IEEE Open Journal of Signal Processing

Received 16 September 2025; revised 9 December 2025; accepted 11 December 2025. Date of publication 19 December 2025; date of current version 16 January 2026. The review of this article was arranged by Associate Editor John Lipor.

Digital Object Identifier 10.1109/OJSP.2025.3646127

Multiple Model Recursive Gaussian Process for Robust Target Tracking

ALI EMRE BALCI (Graduate Student Member, IEEE) AND RAJ THILAK RAJAN (Senior Member, IEEE)

[1]Signal Processing Systems, Faculty of EEMCS, Delft University of Technology, 2628CD Delft, The Netherlands

CORRESPONDING AUTHOR: ALI EMRE BALCI (email: a.e.balci@tudelft.nl).

This work was supported by Sensor AI Lab through the AI Labs Program of the Delft University of Technology, and the RVO (Missie gedreven Onderzoek, Ontwikkeling en Innovatie (MOOI)) Aerial Quadcopter Units for Aquatic Flow Investigation and Nautical Data (AQUAFIND) Project

ABSTRACT Accurate tracking of targets is vital for safe and reliable operations, particularly in complex and dynamic environments such as urban areas. Traditional tracking methods, including Kalman and particle filters, often perform poorly in real world scenarios, due to inaccurate models and sparse or noisy measurements. Gaussian process (GP) based methods offer a flexible and data driven alternative with uncertainty quantification that does not depend on predefined dynamical equations. However, state of the art GP tracking approaches require expensive hyperparameter optimization, which limits their practicality for real time applications. In this work, we introduce a novel GP mixture based computationally efficient tracking method, which is capable of modeling complex system behavior and adapt to changing dynamics. Our proposed solution, named Multiple Model Recursive Gaussian Process (MM-RGP), adapts continuously to changing dynamics, is capable of modeling complex behavior, and is robust against sparse observation. In addition, the proposed method avoids hyperparameter optimization and adapts to incoming data. We demonstrate the effectiveness of our solution using the example of uncrewed aerial vehicle (UAV) tracking, with both simulated and real datasets, and propose directions for extending our work.

INDEX TERMS Gaussian process, target tracking, online learning, adaptive filtering.

I. INTRODUCTION

Target tracking is a fundamental problem in a wide variety of autonomous systems, where accurate localization is critical for safe and efficient operation [1], [2]. Tracking remains a significant challenge, particularly for systems that exhibit highly complex motion or when only sparse observations are available [3], [4], [5]. One of the prominent case is that of UAVs, which are used in numerous application areas including aerial surveillance, agriculture, disaster response, and logistics [6], [7]. Various tracking solutions have been developed employing a wide range of sensors such as ADS-B [5], [8], radar [9], and GPS [10], contributing to better airspace awareness, collision avoidance, and autonomous flight. Nevertheless, the complex and dynamic behaviors of UAVs, combined with sparse or intermittent observations from real-world sensing systems, continue to pose significant challenges for traditional tracking algorithms.

Conventional tracking techniques, such as Kalman filters (KF) and particle filters (PF), are widely used in applications [11], which typically rely on accurate system models to propagate state estimates. In practice, target dynamics are often influenced by unpredictable environmental disturbances, such as wind gusts [12], which can cause model mismatch and lead to poor tracking performance. This issue is especially apparent in complex settings where system behavior deviates from expected models. Urban environments introduce additional challenges for tracking [13], e.g., signal occlusion, sensor limitations, and degraded GPS signal availability often result in sparse and noisy measurements. In such conditions, model based approaches such as KF and PF may fail to maintain reliable state estimates.

Data driven methods have recently gained traction in various fields for inference and control primarily because of their flexibility and robustness, especially with GPs emerging as a promising tool for modeling complex system behavior [14]. GPs offer a nonparametric Bayesian framework that allows for learning arbitrary functions without requiring an explicit dynamical model, making them well suited for unpredictable

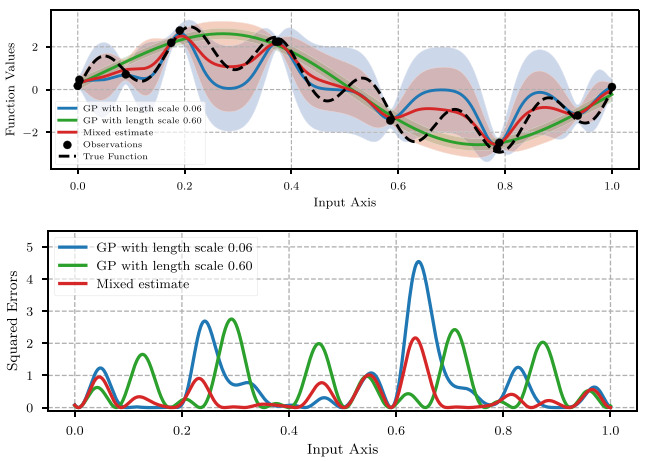

FIGURE 1. Function estimations and the true function (top), and pointwise errors for the corresponding estimated functions (bottom).

and nonlinear target motion [15], [16]. Recent works on GP based learning and target tracking have shown reliable performance across a variety of scenarios, including multi target tracking [17], [18], [19], distributed architectures [20], [21] and generalized non-Gaussian process models [22]. Although valuable, these approaches often rely on manual or computationally expensive hyperparameter tuning, which limits their scalability and use in real world settings. Recently emerged sampling based GP methods [23], [24], [25] became popular for improved tracking performance, however, they tend to suffer from curse of dimensionality, a common problem for sampling based solutions. In the existing literature there is a growing need for improved methods that are not only reliable, scalable, and suitable for onboard computing, but also adaptable to changes in system dynamics due to environmental factors or external disturbances.

In this work, we present a GP mixture based tracking method that addresses these limitations by eliminating the need for hyperparameter tuning and enabling online adaptation through incoming data. Rather than relying on known transition models, we present a robust and computationally efficient framework for estimating the trajectory under sparse observation conditions. Our approach combines the flexibility of GPs with improved computational efficiency, making it well suited for real-time tracking in complex, dynamic, and resource constrained environments. Compared to relatively complex multiple model GP methods [26], [27], [28], [29], our solution is simpler and more practical for real world target tracking tasks. Our main contributions in this work are as follows:

- We propose a flexible model with a GP based target tracking approach that can capture irregular yet smooth dynamic patterns.
- We introduce a scalable GP regression formulation that is readily applicable to real-time target tracking applications for potential edge based implementation.
- We adopt a data driven multiple model online kernel learning approach for improved robustness and adaptability.
- We conduct experiments to compare our solution with the state of the art tracking approaches, i.e., [16] and [30], and show the effectiveness of our approach.

Notation: Scalars are denoted by regular lowercase letters, e.g., x, bold lowercase letters represent vectors, e.g., $\boldsymbol{x}$, bold uppercase letters, e.g., $\boldsymbol{X}$ denote matrices, and calligraphic letters denote sets, e.g., $\mathcal{X}$. A tuple of variables is denoted using parentheses, e.g., (x, y). A sequence of variables is denoted with curly braces, e.g., $\mathcal{X} = \{\boldsymbol{x}_n\}_{n=1}^{N}$ where n indexes elements from 1 to N, and a subsequence is denoted as $\mathcal{X}_{i:j} = \{\boldsymbol{x}_n\}_{n=i}^{j}$ for $1 \leq i \leq j \leq N$. The ith entry of a vector $\boldsymbol{x}$ is denoted by $[\boldsymbol{x}]_i$, the (i, j)th entry of a matrix $\boldsymbol{X}$ is denoted by $[\boldsymbol{X}]_{ij}$. An $N \times N$ identity matrix is written as $\boldsymbol{I}_N$, and equivalence is denoted by $\triangleq$. Probability density functions are generally denoted by $p(\cdot)$. The multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is written as $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and its density evaluated at $\boldsymbol{x}$ is denoted by $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$.

II. PRELIMINARIES

In this section, we introduce the essential background on GP regression, laying the groundwork for the methods employed in our proposed approach. We begin with an overview of standard GP regression, followed by recursive and multiple model extensions of GP regression. These concepts will be extensively utilized and built upon in the development of the proposed approach.

