\Delta}_{\bar{t}}(\boldsymbol{K}_{\bar{t}\boldsymbol{t}_k}) \bar{\boldsymbol{K}}_{\boldsymbol{t}_k}^{-1} (\mathring{\boldsymbol{\mu}}_k^{\boldsymbol{f}^x} - \boldsymbol{m}(\boldsymbol{t}_k)) \tag{71}$$
$$\tilde{C}_k^{\dot{\bar{f}}^x} = \boldsymbol{\Delta}_{\bar{t}\bar{t}}^2(\boldsymbol{K}_{\bar{t}\bar{t}}) - \boldsymbol{\Delta}_{\bar{t}}(\boldsymbol{K}_{\bar{t}\boldsymbol{t}_k}) \bar{\boldsymbol{K}}_{\boldsymbol{t}_k}^{-1} [\boldsymbol{\Delta}_{\bar{t}}(\boldsymbol{K}_{\bar{t}\boldsymbol{t}_k})]^T \tag{72}$$
$$\hat{\mu}_k^{\dot{\bar{f}}^x} = \delta(m(\bar{t})) + \boldsymbol{\Delta}_{\bar{t}}(\boldsymbol{K}_{\bar{t}\boldsymbol{t}'_k}) \bar{\boldsymbol{K}}_{\boldsymbol{t}'_k}^{-1} (\hat{\boldsymbol{\mu}}_k^{\boldsymbol{f}^x} - \boldsymbol{m}(\boldsymbol{t}'_k)) \tag{73}$$
$$\hat{C}_k^{\dot{\bar{f}}^x} = \boldsymbol{\Delta}_{\bar{t}\bar{t}}^2(\boldsymbol{K}_{\bar{t}\bar{t}}) - \boldsymbol{\Delta}_{\bar{t}}(\boldsymbol{K}_{\bar{t}\boldsymbol{t}'_k}) \bar{\boldsymbol{K}}_{\boldsymbol{t}'_k}^{-1} [\boldsymbol{\Delta}_{\bar{t}}(\boldsymbol{K}_{\bar{t}\boldsymbol{t}'_k})]^T. \tag{74}$$

D. Recursive Gaussian Process⋆ Motion Smoother (RGP⋆MS) and Recursive Derivative of Gaussian Process⋆ Motion Smoother (RDGP⋆MS)

Training data for the GPMT prediction and update steps are the recent measurements. Comparatively, the RGP⋆MT prediction is based on the inducing points set. The RGP⋆MT update step is performed using the inducing points set and the current measurement. The inducing points set is updated during each measurement update. This update, given by (38) and (39), is similar to a fixed-lag-recursive-smoother. The smoother lag is d measurement samples long according to (62) and (63). As shown in [29], this smoother is similar to the augmented Kalman smoother of [37], with an additional measurement update step. $\hat{\boldsymbol{\mu}}_k^{\boldsymbol{f}_1^x}$ and $\hat{\boldsymbol{C}}_k^{\boldsymbol{f}_{1,1}^x}$ in (62) and (63) are, respectively, the smoothed position and the corresponding variance evaluated at $\boldsymbol{t}'$. Similarly, $\hat{\boldsymbol{\mu}}_k^{\dot{\boldsymbol{f}}_1^x}$ and $\hat{\boldsymbol{C}}_k^{\dot{\boldsymbol{f}}_{1,1}^x}$ in (68) and (69) are, respectively, the smoothed velocity and the corresponding variance. As for the derivative GP, an alternative way of smoothing is through the use of (73) and (74) and setting $\bar{t} = k - d$ and $\boldsymbol{t}'_k = [k, k-1, \ldots, k-d+1]^T$. The RGP*MS is also explained in Fig. 1.

E. Measurement Noise Uncertainty Analysis

The proposed RGP*MT and RGP*MS does not require prior knowledge of the measurement noise variances. This is achieved by setting the measurement variance as a hyperparameter. The measurement noise variance is recursively estimated at each step. As a result, the proposed filter and smoother are robust to the measurement noise errors, provided that the noise is additive Gaussian. The performance of the proposed approaches is studied using different measurement noise variances and the results are presented in Section IV-F.

F. Sparsity and the Inducing Points

The RGP*MT, RGP*MS, RDGP*MT, and RDGP*MS proposed in this article extend further the sparse GP [38], for target tracking and with studies on the impact of the noise parameters on the approaches' performance. A typical sparse GP application has a bounded input domain, whereas the proposed approaches in this article are based upon a right unbounded input domain. Second, a typical sparse GP application requires an estimate across the whole input domain. In target tracking, the interest lies at the current and future time, i.e., at a specific slice of the input domain. Finally, in a typical sparse GP application the training data is available on both sides of the test data. Conversely, in the assumed target filtering problem, the training data is not available on the right side of the input domain.

The proposed approaches introduce sparsity through the inducing points and the parameter d. The inducing points locations are proposed to be the same as those of the measurements. These can be chosen different from the measurement locations as well. The distance between the inducing points location cannot be kept too close or too far. The optimal grid points location is proposed as a function of the lengthscale hyperparameter in [39]. Similarly, the accuracy of the developed approaches is sensitive to the parameter d. Previously, sparse GP methods have been proposed for the automatic selection of the inducing points [40] and choice of the parameter d [41]. A comprehensive review of these methods is given in [42]. In this article, the parameter d is selected by the trial and error approach. In future, automatic selection can be introduced to improve the robustness of the approach with respect to d.

