sensors

MDPI

Article

Trajectory Modeling by Distributed Gaussian Processes in Multiagent Systems

Dongjin Xin * and Lingfeng Shi

School of Electronic Engineering, Xidian University, Xi'an 710071, China
* Correspondence: xindj@stu.xidian.edu.cn; Tel.: +86-18516200637

Abstract: This paper considers trajectory a modeling problem for a multi-agent system by using the Gaussian processes. The Gaussian process, as the typical data-driven method, is well suited to characterize the model uncertainties and perturbations in a complex environment. To address model uncertainties and noises disturbances, a distributed Gaussian process is proposed to characterize the system model by using local information exchange among neighboring agents, in which a number of agents cooperate without central coordination to estimate a common Gaussian process function based on local measurements and datum received from neighbors. In addition, both the continuous-time system model and the discrete-time system model are considered, in which we design a control Lyapunov function to learn the continuous-time model, and a distributed model predictive control-based approach is used to learn the discrete-time model. Furthermore, we apply a Kullback–Leibler average consensus fusion algorithm to fuse the local prediction results (mean and variance) of the desired Gaussian process. The performance of the proposed distributed Gaussian process is analyzed and is verified by two trajectory tracking examples.

Keywords: trajectory modeling; data-driven approach; distributed Gaussian processes; Lyapunov function; model predictive control (MPC)

Citation: Xin, D.; Shi, L. Trajectory Modeling by Distributed Gaussian Processes in Multiagent Systems. *Sensors* **2022**, *22*, 7887. https://doi.org/10.3390/s22207887

Academic Editor: Wilmar Hernandez

Received: 20 September 2022
Accepted: 13 October 2022
Published: 17 October 2022

1. Introduction

Trajectory tracking is a common problem in control and robotics, and its generation systems represent a large class of dynamical physical models. In the past few decades, various control schemes have been investigated and modeled, and most of them can be considered as a subset of computed torque control laws [1]. Generally speaking, in order to track trajectories, one needs to know the system model, such as the kinematic model, observation model, and motion model [2]. However, in many practical applications, one usually cannot obtain the model information/knowledge, or the system model is dynamical and is difficult to characterize. The system model is often filled with a high degree of uncertainty, nonlinearity, and dependency, which makes it difficult to model accurately. Therefore, traditional modeling methods are no longer suitable for the actual dynamical environment [3]. More recently, data-driven approaches are getting more and more attention in many fields, such as the control and machine learning communities [4,5]. Since data-driven methods can train the system model with high efficiency and precision, they have become the most popular choice for system modeling [6,7]. In particular, the Gaussian process (GP) is the most representative one, and it has been successfully applied to many fields.

A research frontier in the realm of GP is the trajectory modeling issue. Due to its capability to tackle complex perturbations, uncertainties, dependencies, and nonlinearities, GP is becoming a popular choice in various systems, such as in solar power forecasting [8], permanent magnet spherical motors [9], iterative learning control [10], and in swarm kinematic model [11]. In particular, GP has been proven to be effective in improving the learning accuracy and the learning effectiveness of uncertainties and dependencies in low

data regimes [12]. More recently, a non-parametric Gaussian process (GP) was proposed for modeling with quantifiable uncertainty and nonlinearity [13,14] based on implicit variance trade-off [15,16]. This bridges the system modeling and data-driven methods. However, computational burden and hardware requirements make GP impractical for big data sets. Furthermore, the high cost of GP also severely hinders the application to an actual physical system. The engineering community has acknowledged these limitations and has attempted to address the problem. Since one can decompose the learning process into a part for a solution, this inspires one to address it in a distributed manner. Accordingly, a distributed GP is an urgent need [17,18].

Generally speaking, the processing is called distributed manner if it is carried out by a cooperative strategy among nodes without central coordination [19]. The distributed method aims at minimizing the amount of computation and communication required by each node as well as making these requirements scalable in the number of nodes [20]. Distributed methods are available for parameter estimation [14], Kalman filtering [21], control [22], optimization [23], learning [13], etc. A major division among distributed methods is based on whether all nodes estimate the full system state [24] or whether each node only estimates a subset of the state variables [25]. The challenge consists in how to execute the update and fusion step in a distributed manner. Existing fusion strategies are usually from the perspective of state estimation and estimation error covariance. Since GP is indeed a Gaussian probability density function (PDF), the trajectory model constructed by GP requires us to consider fusion strategy from the view of PDF [26]. Therefore, this paper is targeted to design a novel GP fusion strategy for multi-agent systems. Generally, the strategy is organized as follows: after obtaining the local predicted results of GP, we perform a fusion of the Kullback–Leibler average consensus on local predictions of GP among neighbors. The distributed GP model can then be developed and successfully applied in large-scale multi-agent systems.

1.1. Related Works

Gaussian process-based modeling and based trajectory tracking have been widely investigated and applied over the past two decades. In the first place, most focus on the centralized GP and the multi-input–output GP. In addition, they are developed based on the need for engineering applications in learning and control fields such as GP-based tracking control, state space model learning, and their applications to trajectory tracking. For example, Beckers et al. studied the stable Gaussian process-based tracking control of Lagrangian systems [1]. Umlauft et al. learned stable Gaussian process state space models [27], while Mohammad et al. learned stable nonlinear dynamical systems with Gaussian mixture models [5]. In addition, Pushpak et al. designed control barrier functions for unknown nonlinear systems using Gaussian processes [28]. Umlauft et al. considered human motion tracking with stable Gaussian process state space models in ref. [29] and proposed an uncertainty-based control Lyapunov approach for control-affine systems modeled by the Gaussian process [30]. They also calculated uniform error bounds of Gaussian process regression with application to safe control [31]. Even Gaussian process-based trajectory tracking and control are becoming research hotspots; they focus mainly on one agent and are seldom involved in multi-agent systems. In the second place, distributed and centralized GPs are flourishing in solving data-driven learning algorithms for multi-agent systems. Generally speaking, the main research results are organized as follows: (1) For contributions of models and theories, the unknown map function was modeled and characterized as a GP but with zero-mean assumption, and a distributed parameter and non-parameter Gaussian regression was proposed by using Karhunen–Loeve expansion in refs. [13,14]. To scale GP to large datum, Deisenroth et al. introduced a robust Bayesian committee machine, a practical and scalable product-of-experts model for large-scale distributed GP regression [32]. To address the hyperparameter optimization problem in big data processing, Xie et al. proposed an alternative distributed GP hyperparameter optimization scheme using the efficient proximal alternating direction method of multipliers [33]. Multiple-task GP

was studied in ref. [34], while multi-out regression by GP was studied in ref. [35]. Both of them were centralized approaches and could not be extended to a large-scale problem. GP networks were flexible and effective to be used in multi-output regression by combining with variational inference and distributed variational inference in ref. [36], which involved applications to settle non-linear dimension reduction and regression, and provided a powerful tool to address uncertainty and over-fitting problems. (2) For engineering applications, Nerurkar et al. [37] presented a distributed conjugate gradient algorithm for cooperative localization. Franceschelli and Gasparri [38] presented a distributed gossip-based approach to address the pose estimation problem. Cunnigham et al. [39] developed an approach for robot smoothing and mapping by using Gaussian elimination. Distributed localization from distance measurements is studied in [40]. The distributed position estimation was considered in [41]. Distributed rotation estimation algorithm was developed in various engineering [42–44]. Distributed Gauss–Seidel algorithm was studied in [45]. GP for data learning in robotic control was considered in [46]. (3) For trajectory tracking in a multi-agent system, an efficient algorithm was presented in ref. [47] to generate trajectory. Gaussian mixture models were used to learn stable trajectory in ref. [5]. The centralized GP for human motion tracking was studied [48]. (4) For distributed model predictive control (MPC), an overview and future research opportunities were discussed in ref. [49]. A cooperative distributed model predictive control for nonlinear systems was studied in [50], for tracking was studied in ref. [51], for linear systems was studied in ref. [52], and for event-based communication and parallel optimization, it was developed in ref. [53]. Additionally, non-cooperative distributed model predictive control was investigated in ref. [54]. More recently, explicit distributed and localized model predictive control via system-level synthesis was investigated in refs. [55,56] and was applied to the trajectory generation of a multi-agent system in ref. [57]. In short, the study of distributed GP is scarce, especially in the trajectory modeling problem.