A. GP REGRESSION

A GP defines a probability distribution over functions, i.e., $f : \mathcal{S} \to \mathbb{R}$ for a given domain $\mathcal{S} \subseteq \mathbb{R}^D$, such that for any finite set $\{\boldsymbol{x}_n\}_{n=1}^{N}$ of N inputs, the joint distribution of function values $\{f(\boldsymbol{x}_n) \mid \boldsymbol{x}_n \in \mathcal{S}\}_{n=1}^{N}$ follows a multivariate Gaussian distribution [31]. This distribution can be fully specified by a mean function $\mu : \mathcal{S} \to \mathbb{R}$ and a kernel $\kappa : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$, and thus we write $f \sim \mathcal{GP}(\mu, \kappa)$ to denote this overall distribution. Without loss of generality, throughout this manuscript we assume that $\mu(t) = 0$. Consider a dataset $\mathcal{D}_{1:N} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$, where for a given set of inputs $\mathcal{X} \triangleq \{\boldsymbol{x}_n \mid \boldsymbol{x}_n \in \mathcal{S}\}_{n=1}^{N}$ we observe noisy realizations

$$y_n = f(\boldsymbol{x}_n) + \epsilon_n \qquad \epsilon_n \sim \mathcal{N}(0, \sigma_\epsilon^2), \tag{1}$$

where the unknown function, modeled as a GP, can be learned by choosing an appropriate kernel. For illustration, consider the widely used squared exponential (SE) kernel

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j \mid \boldsymbol{\theta}) = \sigma_s^2 \exp\left(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}{2\ell^2}\right), \tag{2}$$

where the hyperparameters, i.e., signal variance σ_s^2 and length scale ℓ, determine the general behavior of the underlying function. The hyperparameters $\boldsymbol{\theta} \triangleq \begin{bmatrix}\sigma_s^2, & \ell\end{bmatrix}^T$ can be tuned to best fit the data $[\boldsymbol{y}]_i \triangleq y_i$ by solving the following optimization

problem

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{y} \mid \mathcal{X}, \boldsymbol{\theta}),$$
$$= \arg\min_{\boldsymbol{\theta}} \left[\boldsymbol{y}^T \boldsymbol{\Gamma}^{-1} \boldsymbol{y} + \log\det(\boldsymbol{\Gamma})\right], \tag{3}$$

where $\boldsymbol{\Gamma} \triangleq \boldsymbol{K}_{XX} + \sigma_\epsilon^2 \boldsymbol{I}_N$ and $[\boldsymbol{K}_{XX}]_{ij} \triangleq \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Now, given the dataset $\mathcal{D}_{1:N}$ and a trained kernel, the predictive distribution at some known test locations $\mathcal{X}^* \triangleq \{\boldsymbol{x}_n^* \mid \boldsymbol{x}_n^* \in \mathcal{S}\}_{n=1}^{\bar{N}}$ can be defined by the joint posterior distribution $\boldsymbol{f}_N \triangleq \boldsymbol{f} \mid \mathcal{D}_{1:N} \sim \mathcal{N}(\hat{\boldsymbol{f}}_N, \boldsymbol{\Sigma}_N)$ of the vector $[\boldsymbol{f}]_n \triangleq f(\boldsymbol{x}_n^*)$, which can be computed in closed form [32, Ch. 15] as

$$\hat{\boldsymbol{f}}_N = \boldsymbol{K}_{X*}^T \boldsymbol{\Gamma}^{-1} \boldsymbol{y}, \tag{4a}$$
$$\boldsymbol{\Sigma}_N = \boldsymbol{K}_{**} - \boldsymbol{K}_{X*}^T \boldsymbol{\Gamma}^{-1} \boldsymbol{K}_{X*}, \tag{4b}$$

where $[\boldsymbol{K}_{X*}]_{ij} \triangleq \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j^*)$ and $[\boldsymbol{K}_{**}]_{ij} \triangleq \kappa(\boldsymbol{x}_i^*, \boldsymbol{x}_j^*)$.

B. RECURSIVE GP REGRESSION

Observe that the computation of both the GP hyperparameter tuning (3) and the GP posterior distribution (4) involves N dimensional matrix inversions, leading to computational complexities of $O(N^3)$, which makes GP regression impractical for large datasets or real time applications. To address this issue Huber [33] proposed a sequential, yet approximate, algorithm called recursive GP (RGP) regression that processes each observation $(\boldsymbol{x}_n, y_n)$ individually. Let $\boldsymbol{f}_{k-1} \triangleq \boldsymbol{f} \mid \mathcal{D}_{1:k-1} \sim \mathcal{N}(\hat{\boldsymbol{f}}_{k-1}, \boldsymbol{\Sigma}_{k-1})$ be the function estimate obtained after processing observations until time k, where $1 \leq k \leq N$. Then, to incorporate a new observation y_k at $\boldsymbol{x}_k$, we apply the widely known Markov assumption [34], [35], [36] to approximate,

$$p(y_k \mid \hat{\boldsymbol{f}}_{k-1}, \boldsymbol{\Sigma}_{k-1}, \boldsymbol{x}_k, \mathcal{D}_{1:k-1}) \approx p(y_k \mid \hat{\boldsymbol{f}}_{k-1}, \boldsymbol{\Sigma}_{k-1}, \boldsymbol{x}_k), \tag{5}$$

which is equivalent to the linear observation model [33]

$$y_k = \boldsymbol{h}_k^T \boldsymbol{f}_{k-1} + \epsilon_k, \ \epsilon_k \sim \mathcal{N}(0, \sigma_k^2), \tag{6a}$$
$$\boldsymbol{h}_k \triangleq \boldsymbol{K}_{**}^{-1} \boldsymbol{\kappa}_{*k}, \tag{6b}$$
$$\sigma_k^2 \triangleq \kappa(\boldsymbol{x}_k, \boldsymbol{x}_k) + \sigma_\epsilon^2 - \boldsymbol{\kappa}_{*k}^T \boldsymbol{K}_{**}^{-1} \boldsymbol{\kappa}_{*k}, \tag{6c}$$

where $[\boldsymbol{\kappa}_{*k}]_i \triangleq \kappa(\boldsymbol{x}_i^*, \boldsymbol{x}_k)$. See Appendix A for a detailed derivation. With this formulation, the posterior density $\boldsymbol{f}_k \sim \mathcal{N}(\hat{\boldsymbol{f}}_k, \boldsymbol{\Sigma}_k)$ can be calculated using the update equations

$$\hat{\boldsymbol{f}}_k = \hat{\boldsymbol{f}}_{k-1} + \boldsymbol{g}_k (y_k - \hat{y}_k), \tag{7a}$$
$$\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}_{k-1} - \boldsymbol{g}_k \boldsymbol{h}_k^T \boldsymbol{\Sigma}_{k-1}, \tag{7b}$$

where,

$$\hat{y}_k \triangleq \boldsymbol{h}_k^T \hat{\boldsymbol{f}}_{k-1}, \tag{8a}$$
$$\boldsymbol{g}_k \triangleq \boldsymbol{\Sigma}_{k-1} \boldsymbol{h}_k s_k^{-1}, \tag{8b}$$
$$s_k \triangleq \boldsymbol{h}_k^T \boldsymbol{\Sigma}_{k-1} \boldsymbol{h}_k + \sigma_k^2. \tag{8c}$$

Here, $\hat{y}_k$, $\boldsymbol{g}_k$, s_k and $\boldsymbol{h}_k^T$ can be interpreted as the predicted observation, gain, innovation variance and observation vector, respectively. Note that with this approach we can eliminate the computational complexity $O(N^3)$ by precomputing $\boldsymbol{K}_{**}^{-1}$ and only inverting the scalar s_k.