IV. PERFORMANCE VALIDATION

The proposed approach has been validated over challenging maneuvering scenarios presented in this section.

A. Compared Methods

Different model-based and model-free filters described below are compared.

F1) *CV*. A KF with state transition modeled using a nearly constant velocity (NCV) [2] model. The process noise variance is set to $q_{\text{NCV}} = 10\ \text{m}^2/\text{s}^4$.

F2) *Fixed grid interacting multiple model (FGIMM)*. An FGIMM [43] is implemented using three KFs. The grid consists of an NCV and two coordinated turn models. The rate of turns are set to $\{-15, 15\}^\circ/\text{s}$. The Markov transition probability of the same mode is set to 0.9 and for changing the mode is 0.05, the initial model probability vector is $\{0.15, 0.7, 0.15\}$ and the process noise variance is set to $26\ \text{m}^2/\text{s}^4$ for each model. This is an optimum process noise variance for a target moving with 200 m/s according to [44].

F3) *Singer*. A KF designed using a Singer [20] state transition model. The model parameters are set as follows; maximum possible acceleration is set to $A_{\max} = 8\ \text{m/s}^2$, probability of no-acceleration is set to $P_0 = 0.4$, probability of maximum acceleration is set to $P_{\max} = 0.1$ and man oeuvre time constant is set to $\tau_m = 8$ s.

F4) *GPMT, FO-GPMT*. A GPMT and FO-GPMT [21] with depth set to $d = 10$ samples.

F5) *CGPMT, FOCGPMT*. A constant (hyperparameters) GPMT and FOGPMT with depth set to $d = 10$ samples. The hyperparameters are initialized using the first d measurement samples and are kept constant there after.

F6) *RGPMT, RDGPMT*. A filter based on RGP. The learning is ML-based and the depth is set to $d = 10$ samples.

F7) *RGP*MT, RDGP*MT, and RGP*MS, RDGP*MS*. The proposed filter and smoother with depth set to $d = 10$ samples. The smoother is denoted as **S*7**.

Filters **(F1)**, **(F2)** and **(F3)** are chosen to compare the proposed approach against the model-based filtering methods. Filter **(F4)** is chosen to study the effect of proposed changes in the training (inducing points instead of measurements) and learning (recursive instead of ML). Filter **(F6)**, which adopts the same training process as the proposed approach, is introduced to study the performance of different learning approaches, ML in **(F6)** versus recursive in the proposed (F10). Finally, filter **(F5)** is introduced to study the performance degradation if the hyperparameters are not learned.

B. Testing Scenarios

The above mentioned approaches are compared on the following six challenging scenarios.

S1) *Uniform*. The target velocity is kept nearly constant. This scenario represents the commonly observed trajectory of airliners.

S2) *Gradual Coordinated Turns*. The target motion is modeled using the left and right coordinated turns (15°/s for 10 s) and the CV motion models. This scenario represents maneuverable targets dynamics, less agile than those in scenario **(S3)**.

TABLE I
Match Mismatch Matrix

	S1	S2	S3	S4	S5	S6
F1	■	□	□	□	□	□
F2	■	■	▨	□	□	□
F3	□	□	□	■	▨	□
F4	□	□	□	□	□	▨
F5	□	□	□	□	□	▨
F6	□	□	□	□	□	▨
F7	□	□	□	□	□	▨
F8	□	□	□	□	□	▨
F9	□	□	□	□	□	▨

□ Mis-match ▨ Matched model ■ Matched model and parameters

S3) *Sharp Coordinated Turns*. This is similar to **(S2)** except the turn rates are set to (30°/s for 9 s). This scenario represents highly maneuverable targets dynamics such as military aircrafts.

S4) *Singer Lazy*. The target motion is modeled using the Singer acceleration model. The parameters are chosen the same as those of the Singer filter **(F3)**.

S5) *Singer Agile*. This scenario is similar to **(S4)**, except the maximum acceleration is increased to $A^2_{\max} = 50\ \mathrm{m}^2/\mathrm{s}^4$. As compared to **(S4)**, an agile target is simulated in this test scenario.

S6) *GP*. The target motion is modeled using two zero mean GPs, each for the x and y coordinate, respectively. The squared exponential covariance kernel is adopted for both GPs. The hyperparameters are kept constant. The magnitude of variance is set to $\sigma^2_{gp} = 1e7\ \mathrm{m}^2$ and length-scale is set to $l_{gp} = 10$ s.

The target velocity is initialized randomly in the range $150 \leq v_0 \leq 250$ m/s for model-based scenarios **(S1)** to **(S5)**. The total time of trajectory is 100 s. The measurement noise standard deviation is set to $\sigma = 25$ m. The simulated scenarios cover a wide range of maneuvering trajectories depicted by the aerial targets. Each scenario is matched, in terms of the structure and the parameters, to at-least one of the filters. This provides a rigorous performance evaluation for each approach. The matrix for the match and mismatch among the filters and the scenarios is depicted in Table I.