1.2. Contributions

More recently, GP was widely used to model tracking systems and applied to track the target in a real-world environment, such as speed racing (quadrotors) [58], trajectory tracking for wheeled mobile robots [59], 3D people tracking [60], and Simultaneous Localization and Mapping (SLAM) [61,62]. However, these applications focus on one agent, which ignores the advantages of multi-agent systems. After surveying these related references, we find that the trajectory tracking problem is mainly solved by control methods, not data-driven methods, and we focus on one agent, not a multi-agent collaboration. In addition, these existing GP-based learning algorithms are limited by training manners. Motivated by the above discussion, we investigate the distributed GP to learn the trajectory system model in this paper. More specifically, the main contributions of the paper are four-fold. (1) Compared with GP in Lagrangian systems [1,58], this paper considers a general state-space model for both discrete-time and continuous-time. (2) Compared with centralized GP in the state space model [27–30], PD control and model predictive control are combined together with GP to achieve tracking system modeling and estimating, which can make the estimation error globally uniformly bounded. (3) Compared with existing multiple GP-based centralized approaches, such as collaborative GP [32–35,63,64], this paper achieves a distributed GP manner to estimate the state, and we apply Kullback–Leibler (KL) average consensus to fuse local training results of GPs, which is different with the Wasserstein metric for measuring GP [65]. (4) Compared with the centralized GP without giving the performance bound [32–35,63,64] or only providing Kullback–Leibler average analysis [26], this paper analyzes the probabilistically globally ultimate bound of distributed GP.

1.3. Paper Structure

The remainder of the paper is organized as follows. Section 2 introduces some preliminaries, including notations, graph theory, Gaussian process, and Kullback–Leibler average consensus. Section 3 states the considered systems. Section 4 designs the local

control strategy and proposes a Kullback–Leibler (KL) average consensus to fuse the local predictions of GP. Section 5 provides two tracking experiments. Finally, Section 6 concludes the paper.

2. Preliminaries

2.1. Notation

Throughout the paper, vectors and vector-valued functions are denoted with bold characters. Matrices are described with capital letters. $trace(\cdot)$, $log(\cdot)$, $det(\cdot)$, $\langle \cdot, \cdot \rangle$, $\| \cdot \|$, $\oplus$, $\odot$, $\mathcal{N}(\cdot, \cdot)$, and $\mathcal{GP}(\cdot, \cdot)$ denote, respectively, the trace operation, the logarithm operation, the determinate operation, the inner product, the 2-norm of a matrix or vector, the addition operation of probability density functions (PDFs), the multiplication operation of PDFs, a Gaussian distribution, and a Gaussian process. Moreover, $\dot{\mathbf{x}}$, $\ddot{\mathbf{x}}$, $\hat{\mathbf{x}}$, and $\bar{\mathbf{x}}$ denote, respectively, the first-order differential operation on $\mathbf{x}$, the second-order differential operation, the prediction, and the mean operation of $\mathbf{x}$. In addition, $D_{\mathrm{KL}}(p \parallel q)$, $\mathbb{E}$ $\mathbb{V}$, $\mathbf{I}_n$ denote, respectively, the KL divergence/distance between probabilities p and q, the expectation operation, the variance operation, and an n-by-n identity matrix. In addition, $\frac{dx}{du}$, $\frac{\partial x}{\partial u}$, and $\mathcal{O}(\cdot)$ denote, respectively, the derivative operation, the partial derivative operation, and the complexity.

2.2. Graph Theory

A graph is defined as $\mathcal{G} = (\mathcal{N}, \varepsilon)$; where $\mathcal{N}$ is a set of nodes and $\varepsilon \subseteq \mathcal{N} \times \mathcal{N}$ a set of edges. In particular, graph $\mathcal{G}$ is undirected iff $(u, v) \in \varepsilon \Leftrightarrow (v, u) \in \varepsilon$ for all $u, v \in \mathcal{N}$. The order is $|\mathcal{N}|$ and the size of $\mathcal{G}$ is $|\varepsilon|$. Further, let $\mathcal{N}^i = \{j \in \mathcal{N} : (j, i)\} \in \varepsilon$ denote the set of neighbors for node i.

2.3. Gaussian Process

Definition 1. ([66]). *A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

A Gaussian process is completely specified by its mean function and covariance function. We define mean function $m(\mathbf{x}) : \mathbf{x} \in \chi \to \mathbb{R}$ and the covariance function (kernel) $k(\mathbf{x}, \mathbf{x}') : \chi \times \chi \to \mathbb{R}$ of a real process $f(\mathbf{x})$ as $m(\mathbf{x}) = \mathbb{E}(f(\mathbf{x}))$, $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - \mathbf{x})(f(\mathbf{x}') - \mathbf{x}')]$, and denote the Gaussian process as $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$. When $f : \chi \to \mathbb{R}^n$ is an n-dimensional map, the GP can be denoted by $f_j(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, $j \in \{1, \ldots, n\}$.

Then, the Gaussian process can be organized as [29]

$$f(\mathbf{x}) = \begin{cases} f_1(\mathbf{x}) \sim \mathcal{GP}(m_1(\mathbf{x}), k_1(\mathbf{x}, \mathbf{x}')) \\ \vdots \\ f_n(\mathbf{x}) \sim \mathcal{GP}(m_n(\mathbf{x}), k_n(\mathbf{x}, \mathbf{x}')) \end{cases}. \tag{1}$$

In addition, the covariance function (kernel) measures similarity between any two states/variables $\mathbf{x}, \mathbf{x}' \in \chi$ and the common kernel functions include the linear kernel, squared-exponential (SE) kernel, the polynomial kernel, the Gaussian kernel, and the Matèrn kernel.

Assumption 1 . *Suppose the measurement equation is $\boldsymbol{y} = f(\boldsymbol{x}) + \epsilon$, where $\boldsymbol{y} \in \mathbb{R}^{\boldsymbol{m}}$ is the observed vector, $\boldsymbol{x} \in \boldsymbol{X} \subset \mathbb{R}^{\boldsymbol{n}}$ is the state vector defined in a compact set $\boldsymbol{X}$, and ϵ is the measurement noise obeying a Gaussian distribution with zero mean and variance $\sigma^2 \boldsymbol{I}_n$ (denoted by $\epsilon \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_n)$). In addition, f is the unknown mapping function ($f : \chi \to \mathbb{R}^n$) and is assumed to be a GP (denoted by $f \sim \mathcal{GP}(0, k_\theta(\boldsymbol{x}, \boldsymbol{x}'))$). Here $k_\theta(\boldsymbol{x}, \boldsymbol{x}')$ is a kernel function with respect to hyper-parameters θ, $\boldsymbol{K}_{\boldsymbol{XX}} = k_\theta(\boldsymbol{X}, \boldsymbol{X})$ denotes the covariance matrix of set $\boldsymbol{X}$, and $\boldsymbol{K}_{\boldsymbol{xX}} = k_\theta(\boldsymbol{x}, \boldsymbol{X})$ denotes the covariance matrix between $\boldsymbol{x}$ and $\boldsymbol{X}$.*

Generally speaking, given a training set $\mathcal{D} = (\mathbf{X}, \mathbf{y})$, (input $\mathbf{X}$ and output $\mathbf{y}$) [29], the log-likelihood can be computed by

$$\begin{aligned} \log p(\mathbf{y}|\mathbf{X}, \theta) = & -\tfrac{1}{2}\mathbf{y}^T\left(\mathbf{K}_{\mathbf{XX}} + \sigma^2\mathbf{I}_n\right)^{-1}\mathbf{y} \\ & -\tfrac{1}{2}\log\left|\mathbf{K}_{\mathbf{XX}} + \sigma^2\mathbf{I}_n\right| - \tfrac{n}{2}\log 2\pi. \end{aligned} \tag{2}$$