C. MULTIPLE MODEL RECURSIVE GP REGRESSION (MM-RGP)

Extending the classical GP regression in Section II-A to a multiple model framework [26], [27], let $\mathcal{K} \triangleq \{\kappa^{(m)}\}_{m=1}^M$ denote the set of candidate kernel functions, where each $\kappa^{(m)} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ represents a distinct behavior of the true function. In contrast to conventional GP, our goal is now to learn the true function with elements of $\mathcal{K}$ without tuning the kernel hyperparameters. To this end the observation log-likelihood in (3) can be modified into an integer programming problem to find the most suitable kernel index $m \in \mathcal{M} \triangleq \{1, 2, \ldots, M\}$ rather than directly tuning the hyperparameters $\boldsymbol{\theta}$ as

$$\hat{m} = \arg\max_{m \in \mathcal{M}} \log p(\boldsymbol{y} \mid \mathcal{X}, m),$$
$$= \arg\min_{m \in \mathcal{M}} \left[\boldsymbol{y}^T (\boldsymbol{\Gamma}^{(m)})^{-1} \boldsymbol{y} + \log\det(\boldsymbol{\Gamma}^{(m)})\right], \tag{9}$$

where $\boldsymbol{\Gamma}^{(m)} \triangleq \boldsymbol{K}_{XX}^{(m)} + \sigma_\epsilon^2 \boldsymbol{I}_N$. Here the model index $\hat{m} \in \mathcal{M}$ represents the kernel function $\kappa^{(\hat{m})} \in \mathcal{K}$. Following (4), for each $\kappa^{(m)} \in \mathcal{K}$, a distinct mean and covariance can be obtained for the posterior distributions $\boldsymbol{f} \mid \mathcal{D}_{1:N}, \kappa^{(m)} \triangleq \boldsymbol{f}_N^{(m)} \sim \mathcal{N}(\hat{\boldsymbol{f}}_N^{(m)}, \boldsymbol{\Sigma}_N^{(m)})$, whose moments are

$$\hat{\boldsymbol{f}}_N^{(m)} = (\boldsymbol{K}_{X*}^{(m)})^T \left(\boldsymbol{K}_{XX}^{(m)} + \sigma_\epsilon^2 \boldsymbol{I}_N\right)^{-1} \boldsymbol{y}, \tag{10a}$$
$$\boldsymbol{\Sigma}_N^{(m)} = \boldsymbol{K}_{**}^{(m)} - (\boldsymbol{K}_{X*}^{(m)})^T \left(\boldsymbol{K}_{XX}^{(m)} + \sigma_\epsilon^2 \boldsymbol{I}_N\right)^{-1} \boldsymbol{K}_{X*}^{(m)}, \tag{10b}$$

where $[\boldsymbol{K}_{**}^{(m)}]_{ij} \triangleq \kappa^{(m)}(\boldsymbol{x}_i^*, \boldsymbol{x}_j^*)$, $[\boldsymbol{K}_{X*}^{(m)}]_{ij} \triangleq \kappa^{(m)}(\boldsymbol{x}_i, \boldsymbol{x}_j^*)$ and $[\boldsymbol{K}_{XX}^{(m)}]_{ij} \triangleq \kappa^{(m)}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. We expect each kernel candidate $\kappa^{(m)}$ to produce a different posterior distribution, depending on the kernel type and the hyperparameters. Some kernels will naturally perform better than others, and this difference can be quantified to select the "best" kernel.

An alternative approach is to employ a weighted average of the probability densities in (10), where the weights are determined based on the data [28], [29], [37], [38]. Consequently, we consider a posterior density of the form

$$p(\bar{\boldsymbol{f}}_N) = \sum_m \omega^{(m)} \mathcal{N}(\bar{\boldsymbol{f}}_N; \hat{\boldsymbol{f}}_N^{(m)}, \boldsymbol{\Sigma}_N^{(m)}), \tag{11}$$

where $\bar{\boldsymbol{f}}_N \triangleq \boldsymbol{f} \mid \mathcal{D}_{1:N}, \mathcal{K}$ the weights satisfy $\sum_m \omega^{(m)} = 1$ to ensure that the resulting posterior is a valid probability distribution. This weighted sum no longer describes a Gaussian distribution, instead a Gaussian mixture, but the distribution can be approximated as a Gaussian $\bar{\boldsymbol{f}}_N \sim \mathcal{N}(\hat{\bar{\boldsymbol{f}}}_N, \bar{\boldsymbol{\Sigma}}_N)$ with a mean and covariance [32, Ch. 11] as follows

$$\hat{\bar{\boldsymbol{f}}}_N = \sum_m \omega^{(m)} \hat{\boldsymbol{f}}_N^{(m)}, \tag{12a}$$
$$\bar{\boldsymbol{\Sigma}}_N = \sum_m \omega^{(m)} \left(\boldsymbol{\Sigma}_N^{(m)} + \Psi\left(\hat{\bar{\boldsymbol{f}}} - \hat{\boldsymbol{f}}_N^{(m)}\right)\right), \tag{12b}$$

where $\Psi : \mathbb{R}^{\bar{N}} \times \mathbb{R}^{\bar{N}} \to \mathbb{R}^{\bar{N}\times\bar{N}}$ denotes the outer product of a vector with itself

$$\Psi(\boldsymbol{v}) \triangleq \boldsymbol{v}\boldsymbol{v}^T. \tag{13}$$

Furthermore, weights can be assigned by using observation log-likelihoods, giving larger weights to kernels whose log-likelihood is closer to zero, for instance

$$\omega^{(m)} = \frac{(\log p(\mathbf{y} \mid \mathcal{X}, \kappa^{(m)}))^{-1}}{\sum_j (\log p(\mathbf{y} \mid \mathcal{X}, \kappa^{(j)}))^{-1}}, \tag{14}$$

which enables the framework to span a broader function space in contrast to each individual kernel candidate in $\mathcal{K}$.

To illustrate the idea, Fig. 1 shows the use of $M = 2$ models equipped with SE kernels in (2) with different length scales. As expected, the kernel with the larger length scale captures low frequency components of the true function, whereas the one with the smaller length scale adapts to higher frequencies. According to (9), the model with the smaller length scale fits the data best, even though it appears to be overfitted. In contrast, the mixture estimate, weighted via (14), strikes a balance between the two kernels and offsets the errors of one by leaning more on the other.

III. MULTIPLE MODEL RECURSIVE GP (MM-RGP)

Multiple model GP regression enhances model capacity and expressivity as compared to conventional GP approaches, while RGP reduces computational complexity by dynamically updating the moments of the posterior distribution for new observations. In this section, we seek to combine the strengths of both frameworks and introduce the Multiple Model Recursive Gaussian Process (MM-RGP) regression for online estimation and adaptive multiple model learning. Inspired by the popular Interacting Multiple Model (IMM) filter [39], MM-RGP seeks to strike a balance between computational efficiency and improved modeling flexibility.

A. MULTIPLE MODEL RECURSIVE GP REGRESSION

We begin by combining the observation model in (6) with a random walk model and propose a state space model, for each kernel function $\kappa^{(i)} \in \mathcal{K}$, where $i \in \mathcal{M}$, i.e.,

$$\boldsymbol{f}^{(i)}_{0|0} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{K}^{(i)}_{**}), \tag{15a}$$

$$\boldsymbol{f}^{(i)}_{k+1} = \boldsymbol{f}^{(i)}_{k} + \boldsymbol{\eta}^{(i)}_{k}, \tag{15b}$$

$$y_{k+1} = (\boldsymbol{h}^{(i)}_{k+1})^T \boldsymbol{f}^{(i)}_{k+1} + \nu^{(i)}_{k+1}, \tag{15c}$$

where $\boldsymbol{f}^{(i)}_{0|0}$ is the prior, $\boldsymbol{\eta}^{(i)}_{k} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}^{(i)}_{k})$ is the process noise modeling possible variations in the true function over time and $\nu^{(i)}_{k+1} \sim \mathcal{N}(0, (\sigma^{(i)}_{k+1})^2)$ is the observation noise associated with the ith filter. Observe that, unlike conventional adaptive filters, we do not depend on predefined dynamical models of the target. Instead, we represent the state vector as a collection of positions across multiple time instances and update the trajectory estimate as a whole with incoming data, allowing us to accommodate complex dynamics more readily.