Scenarios **(S5)** and **(S6)** represent highly maneuvering target trajectories which are not matched to any of the compared filters. A sample trajectory of each scenario is shown in Fig. 2. The filters are initialized using the measurement data. The hyperparameters vector is initialized by maximizing likelihood of first d measurement samples for filters **(F6)** and **(F7)**. The position and velocity root mean square errors (RMSE) are plotted in the figures showing performance graphs. The mean-RMSE errors of the position and velocity are given in the performance tables. The RMSE and mean-RMSE in N_{MC} Monte–Carlo simulation runs are evaluated as

$$\mathrm{rmse}^k_q = \sqrt{\frac{1}{N_{\mathrm{MC}}} \sum_{i=1}^{N_{\mathrm{MC}}} (q_{i,k} - \hat{q}_{i,k})^2} \tag{75}$$

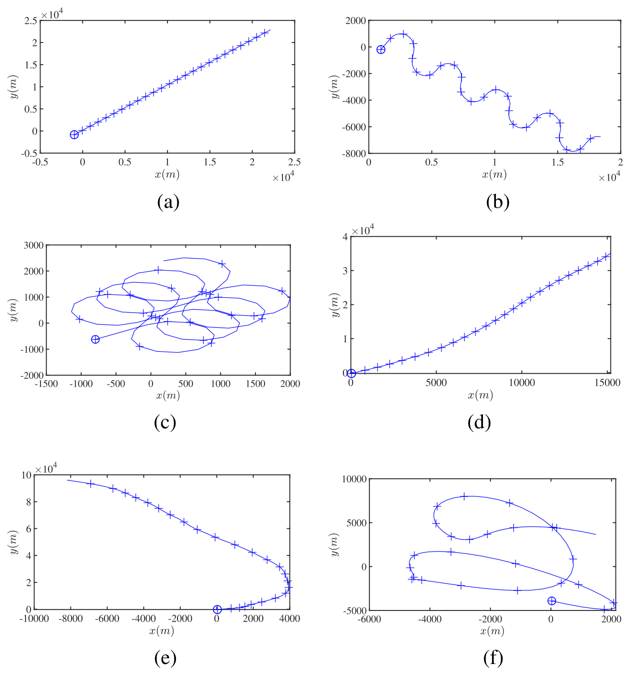

Fig. 2. *Sample Trajectory*. The figure shows a sample trajectory of each scenario. The initial position is indicated by a small circle. (a) S1: CV. (b) S2: Gradual CT. (c) S3: Sharp CT. (d) S4: Singer lazy. (e) S5: Singer Agile. (f) S6: GP.

$$\text{mean rmse} = \sqrt{\frac{1}{K N_{\mathrm{MC}}} \sum_{i=1}^{N_{\mathrm{MC}}} \sum_{k=1}^{K} (q_{i,k} - \hat{q}_{i,k})^2} \tag{76}$$

where rmse^k_q and mean rmse represent, respectively, the RMSE and the mean RMSE, q_k is the true value, $\hat{q}_k$ is the filter output, and K is the total number of samples.

C. Implementation Details

The implementation details of the proposed approach are given in this section. The state vector (excluding the hyperparameters) and the measurement data are scaled down at the input and upscaled at the output of the filter. The scaling is set to $\frac{1}{70}$. The first two inducing points for the DGP are initialized to the same value. The mean hyperparameters vector is initialized using the ML of the first d measurement samples. Let the initial hyperparameters vector is given as $\boldsymbol{\mu}^{\boldsymbol{\theta}}_0 = [\sigma^2_{\mathrm{ker}}, l, \sigma^2]^T$, where σ^2_{ker} and l are the kernel variance and lengthscale hyperparameters and σ^2 is the noise variance hyperparameter. The correlation between the noise variance hyperparameter and $\boldsymbol{f}(\boldsymbol{u}^{\bar{f}}_k)$ is necessary for its learning in (36) and (37). As proposed in [29], this correlation is attained by correlating the noise variance hyperparameter with the remaining hyperparameters. To achieve this, the terms in the covariance of the hyperparameters relating to the noise variance hyperparameter matrix are set to nonzero values. The initial covariance matrix is

$$\boldsymbol{C}^{\boldsymbol{\theta}}_0 = \begin{bmatrix} \sigma^2_{\mathrm{ker}} & 0 & \frac{\sigma^2}{40} \\ 0 & 1 & \frac{\sigma^2}{40} \\ \frac{\sigma^2}{40} & \frac{\sigma^2}{40} & \frac{\sigma^2}{20} \end{bmatrix}. \tag{77}$$