Then, when a new input $\mathbf{x}^*$ is introduced, the posterior prediction of the Gaussian process [29] is $p(f(x^*)) = \mathcal{N}(\mu(x^*), \sum(x^*))$, where

$$\begin{aligned} \mu(\mathbf{x}^*) &= \mathbf{K}_{\mathbf{x}^*\mathbf{X}}\left(\mathbf{K}_{\mathbf{XX}} + \sigma^2\mathbf{I}_n\right)^{-1}\mathbf{y}, \\ \textstyle\sum(\mathbf{x}^*) &= \mathbf{K}_{\mathbf{x}^*\mathbf{X}} - \mathbf{K}_{\mathbf{x}^*\mathbf{X}}\left(\mathbf{K}_{\mathbf{XX}} + \sigma^2\mathbf{I}_n\right)^{-1}\mathbf{K}_{\mathbf{X}\mathbf{x}^*}. \end{aligned} \tag{3}$$

To summarize, the likelihood maximization of (2) is performed to compute gradients for training, and the mean and covariance functions (3) are used for fast predictions.

More specifically, given an arbitrary new testing input $\mathbf{x}^* \in \chi$ conditioning a dataset $\mathcal{D}$ described above, the prediction response y^* is jointly Gaussian distributed with the training set, which is given by

$$\begin{aligned} &\begin{bmatrix} y_j^* \\ \mathbf{y}_j \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m_j(\mathbf{x}^*) \\ \mathbf{m}_j \end{bmatrix}, \begin{bmatrix} k_j^* & k_j^T \\ \mathbf{k}_j & \mathbf{K}_j + \sigma^2\mathbf{I} \end{bmatrix}\right), \\ &\mathbf{y}_j = \begin{bmatrix} y_j^{(1)} & \cdots & y_j^{(N)} \end{bmatrix}^T \in \mathbb{R}^n, \\ &\mathbf{m}_j = \left[m_j\left(x_{(1)}\right) \cdots m_j\left(x_{(N)}\right)\right]^T \in \mathbb{R}^n, \\ &k_j^* = k_j(\mathbf{x}^*, \mathbf{x}^*) \in \mathbb{R}, \\ &\mathbf{k}_j = \left[k_j\left(x_{(1)}, x^*\right) \cdots k_j\left(x_{(N)}, x^*\right)\right]^T, \\ &K_j = \begin{bmatrix} k_j\left(x_{(1)}, x_{(1)}\right) & \cdots & k_j\left(x_{(1)}, x_{(N)}\right) \\ \vdots & \ddots & \vdots \\ k_j\left(x_{(N)}, x_{(1)}\right) & \cdots & k_j\left(x_{(N)}, x_{(N)}\right) \end{bmatrix} \in \mathbb{R}^{n\times n}. \end{aligned} \tag{4}$$

For j = 1, . . . , n, the posterior distribution corresponding to fj(·) at x∗ yields a Gaussian distribution with mean function and covariance function as

$$\begin{aligned} \mathbb{E}\left[y_j^*\middle|\mathcal{D}, \mathbf{x}^*\right] &= m_j(\mathbf{x}^*) + \mathbf{k}_j^T\left(\mathbf{K}_j + \sigma^2\mathbf{I}_n\right)^{-1}\left(\mathbf{y}_j - \mathbf{m}_j\right), \\ \mathbb{V}\left[y_j^*\middle|\mathcal{D}, \mathbf{x}^*\right] &= k_j^* - k_j^T\left(K_j + \sigma^2\mathbf{I}_n\right)^{-1}\mathbf{k}_j. \end{aligned} \tag{5}$$

Furthermore, in order to learn the hyper-parameter θ_j given a chosen kernel, we can use the maximum likelihood function based on Bayes' rules as

$$\max_{\theta_j} \log p\left(\mathbf{y}_j\middle|\mathbf{x}_{(1:n)}, \theta_j\right) = \max_{\theta_j}\left(-\frac{1}{2}\mathbf{y}_j^T\mathbf{K}_j^{-1}\mathbf{y}_j - \frac{1}{2}\log det\mathbf{K}_j - \frac{n}{2}\log(2\pi)\right). \tag{6}$$

which can be solved by gradient-based approaches [29].

2.4. Kullback–Leibler Average Consensus Algorithm

This section introduces the consensus/fusion algorithm of GPs. A Gaussian process is a Gaussian probability density function over mean function and covariance function. Therefore, the fusion of GPs is indeed the fusion of probabilities. It raises a problem: How to achieve consensus/fusion of probabilities among multiple agents?

Before proceeding on, we first introduce some definitions.

Definition 2. (Probability space [26]). Let

$$\mathcal{P} \triangleq \left\{ p(\bullet) : \mathbb{R}^n \to \mathbb{R} \text{ such that } \int_{\mathbb{R}^n} p(\mathbf{x}) d\mathbf{x} = 1 \text{ and } p(\mathbf{x}) \geq 0, \forall \mathbf{x} \in \mathbb{R}^n \right\}$$

denote the set of probabilities (PDFs) over $\mathbb{R}^n$ *and let* $p^i(\cdot) \in \mathcal{P}(i \in \mathcal{N})$ *denote the local probability/PDF of agent* i.

Definition 3. *(Kullback–Leibler divergence* [26]*). In statistics, the Kullback–Leibler divergence,* $D_{KL}(p \parallel q)$ *(also called relative entropy), is a statistical distance: a measure of the probability distribution* $p(\cdot)$ *is different from the probability distribution* $q(\cdot)$*, which is defined as (for distributions* $p(\cdot)$ *and* $q(\cdot)$ *of a continuous random variable* $\boldsymbol{x}$*)*

$$D_{KL}(p \parallel q) = \int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x}. \tag{7}$$

Definition 4. *(Probabilistic operation* [26]*). Define the plus* $\oplus$ *and multiplicative* $\odot$ *operators over probabilities (*$p(\cdot)$ *and* $q(\cdot)$ *) for a variable (*$\boldsymbol{x}$ *) and a real constant as a*

$$\begin{aligned} p(\mathbf{x}) \oplus q(\mathbf{x}) &\triangleq \frac{p(\mathbf{x})q(\mathbf{x})}{\int p(\mathbf{x})q(\mathbf{x})d\mathbf{x}}, \\ a \odot p(x) &\triangleq \frac{[p(\mathbf{x})]^a}{\int [p(\mathbf{x})]^a d\mathbf{x}}. \end{aligned} \tag{8}$$

Then, we attempt to find a Kullback–Leibler average consensus/fusion algorithm over probabilities obtained by multiple agents.

First, according to [26], Kullback–Leibler average (KLA) is to average over probabilities. Motivated by this, we define the weighted KLA ($\overline{p}$) among the probabilities $\{p^i\}_{i=1}^{\mathcal{N}}$ as

$$\overline{p} = \underset{p \in P}{arg\inf} \sum_{i \in \mathcal{N}} a^i D_{\mathrm{KL}}\Big(p \parallel p^i\Big), \tag{9}$$

where $a^i \geq 0$ denotes the weight of agent i and satisfies $\sum_{i \in \mathcal{N}} a^i = 1$.

Then, the average consensus/fusion problem is to achieve

$$\lim_{l \to \infty} p_l^i = \overline{p}, \tag{10}$$

for all agents $i \in \mathcal{N}$, where l is the consensus step and $\overline{p}$ represents the asymptotic KLA with uniform weights.