Consider a discrete random variable $m_k \in \mathcal{M}$ representing the model index at time k, which allows the system to switch between different models over time. We then aim to infer and update its distribution at each time step. To this end we maintain a probability mass function over all possible models, where $\omega^{(i)}_k \triangleq p(m_k = i)$ denotes the probability that the model indexed as i being active at time k. In the absence of data, the model probabilities evolve over time according to a Markov chain, i.e.,

$$\boldsymbol{\omega}_{k+1} = \boldsymbol{\Xi}\boldsymbol{\omega}_k, \tag{16}$$

where the vector $\boldsymbol{\omega}_k \triangleq [\omega^{(1)}_k, \omega^{(2)}_k, \ldots, \omega^{(M)}_k]^T \in \mathbb{R}^M$ consists of model probabilities (or weights), and $\boldsymbol{\Xi}$ is a fixed right stochastic matrix [40] which obeys the constraint $\mathbf{1} = \boldsymbol{\Xi}\mathbf{1}$ where $\mathbf{1}$ represents a vector of ones of appropriate size. The elements of the stochastic matrix then represent the probabilities of transitions between models at each time step, i.e., $[\boldsymbol{\Xi}]_{ij} \triangleq p(m_{k+1} = i \mid m_k = j)$.

Now, to obtain the state estimate within this framework, we combine the outputs of the filters associated with each state space model (15). Before each filter update the most recent estimates are combined into Gaussian mixtures

$$p(\bar{\boldsymbol{f}}^{(i)}_k) = \sum_{j=1}^{M} \gamma^{(ij)}_k \mathcal{N}(\bar{\boldsymbol{f}}^{(i)}_k; \hat{\boldsymbol{f}}^{(j)}_{k|k}, \boldsymbol{\Sigma}^{(j)}_{k|k}), \ \forall i \in \mathcal{M}, \tag{17}$$

where $\hat{\boldsymbol{f}}^{(j)}_{k|k}$ and $\boldsymbol{\Sigma}^{(j)}_{k|k}$ denote the posterior mean and covariance of the filter j at time k respectively, and the mixture probabilities $\gamma^{(ij)}_k$ can be calculated as

$$\gamma^{(ij)}_k \triangleq \frac{\omega^{(j)}_k [\boldsymbol{\Xi}]_{ij}}{\sum_{l=1}^{M} \omega^{(l)}_k [\boldsymbol{\Xi}]_{il}}. \tag{18}$$

The resulting distributions are approximated as Gaussian distributions $\bar{\boldsymbol{f}}^{(i)}_k \sim \mathcal{N}(\hat{\bar{\boldsymbol{f}}}^{(i)}_k, \bar{\boldsymbol{\Sigma}}^{(i)}_k)$, whose moments are

$$\hat{\bar{\boldsymbol{f}}}^{(i)}_k = \sum_{j=1}^{M} \gamma^{(ij)}_k \hat{\boldsymbol{f}}^{(j)}_{k|k}, \tag{19a}$$

$$\bar{\boldsymbol{\Sigma}}^{(i)}_k = \sum_{j=1}^{M} \gamma^{(ij)}_k \left(\boldsymbol{\Sigma}^{(j)}_{k|k} + \Psi(\hat{\boldsymbol{f}}^{(j)}_{k|k} - \hat{\bar{\boldsymbol{f}}}^{(i)}_k) \right), \tag{19b}$$

and then transferred to the filters corresponding to their model indices i as inputs. Each filter then updates its posterior moments as

$$\hat{\boldsymbol{f}}^{(i)}_{k+1|k} = \hat{\bar{\boldsymbol{f}}}^{(i)}_k, \tag{20a}$$

$$\boldsymbol{\Sigma}^{(i)}_{k+1|k} = \bar{\boldsymbol{\Sigma}}^{(i)}_k + \boldsymbol{Q}^{(i)}_k, \tag{20b}$$

$$\hat{\boldsymbol{f}}^{(i)}_{k+1|k+1} = \hat{\boldsymbol{f}}^{(i)}_{k+1|k} + \boldsymbol{g}^{(i)}_{k+1} \left(y_{k+1} - \hat{y}^{(i)}_{k+1} \right), \tag{20c}$$

$$\boldsymbol{\Sigma}^{(i)}_{k+1|k+1} = \boldsymbol{\Sigma}^{(i)}_{k+1|k} - \boldsymbol{g}^{(i)}_{k+1} (\boldsymbol{h}^{(i)}_{k+1})^T \boldsymbol{\Sigma}^{(i)}_{k+1|k}, \tag{20d}$$

where $\boldsymbol{h}^{(i)}_{k+1}$, $\hat{y}^{(i)}_{k+1}$ and $\boldsymbol{g}^{(i)}_{k+1}$ can be calculated using (8). Intermediate quantities computed during the filter update step as

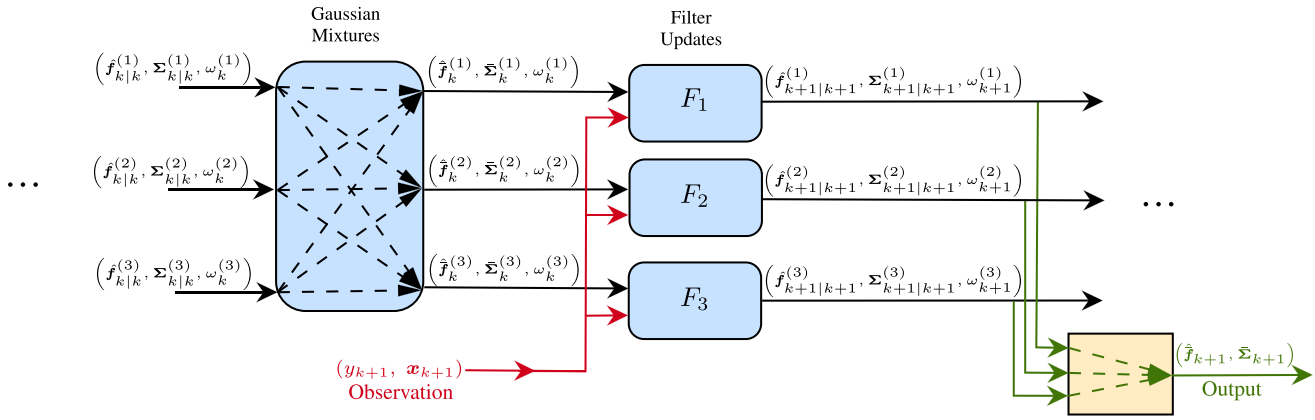

FIGURE 2. A graphical illustration of the proposed algorithm with $M = 3$ models processing the $k+1$th observation. Internal variables are shown in black, external inputs, e.g., observations are in red, and system outputs are in green. The system receives the latest filter estimates and model probabilities from the previous update step k as internal inputs, while observations are acquired externally. The system outputs a combined estimate which is not fed back into the system however individual filter outputs are reused in subsequent iterations.

defined in (8), are not only essential for updating the posterior moments of each model but also directly serve in evaluating the model likelihoods without requiring any additional computation. These variables are fed directly into the likelihood expression

$$p(y_{k+1} \mid \boldsymbol{x}_{k+1}, m_{k+1} = i) = \mathcal{N}(y_{k+1}; \hat{y}^{(i)}_{k+1}, s^{(i)}_{k+1}), \tag{21a}$$

$$\hat{y}^{(i)}_{k+1} = (\boldsymbol{h}^{(i)}_{k+1})^T \hat{\boldsymbol{f}}^{(i)}_{k+1|k}, \tag{21b}$$

$$s^{(i)}_{k+1} = (\boldsymbol{h}^{(i)}_{k+1})^T \boldsymbol{\Sigma}^{(i)}_{k+1|k} \boldsymbol{h}^{(i)}_{k+1} + (\sigma^{(i)}_{k+1})^2, \tag{21c}$$

which are then used to update the model probabilities as

$$\omega^{(i)}_{k+1} = \frac{p(y_{k+1}|\boldsymbol{x}_{k+1}, m_{k+1} = i) \sum_{j=1}^{M} \omega^{(j)}_k [\boldsymbol{\Xi}]_{ij}}{\sum_{l=1}^{M} p(y_{k+1}|\boldsymbol{x}_{k+1}, m_{k+1} = l) \sum_{j=1}^{M} \omega^{(j)}_k [\boldsymbol{\Xi}]_{lj}}. \tag{22}$$