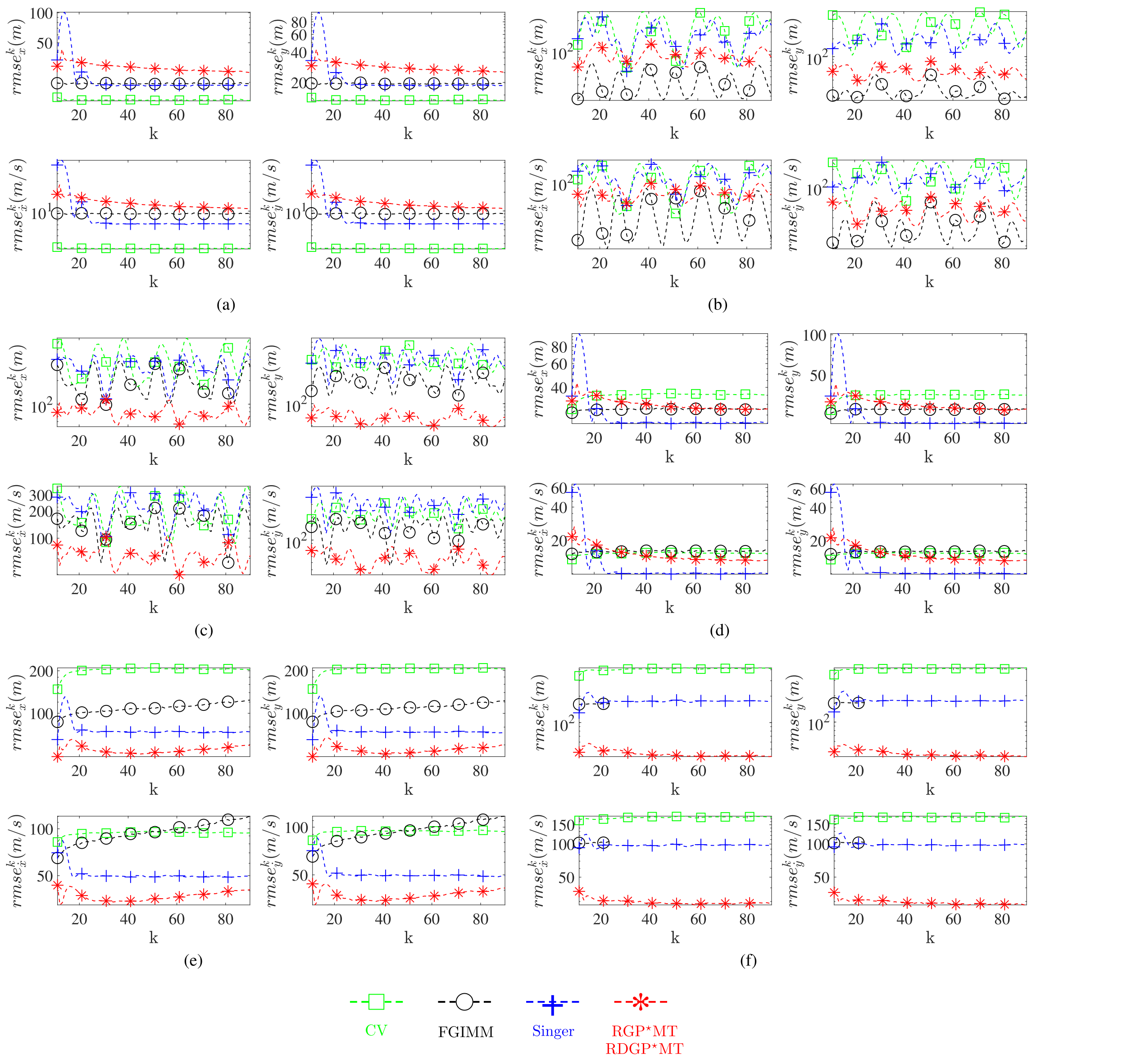

Fig. 3. *Prediction RMSE.* The figure shows the prediction performance in 10 000 Monte Carlo runs based on RMSE. An incomplete plot (such as Fixed Grid Interacting Multiple Model (FGIMM) in S6) means that the corresponding filter sometimes diverges after that time. The *Y*-axis is set to log scale for readability. (a) S1: CV. (b) S2: Gradual CT. (c) S3: Sharp CT. (d) S4: Singer lazy. (e) S5: Singer Agile. (f) S6: GP.

The cross-covariance matrix between the initial inducing points and the hyperparameters is set to a zero matrix $\hat{\boldsymbol{C}}_0^{f\theta} = \boldsymbol{O}$.

D. Results

The results are obtained from 10 000 Monte Carlo runs for each scenario. The accuracies of the prediction and filtering steps are evaluated separately. The graphical and numerical comparisons of the prediction process are given in Fig. 3 and Table II, respectively. It can be observed that the GPMT **(F4)**, the proposed RGP*MT **(F7)** and, a variation of the proposed approach, RGPMT **(F6)** have comparable performance in all scenarios. This shows that the *performance of real-time RGP*MT, using the inducing points and online learning, is as good as ML-based nonreal-time GPMT*. The CGPMT **(F5)** performs poorly during the scenarios **(S1)** and **(S5)**. This is due to the absence of learning in this filter. The RMSE increases with time, which is not a desirable property for filters. The filter FGIMM **(F2)** sometimes diverges in scenario **(S6)**, which is also not a desirable property. The

TABLE II
Prediction Mean RMSE (NAN Means Filter Diverged)

		F1	F2	F3	F4	F5	F6	F7
S1	x	13	19	24	22	41	17	28
	$\dot{x}$	3	9	10	10	31	5	13
S2	x	230	34	181	72	113	148	87
	$\dot{x}$	123	32	130	83	119	98	67
S3	x	365	208	303	69	71	82	79
	$\dot{x}$	203	150	224	82	79	55	66
S4	x	35	28	27	27	45	35	30
	$\dot{x}$	15	16	13	15	35	13	15
S5	x	203	109	77	46	164	69	55
	$\dot{x}$	94	96	51	40	167	40	37
S6	x	407	NAN	176	43	55	51	44
	$\dot{x}$	175	NAN	98	36	43	30	30