Second, we attempt to find the solution of the average consensus $\overline{p}$ in (9). Based on [26], the solution is

$$\overline{p}(\mathbf{x}) = \frac{\prod\limits_{i \in \mathcal{N}} \left[p^i(\mathbf{x})\right]^{a^i}}{\int \prod\limits_{i \in \mathcal{N}} \left[p^i(\mathbf{x})\right]^{a^i} d\mathbf{x}} \doteq \bigoplus_{i \in \mathcal{N}} \Big(a^i \odot p^i(\mathbf{x})\Big), \tag{11}$$

with $a^i = 1/|\mathcal{N}|$. In addition, the local consensus of agent i at the l-th consensus step can be obtained by

$$p_l^i(\mathbf{x}) = \bigoplus_{i \in \mathcal{N}} \Big(a^{i,j} \odot p_{l-1}^j(\mathbf{x})\Big), \forall i \in \mathcal{N}, \tag{12}$$

where a^{ij} is the consensus weight satisfying $a^{ij} \geq 0$, $\sum_{i \in \mathcal{N}} a^{i,j} = 1$ and a^{ij} represents the (i, j)-th component of the consensus matrix $\mathbf{A}$ (if $j \notin \mathcal{N}^i$ $a^{i,j} = 0$). Therefore, when the l-th iteration of the consensus algorithm is initialized by $p_0^i(\cdot) = p^i(\cdot)$, we can finally obtain the consensus as

$$p_l^i(\mathbf{x}) = \underset{i\in\mathcal{N}}{\oplus}\left(a_l^{i,j} \odot p^j(\mathbf{x})\right), \forall i \in \mathcal{N} \tag{13}$$

Third, for special Gaussian case, the local probability $p^i(\cdot)$ takes the form as

$$p^i(\mathbf{x}) = \mathcal{N}\left(\mathbf{x}; \mu^i, \sum^i\right) \triangleq \frac{1}{\sqrt{det\left(2\pi \sum^i\right)}} e^{-\frac{1}{2}(\mathbf{x}-\mu^i)^T(\Sigma^i)^{-1}(\mathbf{x}-\mu^i)}, \tag{14}$$

where $\mu^i \in \mathbb{R}^n$ and $\sum^i \in \mathbb{R}^{n\times n}$ denote the mean vector and the covariance matrix, respectively. In view of this case, the KLA can be directly obtained by operating the means and the covariances instead of probabilities. The following lemma states the KLA on Gaussian distributions.

Lemma 1. ([26]). *Given $\mathcal{N}$ Gaussian distributions $\left(p^i(x), i = 1, \ldots, \mathcal{N}\right)$ defined in (14), with corresponding weigh a^i , then the weighted KLA $\overline{p}(\cdot) = \mathcal{N}\left(\cdot; \overline{\mu}, \overline{\sum}\right)$ can be calculated by directly fusing the means μ^i and the covariance matrices $\sum^i$ as*

$$\begin{aligned} \overline{\sum}^{-1} &= \sum_{i=1}^{\mathcal{N}} a^i \left(\sum^i\right)^{-1}, \\ \overline{\sum}^{-1}\overline{\mu} &= \sum_{i=1}^{\mathcal{N}} a^i \left(\sum^i\right)^{-1}\mu^i. \end{aligned} \tag{15}$$

Lemma 1 indicates that the consensus/fusion of Gaussian probabilities can directly operate their means and covariance matrices. Note that a Gaussian process is indeed a Gaussian probability. Therefore, the KLA consensus/fusion on GPs can be directly obtained by fusing the mean functions and the covariance functions.

2.5. Uniform Error Bounds

This section analyzes the probabilistic uniform error bounds.

Definition 5. *(Probabilistic uniform error bound* [31]). *$\forall \boldsymbol{x} \in \boldsymbol{X}$, if there exists a function $\eta(x)$ such that $\| \mu(x) - f(x) \| \leq \eta(x)$, then, on a compact set $X \subset \mathbb{R}^n$, GP has a uniformly bounded error. A probabilistic uniform error bound is one that holds with a probability of at least $1-\delta$ for any $\delta \in (0,1)$.*

Definition 6. *(Lipschitz constant of the kernel* [64]). *The Lipschitz constant of a differentiable covariance kernel $k(\cdot,\cdot)$ is*

$$L_k := \max_{\mathbf{x},\mathbf{x}'\in\mathbf{X}} \left\| \begin{bmatrix} \frac{\partial k(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_1} & \cdots & \frac{\partial k(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_n} \end{bmatrix}^T \right\|. \tag{16}$$

Next, we show that the posterior prediction (3) of GP is continuous. Given the continuous unknown f with Lipschitz constant Lf and the Lipschitz continuous kernel k with Lipschitz constant L_k, we then have the following theorem.

Theorem 1. ([31]). *Given a GP defined by the continuous covariance kernel function k with Lipschitz constant L_k , a continuous unknown map f with Lipschitz constant L_f and measurements satisfying Assumption 1. Then, the posterior predictions ($\mu(\cdot)$ and $\sum(\cdot)$) of the GP conditioning on the training date set $\mathcal{D} = \left\{\boldsymbol{x}_t, \boldsymbol{y}_t^i\right\}_{t=1,\ldots,N}^{i=1,\ldots,N}$ are continuous with Lipschitz constant L_μ and modulus of continuity $\omega_\sum$ such that*

$$\begin{aligned} L_\mu &\leq L_k\sqrt{N}\left|\left(\mathbf{K}+\sigma^2\mathbf{I}\right)^{-1}\mathbf{y}\right|, \\ \omega_{\sum(\tau)} &\leq \sqrt{2\tau L_k\left(1+N\left|\left(\mathbf{K}+\sigma^2\mathbf{I}\right)^{-1}\right|\max_{\mathbf{x},\mathbf{x}'\in\mathbf{X}} k(\mathbf{x},\mathbf{x}')\right)}. \end{aligned} \tag{17}$$

for any $\tau \in \mathbb{R}_+$ *and* $\delta \in (0,1)$ *with*

$$\begin{aligned} \beta(\tau) &= 2\log\Big(\tfrac{M(\tau,\boldsymbol{X})}{\delta}\Big), \\ \gamma(\tau) &= \Big(L_\mu + L_f\Big)\tau + \sqrt{\beta(\tau)}\omega_{\sum(\tau)}. \end{aligned} \tag{18}$$

In addition, $\forall \boldsymbol{x} \in \boldsymbol{X}$ *, it follows that*

$$p\left(\|f(\mathbf{x})\| - \mu(\mathbf{x}) \leq \gamma(\tau) + \sqrt{\beta(\tau)\sum(\mathbf{x})}\right) \geq 1-\delta. \tag{19}$$

Proof of Theorem 1. The proof is given in Appendix A. □

Asymptotic Analysis

The asymptotic analysis of the error bound (19) in the limit $N \to \infty$ is organized as the following theorem.

Theorem 2. ([31]). *Given a GP defined by the continuous covariance kernel function k with Lipschitz constant* L_k*, and an infinite data response of measurements* $\left(X_t, y_t^i\right)$ *of the continuous unknown map f with Lipschitz constant* L_f *and the maximum absolute value* $\| f^{max} \|$*. The first N measurements inform the posterior predictions of the GP as* $(\mu(\cdot)$ *and* $\sum(\cdot))$*. If there exists a* $\epsilon > 0$ *such that* $\sum(\mathbf{x}) \in \mathcal{O}\left(\log(N)^{-\frac{1}{2}-\epsilon}\right)$, $\forall \mathbf{x} \in \mathbf{X}$ *for any* $\delta \in (0,1)$*, it follows that*

$$p\left(\sup_{\mathbf{x}\in\mathbf{X}} \|f(\mathbf{x}) - \mu_N(\mathbf{x})\| \in \mathcal{O}\left(\log(N)^{-\frac{1}{2}-\epsilon}\right)\right) \geq 1-\delta. \tag{20}$$

Proof of Theorem 2. The proof is given in Appendix B. □

3. Problem Formulation

The trajectories generate from a continuous dynamical system

$$\dot{\mathbf{x}} = f(\mathbf{x}, u) + w, \tag{21}$$

where $\mathbf{x} \in \chi \subset \mathbb{R}^n$ in a compact set, χ denotes the state (location), $u \in \mathcal{U} \subseteq \mathbb{R}^n$ denotes the control input, ω denotes the process noise with $w \sim \mathcal{N}\left(0, \sigma_\omega^2 \mathbf{I}\right)$ and the initial state is $\mathbf{x}(0) = \mathbf{x}_0$. We have N agents/sensors connected with a network to acquire measurements (location or velocity). In particular, sensor i measures

$$\mathbf{y}_j^i = f^i(\mathbf{x}) + \epsilon_j^i, \tag{22}$$

where $\mathbf{y}_j^i$ is the observed vector of sensor $i (i = 1, \ldots, \mathcal{N})$ at the j-th step $(j = 1, \ldots, N)$, ϵ_j^i is the measurement noise with $\epsilon_j^i \sim \mathcal{N}\left(0, \sigma^2 \mathbf{I}\right)$.