Finally, a weighted sum of the posterior densities associated with each model is returned to the user as the output estimate $\bar{\boldsymbol{f}}_{k+1} \sim \mathcal{N}(\hat{\bar{\boldsymbol{f}}}_{k+1}, \bar{\boldsymbol{\Sigma}}_{k+1})$ with

$$\hat{\bar{\boldsymbol{f}}}_{k+1} = \sum_{i=1}^{M} \omega^{(i)}_{k+1} \hat{\boldsymbol{f}}^{(i)}_{k+1|k+1}, \tag{23a}$$

$$\bar{\boldsymbol{\Sigma}}_{k+1} = \sum_{i=1}^{M} \omega^{(i)}_{k+1} \left(\boldsymbol{\Sigma}^{(i)}_{k+1|k+1} + \Psi(\hat{\boldsymbol{f}}^{(i)}_{k+1|k+1} - \hat{\bar{\boldsymbol{f}}}_{k+1}) \right), \tag{23b}$$

and the process repeats when the next observation is received.

A visualization of the algorithm and its pseudocode for processing the $(k+1)$th observation ($k \in \{0, \ldots, N\}$) are shown in Fig. 2 and Algorithm 1, respectively. The algorithm begins by constructing M GP mixtures using the most recent filter estimates. The moments of these mixtures are then utilized as priors. The algorithm proceeds by updating the state estimates and the model probabilities with the new observation. The output of the algorithm consists of the moments of the GP mixture, derived from the latest state estimates and model probabilities. This process is repeated each time a new observation is received. It is important to note that, before processing the first observation, each filter must be initialized using the prior distributions as in (15a).

Algorithm 1: MM-RGP for the $(k+1)$th Observation.

Input: $\{\omega^{(i)}_k\}_{i=1}^M, \{\hat{\boldsymbol{f}}^{(i)}_{k|k}\}_{i=1}^M, \{\boldsymbol{\Sigma}^{(i)}_{k|k}\}_{i=1}^M$
Observation: $(y_{k+1}, \boldsymbol{x}_{k+1})$
for $i = 1$ to M **do** ▷Construct Gaussian mixtures
  Compute mixture probabilities $\gamma^{(ij)}_k$ using (18)
  Compute mixture moments $\hat{\bar{\boldsymbol{f}}}^{(i)}_k$, $\bar{\boldsymbol{\Sigma}}^{(i)}_k$ using (19)
end for
for $i = 1$ to M **do** ▷ Compute filter updates
  Compute updates $\hat{\boldsymbol{f}}^{(i)}_{k+1|k+1}$, $\boldsymbol{\Sigma}^{(i)}_{k+1|k+1}$ using (20)
  Compute probabilities $\omega^{(i)}_{k+1}$ using (22)
end for
Compute moments of $\mathcal{N}(\hat{\bar{\boldsymbol{f}}}_{k+1}, \bar{\boldsymbol{\Sigma}}_{k+1})$ using (23)
Output: $\hat{\bar{\boldsymbol{f}}}_{k+1}, \bar{\boldsymbol{\Sigma}}_{k+1}$

B. PREDICTIVE DISTRIBUTION FOR MULTIPLE MODEL GPS

In our setting, predictive distributions from GPs are especially valuable because they quantify uncertainty about unseen regions of the trajectory, which is essential for robust decision making. In the context of multiple model GPs, predictive distribution can be considered as a mixture of Gaussians, similar to (11), which can be potentially evaluated at a different set of prediction points

$$p(\tilde{\boldsymbol{f}} \mid \mathcal{D}_{1:k}, \mathcal{K}) = \sum_{i=1}^{M} \omega^{(i)}_k \mathcal{N}(\tilde{\boldsymbol{f}}; \hat{\boldsymbol{f}}^{(i)}_{k|k}, \boldsymbol{\Sigma}^{(i)}_{k|k}), \tag{24}$$

where $[\tilde{\boldsymbol{f}}]_n \triangleq f(\boldsymbol{z}_n)$ for test points $\mathcal{Z} \triangleq \{\boldsymbol{z}_n \mid \boldsymbol{z}_n \in \mathcal{S}\}_{n=1}^{\tilde{N}}$. Note that $\mathcal{Z}$ are distinct from $\mathcal{X}^*$, i.e., $\mathcal{X}^* \cap \mathcal{Z} = \emptyset$. Given the

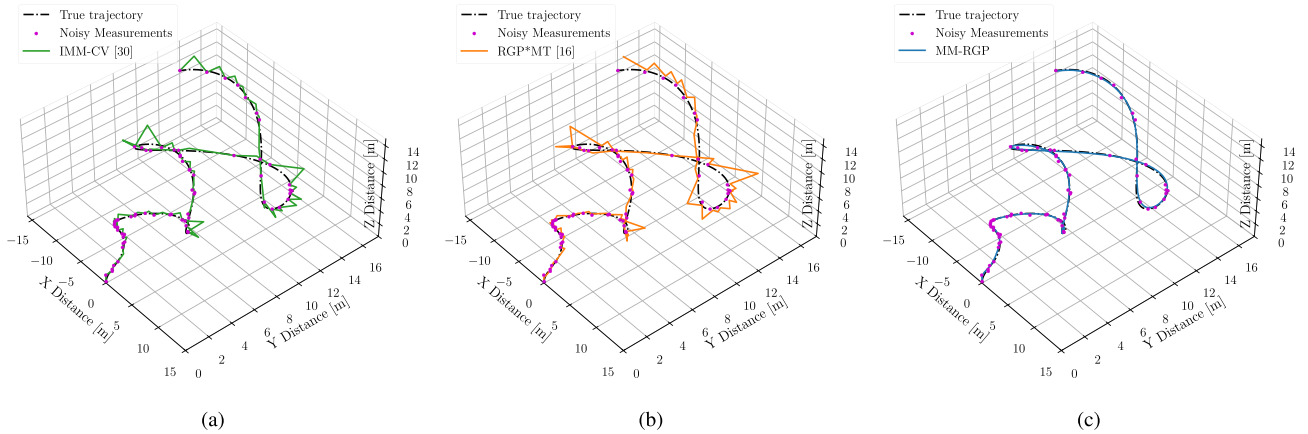

FIGURE 3. Comparison of IMM-CV [30] (a), RGP*MT [16] (b) and MM-RGP algorithm (c) on a set of observations of the trajectory in (26).

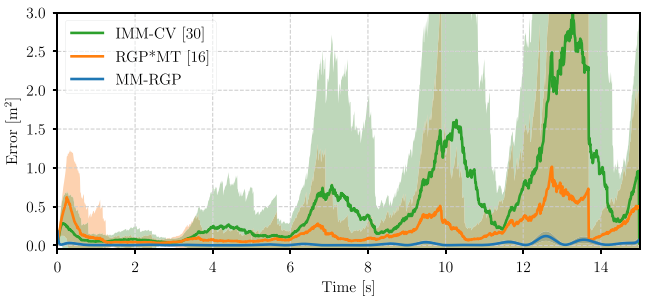

FIGURE 4. Squared errors of IMM-CV [30], RGP*MT [16] and the proposed MM-RGP algorithm over 500 MC runs. Shaded areas indicate 3-σ error bounds.