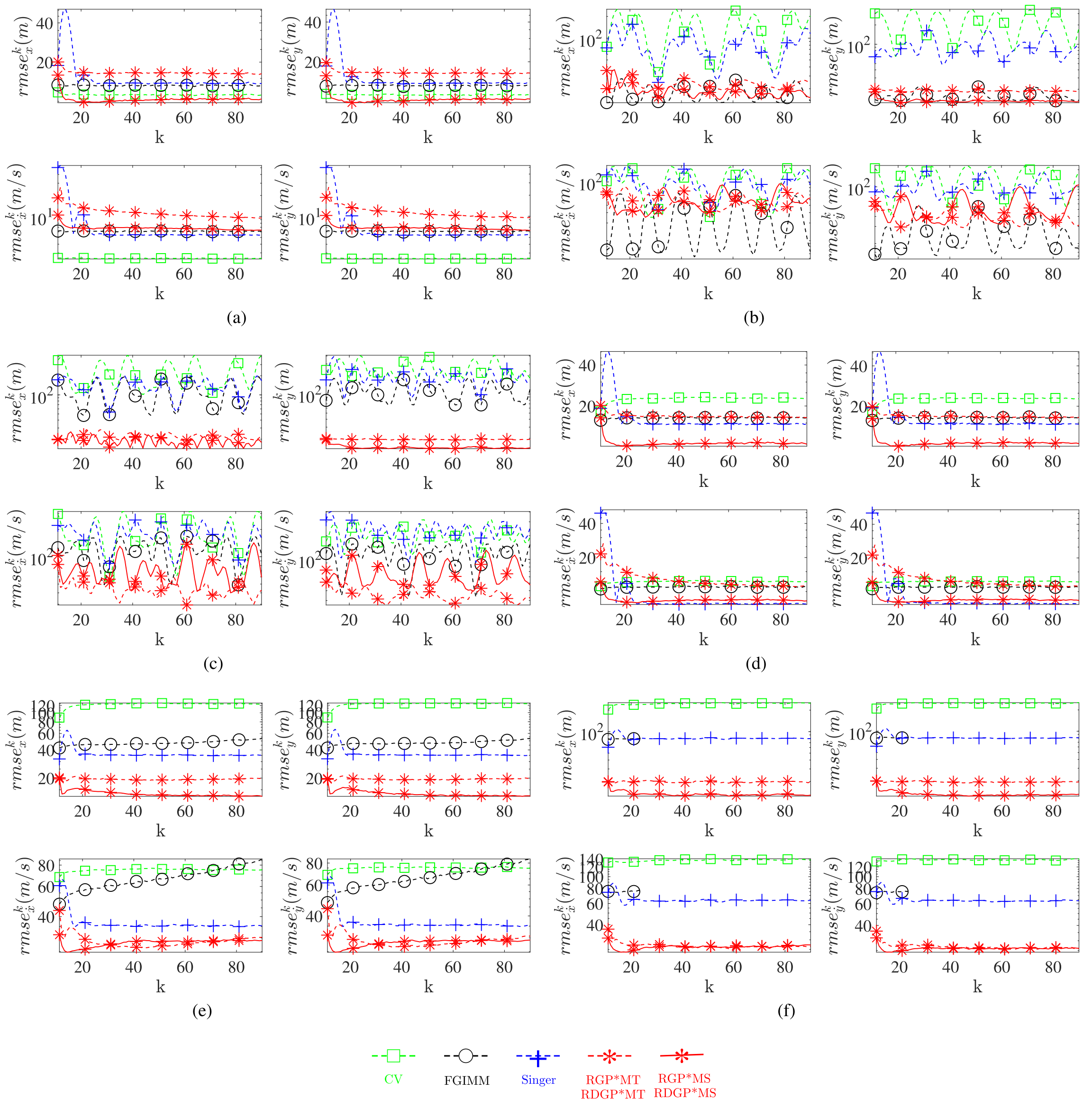

Fig. 4. *Filtering and Smoothing RMSE*. The figure shows the filtering and smoothing performance in 10 000 Monte Carlo runs based on RMSE. An incomplete plot (such as FGIMM in S6) means that the corresponding filter sometimes diverges. The *Y*-axis is set to log scale for readability. (a) S1: CV. (b) S2: Gradual CT. (c) S3: Sharp CT. (d) S4: Singer lazy. (e) S5: Singer Agile. (f) S6: GP.

Singer KF **(F3)** performance degrades during the scenarios **(S2)**, **(S3)**, and **(S6)**. The performance of the CV filter **(F1)** is not satisfactory for **(S2)**, **(S3)**, **(S5)**, and **(S6)**.

The graphical and numerical comparisons of the filtering process are given in Fig. 4 and Table III, respectively. A comparison similar to the prediction process can be done. It can be concluded that the *model-based filters and a constant hyperparameters based GPMT are not suitable for predicting a wide range of target trajectories*. Three highly maneuvering and mismatched scenarios, **(S3)**, **(S5)**, and **(S6)**, are considered to study the effects of the mismatch.

For scenario **(S3)**, the proposed approach provides a performance improvement of $\frac{22-109}{109} \times 100 = 80\%$ in position and $\frac{45-118}{118} \times 100 = 62\%$ in velocity filtering as compared to the best model-based filter **(F2)**. For scenario **(S5)**, the proposed approach provides a performance improvement

TABLE III
Filtering Mean RMSE (NAN Means Filter Diverged)

		F1	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**S*7**
S1	x	11	13	16	15	19	12	17	10
	$\dot{x}$	2	6	8	8	18	5	12	7
S2	x	140	18	83	20	30	49	23	19
	$\dot{x}$	106	25	98	38	61	87	48	48
S3	x	221	109	139	19	21	22	22	19
	$\dot{x}$	175	118	174	38	39	47	45	73
S4	x	23	16	17	16	20	22	17	11
	$\dot{x}$	12	11	10	10	20	13	13	8
S5	x	123	48	37	19	44	29	19	14
	$\dot{x}$	75	67	37	24	85	35	29	28
S6	x	247	NAN	81	18	23	22	20	13
	$\dot{x}$	137	NAN	64	26	30	29	27	26

of $\frac{19-37}{37} \times 100 = 49\%$ in position and $\frac{29-37}{37} \times 100 = 22\%$ in velocity filtering as compared to the best model-based filter **(F3)**.