Suppose that a training data set $\mathcal{D}$ of trajectories is given. $\mathcal{D}$ contains the state (current location) and the measurement, which is denoted by $\mathcal{D} = \left\{x^i, y_j^i\right\}_{i=1\ldots\mathcal{N}}^{j=1\ldots N}$. The nonlinear map function $f : \chi \to \mathbb{R}^n$ is unknown and is assumed to be a Gaussian process. In addition, the following assumption is satisfied.

Assumption 2. $f(x)$ *Suppose the measurement is Lipschitz continuous and has a bounded RKHS (reproducing kernel Hilbert space) norm with respect to fixed common kernel k,* $\| f \|_k = \sqrt{f, f_k} < \infty$.

The objective is to find an estimated $\hat{f}$ of f, for which the output trajectory x tracks the desired trajectory $\mathbf{x}_d = \begin{bmatrix} x_d & \dot{x}_d \end{bmatrix}^T$ such that the tracking error $\mathbf{e} = \mathbf{x} - \mathbf{x}_d = \begin{bmatrix} e_1 & e_2 \end{bmatrix}^T$ vanishes over time, i.e., $\lim\limits_{t\to\infty} \| e \| = 0$. l. Since the noises ω and ϵ_j^i and the uncertain dynamics affect the system and control, we use multiple agents to eliminate the influence

of stochastic uncertainty, i.e., given local $\hat{f}^i$, the goal is also to fuse them and to find a fused/consensus $\overline{f}$ such that the uncertainty also vanishes over time.

4. Control Design and Analysis

Classical control uses static feedback gains. Low feedback gains are designed to avoid saturation of the actuators and good noise suppression. However, the considered unknown dynamics require a more minimal feedback gain to keep the tracking error under a defined limit. After performing a training procedure, we use the mean function of GP to adapt the gains. For this purpose, the uncertainty of the GP and multiple agents are employed to scale the feedback gains.

Before proceeding on, the following natural assumptions and lemmas are given.

Assumption 3. *The desired trajectory* $\boldsymbol{x}_d(t)$ *is bounded by* $\| \boldsymbol{x}_d \| = \| \begin{bmatrix} x_d & \dot{x}_d \end{bmatrix}^T \| \leq \boldsymbol{q}_d = \begin{bmatrix} q_d & \dot{q}_d \end{bmatrix}^T$.

Lemma 2. ([67]) *If there exist a positive constant* $b \in \mathbb{R}_+$ *such that* $\forall a \in \mathbb{R}_+$, *if there exists a function* $T = T(\mathbf{a}, \mathbf{b})$ *satisfying* $\| \mathbf{x}(t_0) \| \leq \mathbf{a}$, *then we have that* $\forall t \geq t_0 + T, \| \mathbf{x}(t) \| \leq \mathbf{b}$, *i.e., the trajectory* $\mathbf{x}(t)$ *of the dynamics (21) is globally ultimately bounded.*

Lemma 3. ([67]) *If there exists a Lyapunov function* $\mathbf{V}$ *such that* $\dot{\mathbf{V}}(\mathbf{x}) < 0$ *for all* $\mathbf{x} \in \mathbf{X} \backslash \mathbb{B}$, *the dynamical system* $\dot{\mathbf{x}} = f(\mathbf{x}, u)$ *is globally ultimately bounded to a set* $\mathbb{B} \subset X$.

Next, we design the controller and the control law such that stability and high-performance tracking are achieved. The controller is designed as

$$u = -\hat{f}(\mathbf{x}) + \rho, \tag{23}$$

where $\hat{f}$ is the model estimation of nonlinear dynamics f and $\hat{f}$ is obtained by utilizing the posterior mean function μ_N of GP trained by the data set $\mathcal{D}$, and ρ is the control law.

In addition, ρ is designed as a Proportional–Derivative (PD) type controller

$$\rho = \ddot{x}_d - k_d r - k_p e_2, \tag{24}$$

where $r = k_p e_1 + e_2$ is the filtered state with $\dot{r} = f(\mathbf{x}) - \hat{f}(\mathbf{x}) - k_c r, k_p \in \mathbb{R}_+$, and the control gain $k_p \in \mathbb{R}_+$.

Given the above controller, one needs to verify the effectiveness of the model estimation $\hat{f}$ and the choices of the parameters k_d and k_p. The following theorem states the control law with guaranteed boundedness of the tracking error.

Theorem 3. *Consider the system (23), where f satisfies Assumption 2 and admits a Lipschitz constant* L_f. *If Assumption 3 is satisfied, then the controller with* $\hat{f} = \mu_N$ *and the control law guarantee that the tracking error is globally ultimately bounded and converges to a ball* $\mathbb{B} = \left\{ \| e \| \leq \frac{\gamma(\tau) + \sqrt{\beta(\tau) \sum_N(x)}}{k_c \sqrt{\lambda^2 + 1}} \right\}, \forall \boldsymbol{x} \in \boldsymbol{X}$, *where* β *and* γ *are given in Theorem 1, with a probability of at least* $1 - \delta$, $\delta \in (0, 1)$.

Proof of Theorem 3. The proof is given in Appendix C. □

Remark 1. *From Theorem 3, it can be seen that trajectory tracking with high probability is achieved with the proposed GP-based controller. Compared with most existing results where only uniformly ultimate boundedness of the trajectory tracking errors was achieved* [1,68], *the proposed control law ensures high control precision in the presence of the estimation errors from GP.*

Proof of Remark 1. The proof is given in Appendix C. □

4.1. Consensus

The aforementioned control law focuses one agent/sensor. Since the noises ω and ϵ_j^i affect the measurements and the dynamical system, the proposed controller will fluctuate for different agents. Furthermore, since the uncertainty of dynamics exists, the proposed controller may also change for different agents. Therefore, this section will fuse them and make them reach a consensus, i.e., given local $\hat{f}^i$ (μ_N^i and $\sum_N^i$), the goal is to fuse them and to find a fused/consensus $\overline{f}\left(\overline{\mu}_N \text{ and } \overline{\sum}_N\right)$ such that the uncertainty and the disturbance can vanish over time. Obviously, the controller (23) and the control law (24) for different agents can reach a consensus $\overline{f}\left(\overline{\mu}_N \text{ and } \overline{\sum}_N\right)$.