posterior means $\hat{\boldsymbol{f}}_{k|k}^{(i)}$ and the covariances $\boldsymbol{\Sigma}_{k|k}^{(i)}$, the predictive distributions $\tilde{\boldsymbol{f}} \mid \mathcal{D}_{1:k}, \kappa^{(i)} \sim \mathcal{N}(\hat{\tilde{\boldsymbol{f}}}_k^{(i)}, \tilde{\boldsymbol{\Sigma}}_k^{(i)})$ can be calculated for each component individually as

$$\hat{\tilde{\boldsymbol{f}}}_k^{(i)} = (\boldsymbol{K}_{*Z}^{(i)})^T (\boldsymbol{K}_{**}^{(i)})^{-1} \hat{\boldsymbol{f}}_{k|k}^{(i)}, \tag{25a}$$

$$\tilde{\boldsymbol{\Sigma}}_k^{(i)} = \boldsymbol{K}_{ZZ}^{(i)} + (\boldsymbol{K}_{*Z}^{(i)})^T \left((\boldsymbol{K}_{**}^{(i)})^{-1} \boldsymbol{\Sigma}_{k|k}^{(i)} - \boldsymbol{I}_N \right) (\boldsymbol{K}_{**}^{(i)})^{-1} \boldsymbol{K}_{*Z}^{(i)}, \tag{25b}$$

where $[\boldsymbol{K}_{*Z}^{(i)}]_{jl} \triangleq \kappa^{(i)}(\boldsymbol{x}_j^*, \boldsymbol{z}_l)$ and $[\boldsymbol{K}_{ZZ}^{(i)}]_{jl} \triangleq \kappa^{(i)}(\boldsymbol{z}_j, \boldsymbol{z}_l)$ represent the relationship between $(\mathcal{X}^*, \mathcal{Z})$ and $(\mathcal{Z}, \mathcal{Z})$, respectively. Together, these predictive components can be combined, similar to (11), to express predictions with uncertainty in unobserved areas of the true function.

IV. EXPERIMENTS

We now present the experimental evaluations of our proposed algorithm, comparing and benchmarking it against two algorithms namely the RGP*MT method introduced in [16] and a conventional IMM filter composed of constant velocity models [30], i.e., IMM-CV. The RGP*MT method is a recently proposed GP based tracking approach with online kernel learning. In contrast, the IMM-CV tracker comprises multiple Kalman filters, each equipped with different process noise covariances.

Our evaluation consists of two datasets. The first dataset examines a simulated 3D trajectory designed to highlight the behavior of each method when applied to an infinitely differentiable and complex motion pattern. The second uses a real-world dataset, i.e., the Zurich urban micro aerial vehicle dataset [41], which includes ground truth trajectories and onboard GPS measurements from a micro aerial vehicle (MAV). For both datasets, we uniformly sample a small subset of time instances from the full set and use the corresponding noisy observations, which effectively sparsifies the data. We evaluate performance by computing squared errors over time to assess the accuracy of trajectory prediction. In addition, we report average runtimes to compare the computational efficiency of the algorithms. To evaluate the proposed algorithm we use the predictive distribution, as presented in III-B, throughout the whole trajectory, while RGP*MT and IMM-CV will rely on their respective prediction schemes. The Python code for all the experiments are available at https://github.com/asil-lab/aeb_mm_rgp.

A. SIMULATED 3D TRAJECTORY

We begin with a toy example involving the estimation of a simulated 3D trajectory from noisy point measurements. The ground truth trajectory is defined as a function of time

$$\boldsymbol{f}(t) = \begin{bmatrix} f_x(t) \\ f_y(t) \\ f_z(t) \end{bmatrix} = \begin{bmatrix} t \cos\big(t + \log(t+1)/5\big) \\ t \log\big(\sin^2(t) + 2\big) \\ 3\sqrt{t} \end{bmatrix}, \tag{26}$$

for $t \in [0,\ 15]$. This trajectory is infinitely differentiable but does not conform to standard motion models such as constant velocity, acceleration, or jerk. As such, model based algorithms are generally ill suited for this type of motion.

In this scenario, we choose to use five different SE kernels for MM-RGP, as defined in (2), with length scale parameters $\ell \in \{1,\ 3,\ 5,\ 7,\ 9\}$ coupled with a stochastic matrix selected as 0.9 for the diagonal elements and 0.025 for the off diagonal

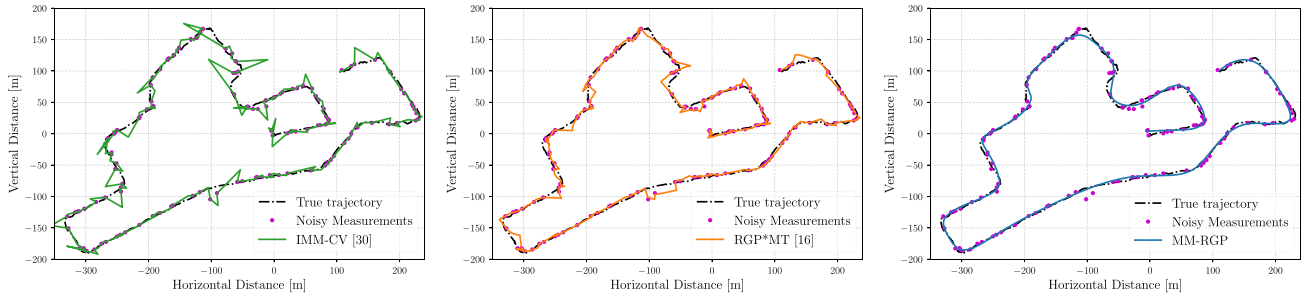

FIGURE 5. Comparison of IMM-CV [30] (a), RGP*MT [16] (b) and MM-RGP algorithm (c) on UZH MAV dataset [41].

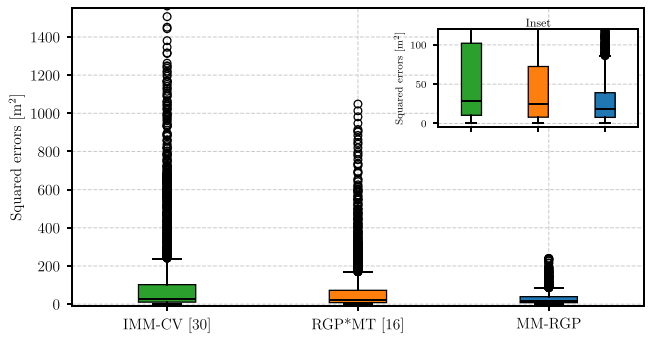

FIGURE 6. Comparison of the squared errors and outliers of the algorithms on the UZH MAV dataset [41]. An inset in the top right corner provides a closer view.

elements. The process noise covariance is selected as $\boldsymbol{Q}_k^{(i)} = \alpha \boldsymbol{\Sigma}_{k|k}^{(i)}$ where $\alpha = 10^{-6}$. The RGP*MT algorithm uses an SE kernel with an initial length scale of $\ell = 5$ and optimizes the hyperparameters over time. For IMM-CV, we use five process noise covariance matrices of the form $\boldsymbol{Q}_k = q\boldsymbol{I}_4$, with $q \in \{1 \times 10^{-2}, 5 \times 10^{-2}, 1 \times 10^{-1}, 1.5 \times 10^{-1}, 2 \times 10^{-1}\}$.

Fig. 3 shows three figures with the true trajectory along with the same set of noisy observations that are inputs to the three algorithms under consideration. From left to right, these plots are overlaid with the estimated trajectories produced by IMM-CV, RGP*MT, and the proposed algorithm MM-RGP. In this single trial, notice that the performance of the tracking significantly improves as we move from left to right. To further understand the performance, we conducted 500 Monte Carlo runs on the same trajectory, with random sampling of observation times and noise. The mean squared error versus time plots of the three algorithms are shown in Fig. 4, along with their respective 3-σ error bounds. The results clearly demonstrate that the superior performance of our proposed MM-RGP significantly outperforms the state of the art IMM-CV and RGP*MT methods.

The IMM-CV method [30] uses constant velocity motion models, which are insufficient to capture the complexity of the target's movement in this scenario, leading to inaccurate predictions. Classical IMM filters handle smooth motion transitions, but tend to overshoot and recover slowly after abrupt motion behavior changes. In contrast our approach avoids predefined motion models which is the key reason for lower error means and variances, despite this complex motion scenario. The RGP*MT approach [16] appears to suffer from overfitting, likely due to its reliance on hyperparameter optimization. This overfitting diminishes its robustness, particularly in the presence of sparse data. Our adaptive kernel learning approach embedded within the proposed MM-RGP addresses these challenges by dynamically adjusting model probabilities based on the incoming data, which helps avoid overfitting and provides more accurate predictions.