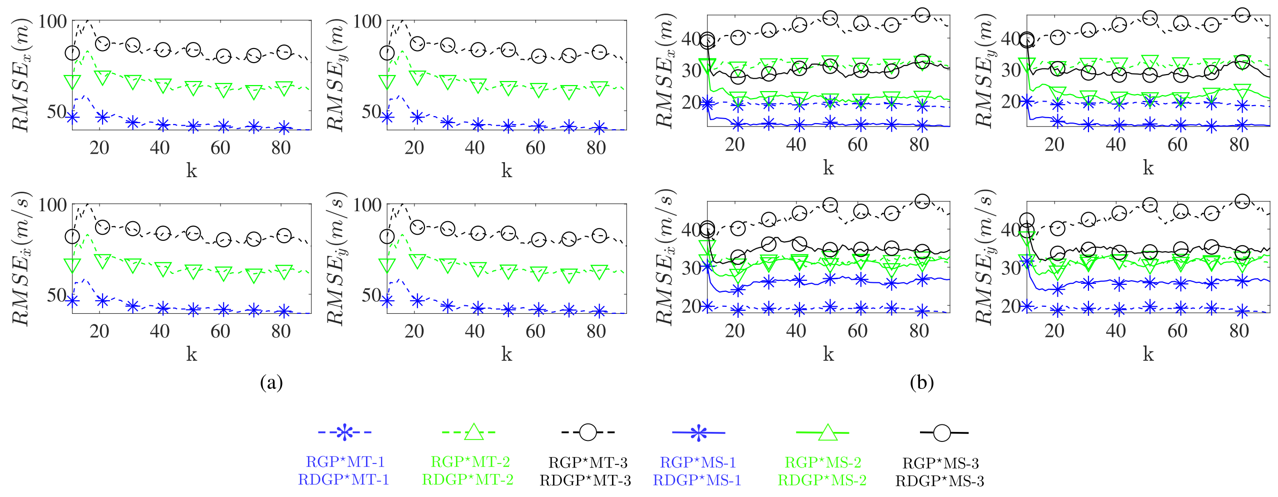

Fig. 5. *Performance versus measurement noise variance*. This figure shows the performance of the proposed approaches against different measurement noise variances. The left and right halves correspond, respectively, to the prediction and the filtering processes. The numerics after the name of the approach represents the scenarios, i.e., the postfix numeric 1 corresponds to 25 m standard deviation in measurement noise while 2 and 3 represent, respectively, 40 and 50 m standard deviations. (a) Prediction. (b) Filtering.

TABLE IV
Processing Time in Milliseconds

Method	Time (msec)
CGPMT + FOCGPMT	0.01
CV	0.05
Singer	0.07
FGIMM	0.5
RGP*MT + RDGP*MT + RGP*MS + RDGP*MS	5
GPMT + FO-GPMT	41
RGPMT + RDGPMT + RGPMS + RDGPMS	52

For scenario **(S6)**, the proposed approach provides a performance improvement of $\frac{20-81}{81} \times 100 = 75\%$ in position and $\frac{27-64}{64} \times 100 = 58\%$ in velocity filtering as compared to the best model-based filter **(F3)**. The smoothing performance is also demonstrated in Fig. 4 and Table III.

E. Processing Time

The program was run on MATLAB R2018b on a Windows 7 Home (64 bit) Laptop computer installed with an Intel(R) Core(TM) i3-M350 CPU @ 2.27GHz(2 CPUs) and 4GB RAM. The processing time of a single filter iteration averaged over 500 Monte Carlo runs is given in Table IV, sorted from fastest to slowest. Filters **(F4)** and **(F6)** take maximum time per iteration due to the time taken for ML-based learning. These are not suitable for real-time processing. Filters **(F1)**, **(F3)**, and **(F5)** are the fastest among the compared approaches. The combined processing time for the proposed filter and smoother is around 5ms. It is almost 10 times slower than **(F2)**. The processing time can be improved further by optimization of the code or by changing the platform to C++.

F. Impact of Increased Noise Variance on the Performance

A simulation based study on the performance of the proposed approaches with an increasing noise variance is given in this section for x coordinate. Testing scenario **(S6)** is considered for the evaluation. The results are obtained for three different measurement noise standard deviations, $\sigma_x = \{25, 40, 50\}$ m. The position and velocity RMSE for 5000 Monte Carlo runs of the three test scenarios are given in Fig. 5. It can be observed that the performance degrades as the measurement noise variance increases. It is important to note that no prior information regarding the change in the noise variance is provided to the filter and smoother. The proposed approaches adapt to the changing noise variance scenarios through recursive estimation of the variance and divergence is avoided. The percentage degradation in the prediction, filtering, and the smoothing with respect to the percentage increase in the noise variance is given in Table V. It can be observed that the percentage degradation in the position prediction, filtering, and smoothing is comparable to the percentage increase in the noise variance, e.g., a 100% increase in the noise variance degrades the position prediction by 92%. The percentage degradation in the velocity prediction, filtering, and smoothing is less as compared with the percentage degradation of the position estimate, e.g., a 100% increase in the noise variance degrades the velocity prediction by 42%.