More specially, after training the local $\hat{f}^i$ by using GP, node i sends the result to its neighbors $\mathcal{N}^i$. After collecting the training results from neighbors, it performs the following dynamic consensus/fusion step. Given weights a^i satisfying $a^i \geq 0$ and $\sum\limits_{\mathrm{i}\in\mathcal{N}^i} a^i = 1$ based on the Kullback–Leibler average consensus given in Section 2.4, the desired weighted KLA takes the Gaussian form as $\overline{f}(\cdot) = \mathcal{N}\left(\cdot; \overline{\mu}, \overline{\sum}\right)$, in which the fusion of the mean function $\overline{\mu}$ and the fusion of the covariance function $\overline{\sum}$ can be calculated by

$$\begin{aligned} \overline{\sum}^{-1} &= \sum_{i=1}^{M} a^i \left(\sum_N^i\right)^{-1}, \\ \overline{\sum}^{-1}\overline{\mu} &= \sum_{i=1}^{M} a^i \left(\sum_N^i\right)^{-1} \mu_N^i, \end{aligned} \tag{25}$$

while the global/centralized fusion using $i = 1, \ldots, \mathcal{N}$. The flowchart is given in Figure 1.

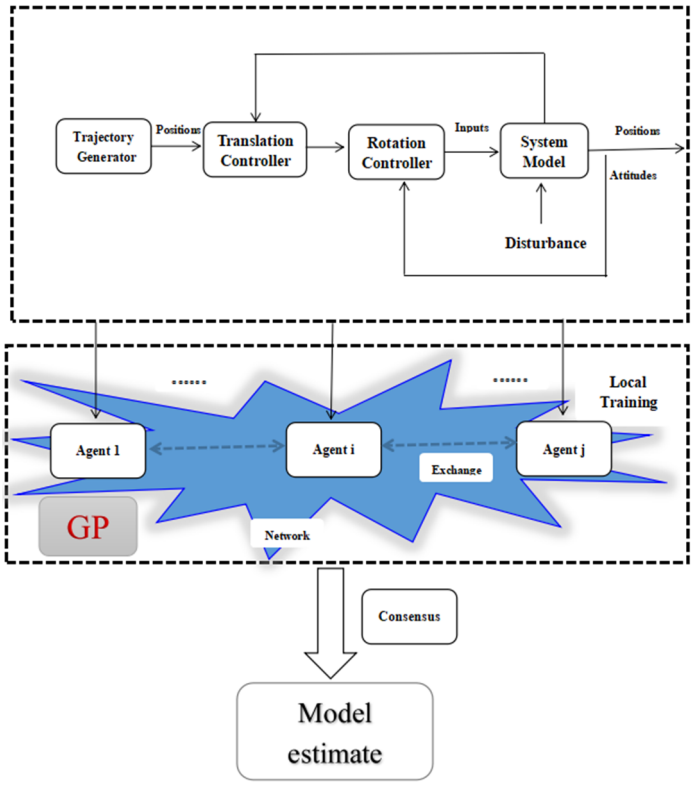


Figure 1. The flowchart of consensus/fusion.

After obtaining the consensus mean function, the controllers of different agents can be designed to be a unified controller.

Remark 2. *The main advantage of the distributed method lies in that local nodes can only receive part of the training data or even missing data. Neighboring nodes can make the prediction faster and keep high accuracy through information interaction and consensus algorithm, which can also avoid processor failure caused by data loss or node/sensor failure.*

4.2. GP-Based Model Predictive Control for Discrete-Time System

The above discussion discusses the continuous-time system. Usually, we need to discretize the system in an actual physical system. This section designs the control strategy for discrete-time by using GP-based model predictive control (MPC).

First, the considered system (21) is assumed to be discrete-time and can be modeled by GP, where the control tuple $\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_k & \mathbf{u}_k \end{bmatrix}^T$ and the state difference $\delta\mathbf{x}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ are, respectively, designed as the training input and the desired target. Given the training date set $\mathcal{D} = \{\mathbf{x}_t, \mathbf{y} = \delta\mathbf{x}_k\}_{k=1\ldots N}$, according to (4) and (5), at a new training input $\mathbf{x}^*$, we can obtain the mean function and covariance function as follows

$$\begin{aligned} \mathbb{E}[\delta\mathbf{x}_k|\mathcal{D}, \mathbf{x}^*] &= \mathbf{k}^T \left(K + \sigma^2 \mathbf{I}_N\right)^{-1} \mathbf{y}, \\ \mathbb{V}[\delta\mathbf{x}_k|\mathcal{D}, \mathbf{x}^*] &= k^* - \mathbf{k}^T \left(K + \sigma^2 \mathbf{I}_N\right)^{-1} \mathbf{k}, \end{aligned} \tag{26}$$

where $k^* = k(\mathbf{x}^*, \mathbf{x}^*)$, $\mathbf{k} = \begin{bmatrix} k\left(\mathbf{x}_{(1)}, \mathbf{x}^*\right) & \ldots & k\left(\mathbf{x}_{(n)}, \mathbf{x}^*\right) \end{bmatrix}^T$, and K is defined in (4). Therefore, (26) is given to predict the next step. By using the moment matching approach [46], the mean function and covariance function of the training target at time k can be calculated by

$$\begin{aligned} \mu_k^\delta &= \mathbb{E}[\mathbb{E}[\delta\mathbf{x}_k]], \\ \sum\nolimits_k^\delta &= \begin{bmatrix} k\left(\delta\mathbf{x}_{k_1}, \delta\mathbf{x}_{k_1}\right) & \cdots & k(\delta\mathbf{x}_{k_n}, \delta\mathbf{x}_{k1}) \\ \vdots & \ddots & \vdots \\ k\left(\delta\mathbf{x}_{k_1}, \delta\mathbf{x}_{k_n}\right) & \cdots & k(\delta\mathbf{x}_{k_n}, \delta\mathbf{x}_{k_n}) \end{bmatrix}. \end{aligned} \tag{27}$$

At time $k+1$, the mean and covariance functions are updated as

$$\begin{aligned} \mu_{k+1} &= \mu_k + \mu_k^\delta, \\ \sum\nolimits_{k+1} &= \sum\nolimits_k + \sum\nolimits_k^\delta + k(\mathbf{x}_k, \delta\mathbf{x}_k) + k(\delta\mathbf{x}_k, \mathbf{x}_k). \end{aligned} \tag{28}$$

For more details, please refer to [46].

Then, based on (6), we next attempt to learn the hyper-parameters θ. A distributed GP-based MPC scheme is presented to address this problem. First, we design the objective function as

$$J_k = \min_u \mathbb{E}[\mathcal{V}(\mathbf{x}_k, \mathbf{u}_{k-1})], \tag{29}$$

where the cost function is

$$\mathbb{E}[\mathcal{V}(\mathbf{x}, \mathbf{u})] = \sum_{l=1}^{L} \mathbb{E}\Big[\left(\mathbf{x}_{k+l} - \mathbf{p}_{k+l}\right)^T \mathbf{Q} \left(\mathbf{x}_{k+l} - \mathbf{p}_{k+l}\right) + u_{k+l-1}^T \mathbf{R} \mathbf{u}_{k+l-1} \Big], \tag{30}$$

where $\mathbf{p}$ is the desired trajectory (desired state), $\mathbf{Q}$ and $\mathbf{R}$ are positive definite weight matrices, and L is the prediction horizon and also the control horizon. According to GP in Section 2, (30) can be rewritten as

$$\mathbb{E}[\mathcal{V}(\mathbf{x}, \mathbf{u})] = \sum_{l=1}^{L} \mathbb{E}\begin{bmatrix} \left(\mathbf{u}_{k+l} - \mathbf{p}_{k+l}\right)^T \mathbf{Q} \left(\mathbf{u}_{k+l} - \mathbf{p}_{k+l}\right) \\ + \text{trace}(\mathbf{Q} \sum_{k+l}) + \mathbf{u}_{k+l-1}^T \mathbf{R} \mathbf{u}_{k+l-1} \end{bmatrix}, \tag{31}$$