B. ZURICH URBAN MAV DATASET

The Zurich Urban MAV Dataset [41] provides high quality multimodal sensor data for drone navigation research. It includes synchronized recordings from event cameras, standard RGB cameras, IMU, GPS, and a motion capture based ground truth system, which enable detailed analysis of flight dynamics and positioning. In our work, we used the 2D projection of dataset's ground truth data over 2650 time instances and the 2D projections of 140 randomly selected GPS measurements.

For this experiment MM-RGP is equipped with five distinct Matérn kernels [31, Chapter 4], each with a smoothness parameter of 3/2 and length scales $\ell \in \{50, 150, 250, 350, 450\}$. The choice of a different kernel, i.e., Matern, as compared to SE in the previous simulated dataset, is to illustrate the expressivity of our proposed approach. As the process noise covariance we choose $\boldsymbol{Q}_k^{(i)} = \alpha \boldsymbol{\Sigma}_{k|k}^{(i)}$ where $\alpha = 10^{-6}$. The matrix $\boldsymbol{\Xi}$ is selected as 0.9 along the diagonal, 0.025 for off diagonal elements. To ensure a fair comparison, the Matérn kernel is also used for the RGP*MT algorithm with the same smoothness parameter and an initial length scale of $\ell = 250$. The IMM-CV algorithm uses five process noise covariance matrices of the form $\boldsymbol{Q}_k = q\boldsymbol{I}_4$, with $q \in \{1,\ 2,\ 3,\ 4,\ 5\}$.

Fig. 5 shows the true trajectory along with the estimates of all algorithms under consideration, using the same set of observations. The IMM-CV algorithm is observed to overfit to the measurements, which leads to noticeable deviations from the true trajectory, especially during abrupt changes in the MAV's direction. In contrast, the RGP*MT algorithm performs well overall, although its accuracy deteriorates in

TABLE I. Average Runtime Comparison of IMM-CV, RGP*MT, and the Proposed Algorithm MM-RGP

Algorithm	Configuration	Avg. Runtime (μs)
IMM-CV	4 states, 5 models	335
Proposed	50 states, 5 models, SE kernel	763
Proposed	50 states, 5 models, Matérn kernel	949
RGP*MT	50 states, SE kernel	2250
RGP*MT	50 states, Matérn kernel	8904

the presence of outlier observations. The proposed approach demonstrates superior robustness, effectively handling outliers and avoiding model mismatch, which makes it the top overall performer in real world scenarios. In Fig. 6, the box plot shows that the outlier errors are much less spread for our approach, indicating the robustness of the algorithm in real world scenarios. It is also worth noting that the proposed approach achieves this performance in a scenario where the drone follows a trajectory with non-smooth and irregular motion patterns that are commonly observed in practical tracking situations. Furthermore, the flexibility of the proposed approach enables it to maintain strong performance even under the additional challenge of the sparse measurements present in this experiment.

C. RUNTIME COMPARISON

All experiments were conducted on a MacBook Pro equipped with an Apple M3 chip, using Python 3.11. Table 1 presents the average runtimes to process a single observation within the same experimental setup, with specified state vector sizes and kernels used. Note that IMM-CV uses state vectors of size 2, consisting of target position and velocity, for each tracking dimension. In contrast, both RGP*MT and MM-RGP use state vectors where the state vector sizes are equal to the number of inducing points in each tracking dimension. Hence, IMM-CV is significantly faster than both RGP*MT and MM-RGP. However, MM-RGP is an order of magnitude faster than RGP*MT, despite outperforming in both simulated and real-world datasets. MM-RGP thus strikes a balance between model complexity and computational efficiency, while delivering superior overall performance.

V. CONCLUSION

In this paper, we presented an adaptive GP mixture based approach for tracking highly maneuverable targets such as UAVs. Our method employs an ensemble of filters with an online mixture weight update scheme inspired by the IMM filter formulation [39]. Compared to previous ensemble GP approaches [28], [29], our formulation is simpler yet effective and readily extensible to a broader range of topologies without relying on basis function representations [42], [43]. We demonstrated the efficacy of our approach against RGP*MT, a previously proposed GP based tracker [16], and IMM-CV, a classical constant velocity IMM tracker [30], achieving lower tracking errors and smoother trajectory estimates. Additionally, our method exhibits improved estimation and runtime performance compared to RGP*MT, with performance gains becoming more pronounced as the number of basis points increases. For future work, we aim to extend our approach to scenarios with sparse observations, such as ADS-B based aircraft tracking in regions with limited ground station coverage [5] and intermittently connected drone swarms, where observations are typically sparse [44]. At the intersection of machine learning [45], [46], [47], [48] and target tracking [49], [50] fields, our work aims to contribute a solution that is not only effective in learning dynamics, but also broadly applicable to related fields.

APPENDIX A

We begin with the observation that the distribution of $\boldsymbol{f}_k \triangleq \boldsymbol{f} \mid \mathcal{D}_{1:k}$ can be factorized using the chain rule as follows

$$p(\boldsymbol{f} \mid \mathcal{D}_{1:k}) \propto \prod_{n=1}^{k} p(y_n \mid \boldsymbol{f}, \boldsymbol{x}_n, \mathcal{D}_{1:n-1})\, p(\boldsymbol{f}), \tag{27}$$

and subsequently using (5) we have

$$p(\boldsymbol{f} \mid \mathcal{D}_{1:k}) \propto \prod_{n=1}^{k} p(y_n \mid \boldsymbol{f}, \boldsymbol{x}_n)\, p(\boldsymbol{f}), \tag{28}$$

$$\propto p(y_k \mid \boldsymbol{f}, \boldsymbol{x}_k)\, p(\boldsymbol{f} \mid \mathcal{D}_{1:k-1}), \tag{29}$$

which results in the following recursions

$$p(\boldsymbol{f} \mid \mathcal{D}_{1:k}) \propto p(y_k \mid \boldsymbol{f}, \boldsymbol{x}_k)\, p(\boldsymbol{f} \mid \mathcal{D}_{1:k-1}). \tag{30}$$

We can then calculate the observation likelihood $p(y_k \mid \boldsymbol{f}, \boldsymbol{x}_k)$ by considering the joint distribution

$$\begin{bmatrix} \boldsymbol{f} \\ y_k \end{bmatrix} \mid \boldsymbol{x}_k \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K}_{**} & \boldsymbol{\kappa}_{*k} \\ \boldsymbol{\kappa}_{*k}^T & \kappa(\boldsymbol{x}_k, \boldsymbol{x}_k) + \sigma_\epsilon^2 \end{bmatrix}\right), \tag{31}$$

and using Gaussian conditioning rules to obtain

$$y_k \mid \boldsymbol{f}, \boldsymbol{x}_k \sim \mathcal{N}(\boldsymbol{h}_k^T \boldsymbol{f},\ \kappa(\boldsymbol{x}_k, \boldsymbol{x}_k) + \sigma_\epsilon^2 - \boldsymbol{h}_k^T \boldsymbol{K}_{**} \boldsymbol{h}_k), \tag{32}$$

where $\boldsymbol{h}_k \triangleq (\boldsymbol{\kappa}_{*k}^T \boldsymbol{K}_{**}^{-1})^T$ which is equivalent to the linear observation model (6).

REFERENCES

[1] Y. Bar-Shalom and X.-R. Li, *Multitarget-Multisensor Tracking: Princ. Techn.*, vol. 19. Storrs, CT, USA: YBS Publishing 1995.

[2] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation With Applications to Tracking and Navigation: Theory Algorithms and Software*. Hoboken, NJ, USA: Wiley, 2001.

[3] C. Spring-Turner and R. T. Rajan, "Performance bounds for cooperative localisation in the starlink network," 2022, *arXiv:2207.04691*.

[4] T. Manteaux, D. Rodríguez-Martínez, and R. T. Rajan, "RAPF: Efficient path planning for lunar microrovers," in *Proc. 2024 Int. Conf. Space Robot.*, 2024, pp. 56–63.

[5] X. Yang, J. Sun, and R. T. Rajan, "Aircraft trajectory prediction using ADS-B data," in *Proc. Symp. Inf. Theory Signal Process. Benelux*, 2022, pp. 113–122.