TABLE V
Percentage Degradation

	% Increment	% degradation			
		Prediction		Filtering	
	σ_x	pos	vel	pos	vel
Filter	60	51	25	66	22
	100	92	42	127	40
Smoother	60	-	-	66	18
	100	-	-	130	33

V. CONCLUSION

Model-based approaches are well established and widely used for state estimation. This article shows that data-driven approaches can perform equally well and even better than model-based approaches. A novel GP regression approach with learning is proposed for tracking and smoothing purposes. Thanks to recursive learning of the hyperparameters of the GP, the approach is suitable for online estimation. The performance evaluation, carried out in challenging scenarios and compared with other online model-based filters, shows that the proposed approach gives overall the best performance for a wide set of target trajectories. For the two highly maneuverable and mismatched scenarios, **(S3)** CT and **(S5)** Singer, the proposed approach provides 80% and 62% performance improvement in the position estimates and 49% and 22% in the velocity estimates, respectively, as compared to the best model-based filter **(F3)**, i.e., a Singer motion model based KF. The flexibility provided by the proposed model-free approach strongly advocates its preference over model-based approaches for applications involving a wide set of target trajectories. Current work is focused on the performance of GP approaches to quantify the impact of uncertainties with information measures.

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor and the anonymous reviewers who helped to improve this paper.

REFERENCES

[1] Y. Bar-Shalom, T. E. Fortmann, and P. G. Cable
Tracking and Data Association. New York, NY, USA: Elsevier, 1990.

[2] S. Blackman and R. Popoli
Design and Analysis of Modern Tracking Systems. Norwood, MA, USA: Artech House, 1999.

[3] N. N. Kachouie and P. W. Fieguth
Extended-Hungarian-JPDA: Exact single-frame stem cell tracking
IEEE Trans. Biomed. Eng., vol. 54, no. 11, pp. 2011–2019, Nov. 2007.

[4] S. S. Blackman
Multiple-Target Tracking With Radar Applications
Dedham, MA, USA: Artech House, 1986, p. 463.

[5] Y. Bar-Shalom, P. K. Willett, and X. Tian
Tracking and Data Fusion. Storrs, CT, USA: YBS Publishing, 2011.

[6] Z.-M. Qian, X. E. Cheng, and Y. Q. Chen
Automatically detect and track multiple fish swimming in shallow water with frequent occlusion
PLoS One, vol. 9, no. 9, 2014, Art. no. e106506.

[7] K. Granström, A. Natale, P. Braca, G. Ludeno, and F. Serafino
Gamma Gaussian inverse Wishart probability hypothesis density for extended target tracking using X-band marine radar data
IEEE Trans. Geosci. Remote Sens., vol. 53, no. 12, pp. 6617–6631, Dec. 2015.

[8] A. Petrovskaya and S. Thrun
Model based vehicle detection and tracking for autonomous urban driving
Auton. Robots, vol. 26, no. 2/3, pp. 123–139, 2009.

[9] K. Wenzl, H. Ruser, and C. Kargel
Configuration of a sparse network of LiDAR sensors to identify security-relevant behavior of people
in *Proc. SPIE Unmanned/Unattended Sensors Sensor Netw.*, E. M. Carapezza, Ed., vol. 7480, 2009, pp. 48–57.

[10] L. Mihaylova, A. Y. Carmi, F. Septier, A. Gning, S. K. Pang, and S. Godsill
Overview of Bayesian sequential Monte Carlo methods for group and extended object tracking
Digit. Signal Process., vol. 25, pp. 1–16, 2014.

[11] K. Granström, M. Baum, and S. Reuter
Extended object tracking: Introduction, overview, and applications
J. Adv. Inf. Fusion, vol. 12, no. 2, pp. 139–174, 2017.

[12] D. Reid
An algorithm for tracking multiple targets
IEEE Trans. Autom. Control, vol. 24, no. 6, pp. 843–854, Dec. 1979.

[13] S. Oh, S. Russell, and S. Sastry
Markov chain Monte Carlo data association for general multiple-target tracking problems
in *Proc. 43 rd IEEE Conf. Decis. Control*, 2004, vol. 1, pp. 735–742.

[14] D. E. Clark
Multiple target tracking with the probability hypothesis density filter
Ph.D. dissertation, Edinburgh, U.K.: Heriot-Watt Univ., 2006.

[15] R. E. Kalman
A new approach to linear filtering and prediction problems
J. Basic Eng., vol. 82, no. 1, pp. 35–45, 1960.

[16] H. W. Sorenson
Kalman Filtering: Theory and Application. Piscataway, NJ, USA: IEEE Press, 1985.

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

[18] H. A. Blom and Y. Bar-Shalom
The interacting multiple model algorithm for systems with Markovian switching coefficients
IEEE Trans. Autom. Control, vol. 33, no. 8, pp. 780–783, Aug. 1988.

[19] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan
Interacting multiple model methods in target tracking: A survey
IEEE Trans. Aerosp. Electron. Syst., vol. 34, no. 1, pp. 103–123, Jan. 1998.

[20] R. A. Singer
Estimating optimal tracking filter performance for manned maneuvering targets
IEEE Trans. Aerosp. Electron. Syst., vol. AES-6, no. 4, pp. 473–483, Jul. 1970.

[21] W. Aftab and L. Mihaylova
A Gaussian process regression approach for point target tracking
in *Proc. 22nd Int. Conf. Inf. Fusion*, 2019, pp. 1–8.

[22] J. Hartikainen and S. Särkkä
Kalman filtering and smoothing solutions to temporal Gaussian process regression models
in *IEEE Int. Workshop Mach. Learn. Signal Process.*, 2010, pp. 379–384.

[23] C. E. Rasmussen and C. K. Williams
Gaussian Processes for Machine Learning, vol. 1, Cambridge, MA, USA: MIT Press, 2006.