Next, to address the optimization problem (29), a gradient-based method is used. Set $F_l = \left(\mathbf{u}_{k+l} - \mathbf{p}_{k+l}\right)^T \mathbf{Q} \left(\mathbf{u}_{k+l} - \mathbf{p}_{k+l}\right) + \text{trace}(\mathbf{Q} \sum_{k+l}) + \mathbf{u}_{k+l-1}^T \mathbf{R} \mathbf{u}_{k+l-1}$ and $\mathbb{E}[\mathcal{V}(\mathbf{x}, \mathbf{u})] = \sum\limits_{l=1}^{L} \mathbb{E} F_l$. Using the chain rule, the gradient can be calculated by

$$\begin{aligned} \frac{d}{d\mathbf{u}_{k-1}}\mathbb{E}[\mathcal{V}(\mathbf{x}_k,\mathbf{u}_{k-1})] &= \sum_{l=1}^{L}\frac{dF_l}{d\mathbf{u}_{k+l-1}}, \\ \frac{dF_l}{d\mathbf{u}_{k+l-1}} &= \frac{\partial F_l}{\partial \mathbf{u}_{k+l}}\frac{\partial \mathbf{u}_{k+l}}{\partial \mathbf{u}_{k+l-1}} + \frac{\partial F_l}{\partial \sum_{k+l}}\frac{\partial \sum_{k+l}}{\partial \mathbf{u}_{k+l-1}} + \frac{\partial F_l}{\partial \mathbf{u}_{k+l-1}}. \end{aligned} \tag{32}$$

where $\frac{\partial F_l}{\partial \mathbf{u}_l}$, $\frac{\partial F_l}{\partial \sum_l}$ and $\frac{\partial F_l}{\partial \mathbf{u}_{l-1}}$ are easy to calculate. In addition,

$$\begin{aligned} \frac{\partial \mathbf{u}_{k+l}}{\partial \mathbf{u}_{k+l-1}} &= \frac{\partial \mathbf{u}_{k+l}}{\partial \mathbf{u}_{k+l-1}}\frac{\partial \mathbf{u}_{k+l-1}}{\partial \mathbf{u}_{k+l-1}}, \\ \frac{\partial \sum_{k+l}}{\partial \mathbf{u}_{k+l-1}} &= \frac{\partial \sum_{k+l}}{\partial \sum_{k+l-1}}\frac{\partial \sum_{k+l-1}}{\partial \mathbf{u}_{k+l-1}}, \end{aligned} \tag{33}$$

where $\frac{\partial \mathbf{u}_{k+l-1}}{\partial u_{k+l-1}}$ and $\frac{\partial \sum_{k+l-1}}{\partial \mathbf{u}_{k+l-1}}$ are easy to calculate.

Finally, the gradient-based algorithm is formulated as Algorithm 1.

Algorithm 1 Gradient-based optimization method

Input: learning GP, L, $\mathbf{p}$, $\mathbf{Q}$, and $\mathbf{R}$
Output: Optimal control u^*
1: Initialization: Max iteration number $N = 1000$, threshold $\varepsilon = 10^{-8}$, the initialized input u_0 and optimal control $u^* = u_0$;
2: **for** $k = 1$ to N **do**
3: **if** $\mathbb{E}[\mathcal{V}] < \varepsilon$ **then**
4: $u^* = u_k$;
5: **end** Loop;
6: **else**
7: Calculate the gradient $\frac{d\mathbb{E}[\mathcal{V}(u_k)]}{du_{k-1}}$ by (32);
8: Update search step size based on [69];
9: Update control $u_{k+l} = u_k + \alpha_l \frac{d\mathbb{E}[\mathcal{V}(u_l)]}{du_{l-1}}$;
10: Go next $l \to l+1$
end
end
11: **return** Optimal control u^*.

Remark 3. *Similarly, due to the stochastic uncertainty caused by the noises and model perturbations, we can use multiple agents to address this problem. The consensus/fusion algorithm is given above, which is similar to the continuous system. Therefore, we will not introduce it any more.*

Remark 4. *The GP has been widely applied in various real-world applications such as quadrotor tracking, 3D people tracking, localization and mapping, and control-based application models. These applications have attracted much attention from engineers and researchers. As for the limitations, in our opinion, the first is that the model needs real-world data to achieve perfect training and application. The second is that the dynamics are Gaussian distributed or Gaussian-approximate distributed.*

5. Simulations

To evaluate the performance and to verify the effectiveness of the proposed algorithms, this section provides two trajectory tracking examples, where one is the trajectory tracking of a robotic manipulator and the other one is the trajectory tracking of an unmanned quadrotor. All simulations are conducted on a computer with 2.6 GHz Intel(R) Core(TM) i7-5600U CPU and MATLAB R2015b.

5.1. Trajectory Tracking of Robotic Manipulator

First, we consider the trajectory of a Puma 560 robot arm manipulator in x-y-z plane with 6 degrees of freedom (DoFs), which is shown in Figure 2. The Puma 560 robot was designed to have approximately the dimensions and reach of a human worker. It also had a spherical joint at the wrist, just as humans have. Roboticists use like waist, shoulder, elbow,

and wrist when describing serial link manipulators. For the Puma, these terms correspond respectively to joints 1, 2, 3, and 4–6, which is shown in Figure 2.

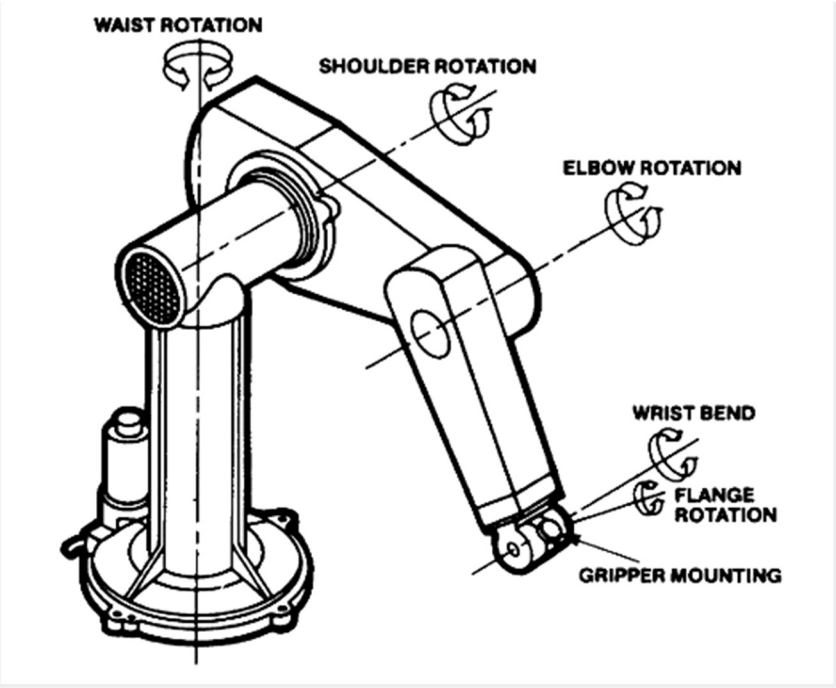


Figure 2. The diagram of the Puma 560 robot arm manipulator (6 DoFs).

For the considered robot arm, $\tau 1$, $\tau 2$, and $\tau 3$ are the control torques of the motors controlling the joint angles ϕ, θ, ψ. The trajectory of the robotic manipulator can be controlled by these torques. The motion can be described by the following Lagrangian system [1]

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) + \kappa(\breve{\mathbf{q}}) = \boldsymbol{\tau}, \tag{34}$$

where $\mathbf{q}$ denotes the generalized coordinates with their time derivatives $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$ and $\boldsymbol{\tau}$ denotes the generalized input. $\mathbf{H}$ is the mass matrix, $\mathbf{C}$ is the Coriolis matrix and $\mathbf{G}$ is the potential energy matrix. An additional unknown dynamic $\kappa(\breve{\mathbf{q}})$(train to obtain its form), which depends on $\breve{\mathbf{q}} = \left[\ddot{\mathbf{q}}^T, \dot{\mathbf{q}}^T, \mathbf{q}^T\right]^T$, affects the system as a generalized force, in which one can refer to [2] for details. The process methodology is illustrated in Figure 3.