[6] H. Shakhatreh et al., "Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges," *IEEE Access*, vol. 7, pp. 48572–48634, 2019.

[7] N. Mohd Noor, A. Abdullah, and M. Hashim, "Remote sensing UAV/drones and its applications for urban areas: A review," in *Proc. IOP Conf. Ser., Earth Environ. Sci.*, vol. 169, no. 1, 2018, doi: 10.1088/1755-1315/169/1/012003.

[8] Y. Zhu et al., "UAV trajectory tracking via RNN-enhanced IMM-KF with ADS-B data," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2024, pp. 1–6.

[9] F. Hoffmann, M. Ritchie, F. Fioranelli, A. Charlish, and H. Griffiths, "Micro-Doppler based detection and tracking of UAVs with multistatic radar," in *Proc. IEEE Radar Conf.*, 2016, pp. 1–6.

[10] R. Abdelfatah, A. Moawad, N. Alshaer, and T. Ismail, "UAV tracking system using integrated sensor fusion with RTK-GPS," in *Proc. Int. Mobile, Intell., Ubiquitous Comput. Conf.*, Cairo, Egypt, 2021, pp. 26–27.

[11] X. Yan et al., "UAV detection and tracking in urban environments using passive sensors: A survey," *Appl. Sci.*, vol. 13, no. 20, 2023, Art. no. 11320.

[12] J. R. S. Benevides, M. A. D. Paiva, P. V. G. Simplício, R. S. Inoue, and M. H. Terra, "Disturbance observer-based robust control of a quadrotor subject to parametric uncertainties and wind disturbance," *IEEE Access*, vol. 10, pp. 7554–7565, 2022.

[13] R. A. Zitar, A. Mohsen, A. E. Seghrouchni, F. Barbaresco, and N. A. Al-Dmour, "Intensive review of drones detection and tracking: Linear Kalman filter versus nonlinear regression, an analysis case," *Arch. Comput. Methods Eng.*, vol. 30, no. 5, pp. 2811–2830, 2023.

[14] M. Liu, G. Chowdhary, B. C. da Silva, S.-Y. Liu, and J. P. How, "Gaussian processes for learning and control: A tutorial with examples," *IEEE Control Syst. Mag.*, vol. 38, no. 5, pp. 53–86, Oct. 2018.

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

[16] W. Aftab and L. Mihaylova, "A learning Gaussian process approach for maneuvering target tracking and smoothing," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 57, no. 1, pp. 278–292, Feb. 2021.

[17] F. Lydeard, B. I. Ahmad, and S. Godsill, "Integrated Gaussian processes for robust and adaptive multi-object tracking," 2025, *arXiv:2507.04116*.

[18] F. Goodyer, B. I. Ahmad, and S. Godsill, "GaPP: Multi-target tracking with Gaussian processes," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2023, pp. 1–5.

[19] Q. Guo, L. Teng, T. Yin, Y. Guo, X. Wu, and W. Song, "Hybrid-driven Gaussian process online learning for highly maneuvering multi-target tracking," *Front. Inf. Technol. Electron. Eng.*, vol. 24, no. 11, pp. 1647–1656, 2023.

[20] X. Liu, C. Lyu, J. George, T. Pham, and L. Mihaylova, "A learning distributed Gaussian process approach for target tracking over sensor networks," in *Proc. 25th Int. Conf. Inf. Fusion*, 2022, pp. 1–8.

[21] P. Zhai and R. T. Rajan, "Distributed Gaussian process hyperparameter optimization for multi-agent systems," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2023, pp. 1–5.

[22] Y. Kındap and S. Godsill, "Non-Gaussian process dynamical models," *IEEE Open J. Signal Process.*, vol. 6, pp. 213–221, 2025.

[23] Z. Hu and T. Li, "A particle Bernoulli filter based on Gaussian process learning for maneuvering target tracking," in *Proc. 30th Eur. Signal Process. Conf.*, 2022, pp. 777–781.

[24] H. Zhang, X. Ye, and Q. Hu, "Spatiotemporal learning via mixture importance Gaussian filtering with sparse regularization," *IEEE Signal Process. Lett.*, vol. 30, pp. 279–283, 2023.

[25] M. Sun, M. E. Davies, I. K. Proudler, and J. R. Hopgood, "A Gaussian process regression based dynamical models learning algorithm for target tracking," 2022, *arXiv:2211.14162*.

[26] V. Tresp, "Mixtures of Gaussian processes," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 13, 2000, pp. 633–639.

[27] S. Seitz, "Mixtures of Gaussian processes for regression under multiple prior distributions," 2021, *arXiv:2104.09185*.

[28] Q. Lu, G. Karanikolas, Y. Shen, and G. B. Giannakis, "Ensemble Gaussian processes with spectral features for online interactive learning with scalability," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 1910–1920.

[29] Q. Lu, G. V. Karanikolas, and G. B. Giannakis, "Incremental ensemble Gaussian processes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 1876–1893, Feb. 2023.

[30] T. Kirubarajan and Y. Bar-Shalom, "Kalman filter versus IMM estimator: When do we need the latter?," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 39, no. 4, pp. 1452–1457, Oct. 2003.

[31] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2005.

[32] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.

[33] M. F. Huber, "Recursive Gaussian process regression," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2013, pp. 3362–3366.

[34] R. Durrett, *Probability: Theory and Examples*, vol. 49. Cambridge, U.K.: Cambridge Univ. Press, 2019.

[35] S. N. Ethier and T. G. Kurtz, *Markov Processes: Characterization and Convergence*. Hoboken, NJ, USA: Wiley, 2009.

[36] W.-K. Ching and M. K. Ng, *Markov Chains: Models, Algorithms and Applications*. Berlin, Germany: Springer, 2006.

[37] C. Rasmussen and Z. Ghahramani, "Infinite mixtures of Gaussian process experts," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 14, 2001, pp. 881–888.

[38] E. Meeds and S. Osindero, "An alternative infinite mixture of Gaussian process experts," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 18, 2005, pp. 883–890.

[39] 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.

[40] P. Billingsley, "Statistical methods in Markov chains," *Ann. Math. Statist.*, Inst. Math. Statist., vol. 32, no. 1, pp. 12–40, 1961.

[41] A. L. Majdik, C. Till, and D. Scaramuzza, "The Zurich urban micro aerial vehicle dataset," *Int. J. Robot. Res.*, vol. 36, no. 3, pp. 269–273, 2017.

[42] T.-M. Nguyen et al., "A third-order Gaussian process trajectory representation framework with closed-form kinematics for continuous-time motion estimation," 2024, *arXiv:2410.22931*.

[43] Y. Sun and S. Yang, "Manifold-constrained gaussian process inference for time-varying parameters in dynamic systems," *Statist. Comput.*, vol. 33, no. 6, 2023, Art. no. 142.

[44] Y. Park and Y. Kim, "Circumnavigation of multiple drones under intermittent observation: An integration of guidance, control, and estimation," *Int. J. Aeronautical Space Sci.*, vol. 23, no. 2, pp. 423–433, 2022.

[45] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Front. Comput. Sci.*, vol. 14, no. 2, pp. 241–258, 2020.

[46] A. Jain, F. Smarra, and R. Mangharam, "Data predictive control using regression trees and ensemble learning," in *Proc. IEEE 56th Annu. Conf. Decis. Control*, 2017, pp. 4446–4451.

[47] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 6405–6416.

[48] L. V. Krannichfeldt, Y. Wang, and G. Hug, "Online ensemble learning for load forecasting," *IEEE Trans. Power Syst.*, vol. 36, no. 1, pp. 545–548, Jan. 2021.

[49] J. Lim, H.-S. Kim, and H.-M. Park, "Interactive-multiple-model algorithm based on minimax particle filtering," *IEEE Signal Process. Lett.*, vol. 27, pp. 36–40, 2020.

[50] G. Wang, X. Wang, and Y. Zhang, "Variational Bayesian IMM-filter for JMSS with unknown noise covariances," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 2, pp. 1652–1661, Apr. 2020.