[24] T. D. Barfoot, C. H. Tong, and S. Särkkä
Batch continuous-time trajectory estimation as exactly sparse Gaussian process regression
Robot., Sci. Syst., vol. 10, 2014, Paper A1.

[25] J. Dong, B. Boots, and F. Dellaert
Sparse Gaussian processes for continuous-time trajectory estimation on matrix lie groups
2017, *arXiv.1705.06020*.

[26] N. Wahlström and E. Özkan
Extended target tracking using Gaussian processes
IEEE Trans. Signal Process., vol. 63, no. 16, pp. 4165–4178, Aug. 2015.

[27] W. Aftab, A. De Freitas, M. Arvaneh, and L. Mihaylova
A Gaussian process approach for extended object tracking with random shapes and for dealing with intractable likelihoods
in *Proc. 22nd Int. Conf. Digit. Signal Process.*, 2017, pp. 1–5.

[28] M. Lázaro-Gredilla, S. Van Vaerenbergh, and N. D. Lawrence
Overlapping mixtures of Gaussian processes for the data association problem
Pattern Recognit., vol. 45, no. 4, pp. 1386–1395, 2012.

[29] M. F. Huber
Recursive Gaussian process: On-line regression and learning
Pattern Recognit. Lett., vol. 45, pp. 85–91, 2014.

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

[31] X. R. Li and V. P. Jilkov
Survey of maneuvering target tracking. Part I. Dynamic models
IEEE Trans. Aerosp. Electron. Syst., vol. 39, no. 4, pp. 1333–1364, Nov. 2003.

[32] O. Rudovic, M. Pantic, and I. Patras
Coupled Gaussian processes for pose-invariant facial expression recognition
IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 6, pp. 1357–1369, Jun. 2013.

[33] P. Boyle
Gaussian processes for regression and optimisation
Victoria Univ. Wellington, Wellington, New Zealand, 2007.

[34] P. S. Swain *et al.*
Inferring time derivatives including cell growth rates using Gaussian processes
Nature Commun., 2016, vol. 7, Art. no. 13766.

[35] F. Beutler, M. F. Huber, and U. D. Hanebeck
Gaussian filtering using state decomposition methods
in *Proc. 12th Int. Conf. Inf. Fusion*, 2009, pp. 579–586.

[36] R. Kandepu, L. Imsland, and B. A. Foss
Constrained state estimation using the unscented Kalman filter
in *Proc. 16th Mediterranean Conf. Control Autom.*, 2008, pp. 1453–1458.

[37] S. Reece and S. Roberts
An introduction to Gaussian processes for the Kalman filter expert
in *Proc. 13th Int. Conf. Inf. Fusion*, 2010, pp. 1–9.

[38] A. J. Smola and P. L. Bartlett
Sparse greedy Gaussian process regression
in *Proc. Adv. Neural Inf. Process. Syst.*, 2001, pp. 619–625.

[39] C. Veibäck, J. Olofsson, T. R. Lauknes, and G. Hendeby
Learning target dynamics while tracking using Gaussian processes
IEEE Trans. Aerosp. Electron. Syst., vol. 56, no. 4, pp. 2591–2602, Aug. 2020.

[40] E. Snelson and Z. Ghahramani
Sparse Gaussian processes using pseudo-inputs
in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 1257–1264.

[41] H.-M. Kim, B. K. Mallick, and C. Holmes
Analyzing nonstationary spatial data using piecewise Gaussian processes
J. Amer. Statist. Assoc., vol. 100, no. 470, pp. 653–668, 2005.

[42] H. Liu, Y. Ong, X. Shen, and J. Cai
When Gaussian process meets big data: A review of scalable GPs
IEEE Trans. Neural Netw. Learn. Syst., pp. 1–19, 2020.

[43] X. R. Li and V. P. Jilkov
Survey of maneuvering target tracking. Part V. Multiple-model methods
IEEE Trans. Aerosp. Electron. Syst., vol. 41, no. 4, pp. 1255–1321, Oct. 2005.

[44] A. Munir and D. Atherton
Maneuvring target tracking using different turn rate models in the interacting multiple model algorithm
in *Proc. 34th IEEE Conf. Decis. Control*, 1995, vol. 3, pp. 2747–2751.

Waqas Aftab received the B.Eng. degree in avionics from the College of Aeronautical Engineering (NUST), Risalpur, Pakistan, in 2005, and the M.S. degree in control systems engineering from Air University, Islamabad, Pakistan, in 2014.

He is a Ph.D. Research Student with The University of Sheffield, Sheffield, U.K. His main interests are multisensor multitarget tracking and Gaussian processes.

Lyudmila Mihaylova (Senior Member, IEEE) received the M.Eng. degree in systems and control engineering, the M.Sc. degree in applied mathematics and informatics, and the Ph.D. degree in systems and control engineering from the Technical University of Sofia, Bulgaria.

She is currently a Professor of Signal Processing and Control with the Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, U.K. Her research interests include machine learning and autonomous systems with various applications such as navigation, surveillance, and sensor network systems. She is an Associate Editor for the IEEE TRANSACTIONS ON AEROSPACE AND ELECTRONIC SYSTEMS and for the *Elsevier Signal Processing Journal*. She was the President of the International Society of Information Fusion (ISIF) in the period 2016–2018. She is on the Board of Directors of ISIF. She was the General Vice-Chair for the International Conference on Information Fusion 2018 (Cambridge, U.K.), of the IET Data Fusion and Target Tracking 2014 and 2012 Conferences, Program Co-Chair for the 19th International Conference on Information Fusion 2020, 2016, and others.