Tracking error is the error between the actual value of joint angle or velocity with the desired values

$$\begin{aligned} \tilde{\mathbf{q}} &= \begin{bmatrix} \phi - \phi_d \\ \theta - \theta_d \\ \psi - \psi_d \end{bmatrix} = \mathbf{q}(t) - \mathbf{q}_d(t), \\ \dot{\tilde{\mathbf{q}}} &= \dot{\mathbf{q}}(t) - \dot{\mathbf{q}}_d(t). \end{aligned} \tag{35}$$

The following controllers are tested. (1) Computed torque (CT) controller: $\boldsymbol{\tau}_{in} = \mathbf{H}(\mathbf{q})\ddot{\mathbf{q}}_d + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}_d + \mathbf{G}(\mathbf{q}) - K_p\left(\tilde{\mathbf{q}}\right) - K_d\left(\dot{\tilde{\mathbf{q}}}\right)$. The gains for this controller is K_p = 50 and K_d = 40. (2) The proposed PD controller (24): the composite error is $S = \dot{\tilde{\mathbf{q}}} + \lambda\tilde{\mathbf{q}}$, a reference velocity $\dot{\mathbf{q}}_r$ is $\dot{\mathbf{q}}_r = \dot{\mathbf{q}}_d - \lambda\tilde{\mathbf{q}}\tilde{\mathbf{q}} = \mathbf{q}(t) - \mathbf{q}_d(t)$, and the control torque is $\boldsymbol{\tau}_{in} = \mathbf{H}(\mathbf{q})\dot{\mathbf{q}}_r + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}_r + \mathbf{G}(\mathbf{q}) - K_p\left(\tilde{\mathbf{q}}\right) - K_d S$. The gains for this controller are λ = 30 and K_d = 20. (3) The adaptive controller: the control torque is $\boldsymbol{\tau}_{in}$=$\mathbf{Y}_0 + \mathbf{Y}_1\hat{\mathbf{m}}_3 + \mathbf{Y}_2\hat{\mathbf{I}}_{\mathbf{xx}_3} + \mathbf{Y}_3\hat{\mathbf{I}}_{\mathbf{yy}_3} + \mathbf{Y}_4\hat{\mathbf{I}}_{\mathbf{zz}_3} - K_d\mathbf{S}$, where $\hat{\mathbf{m}}_3$, $\hat{\mathbf{I}}_{\mathbf{xx}_3}$, $\hat{\mathbf{I}}_{\mathbf{yy}_3}$, $\hat{\mathbf{I}}_{\mathbf{zz}_3}$ are the unknown parameters. $\mathbf{Y}_0$ $\mathbf{Y}_1$ $\mathbf{Y}_2$ $\mathbf{Y}_3$ $\mathbf{Y}_4$ are called the regressor vectors defined as [70]. $\hat{\mathbf{m}}_3 = -\left(\mathbf{S}^T\mathbf{Y}_1\right)/\gamma_1$,

$\hat{\mathbf{I}}_{\mathbf{xx}_3} = -\left(\mathbf{S}^T\mathbf{Y}_2\right)/\gamma_2$, $\hat{\mathbf{I}}_{\mathbf{yy}_3} = -\left(\mathbf{S}^T\mathbf{Y}_3\right)/\gamma_3$, $\hat{\mathbf{I}}_{\mathbf{zz}_3} = -\left(\mathbf{S}^T\mathbf{Y}_4\right)/\gamma_4$, where γ_1, γ_2, γ_3, γ_4 are controller gains, in addition to $\gamma_1 = 50$, $\gamma_2 = 30$, $\gamma_3 = 20$, $\gamma_4 = 50$, and $K_d = 20$.

Data. Speeds: 5 s, 10 s, 15 s, and 20 s completion times; 4 paths $\times$ 4 speeds with 16 different trajectories; 15 loads (0.2 kg ... 3.0 kg), various shapes and sizes; 10 agents. Training Data. One desired trajectory common to handling of all loads; one trajectory has no data for any context; sixteen unique training trajectories, one for each load. Test Data. Interpolation data sets for testing on reference trajectory and the unique trajectory for each load. Extrapolation data sets for testing on all trajectories.

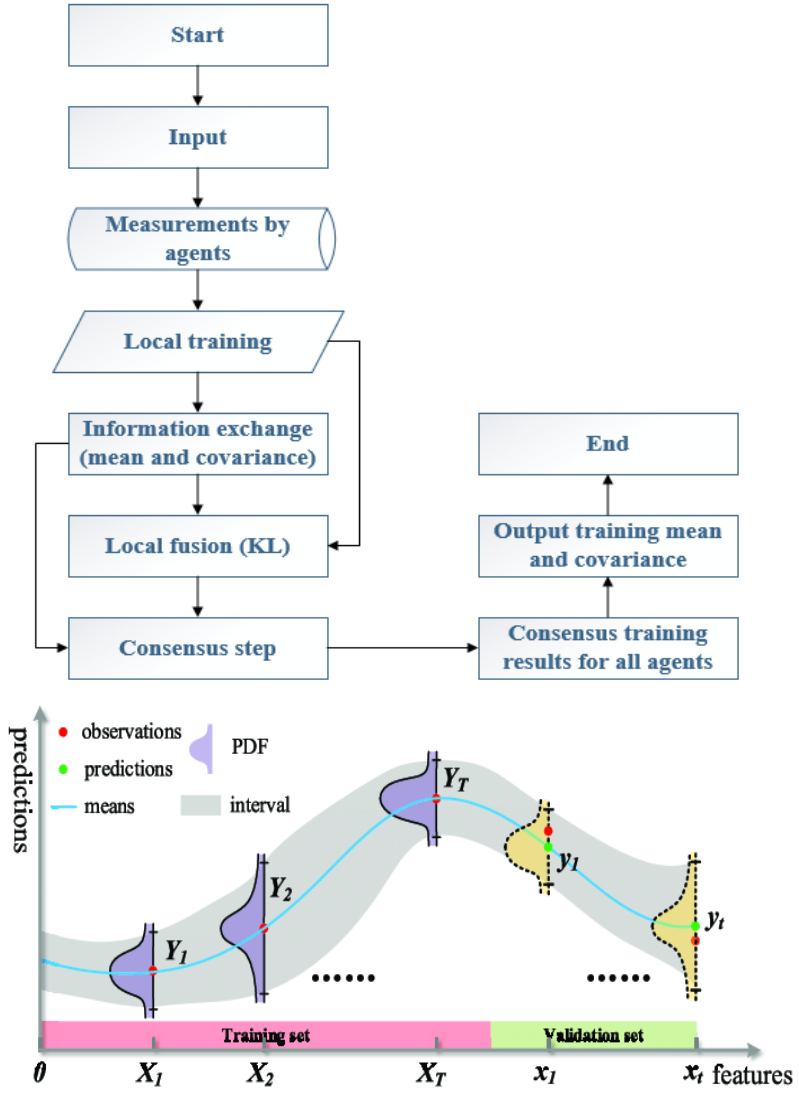


Figure 3. The process methodology and flowchart.

From Figure 4, it can be seen that the PD controller action is able to hold the position of the robot arm at the desired joint angles. $\lambda = 30$ and $K_d = 20$ are the gains associated with holding the respective positions necessary. The convergence of the plots is achieved in about 10 s. The first couple runs of the robot can be used for tuning the robot, and the robot should have good repeatability after that. In addition, from the position and velocity plots of a computed torque controller (Figure 5) and adaptive controller (Figure 6), it is observed that both controllers are able to achieve convergence of parameters to the desired values. However, the proposed PD torque controller has a quicker convergence time and also has lesser gains compared to the computed torque controller. The error results from Table 1 further verify the effectiveness. Therefore, the proposed PD torque controller is

better for the considered application. In addition, we can also obtain that the distributed GP can effectively eliminate the uncertainty and disturbance caused by the system model and the noises.

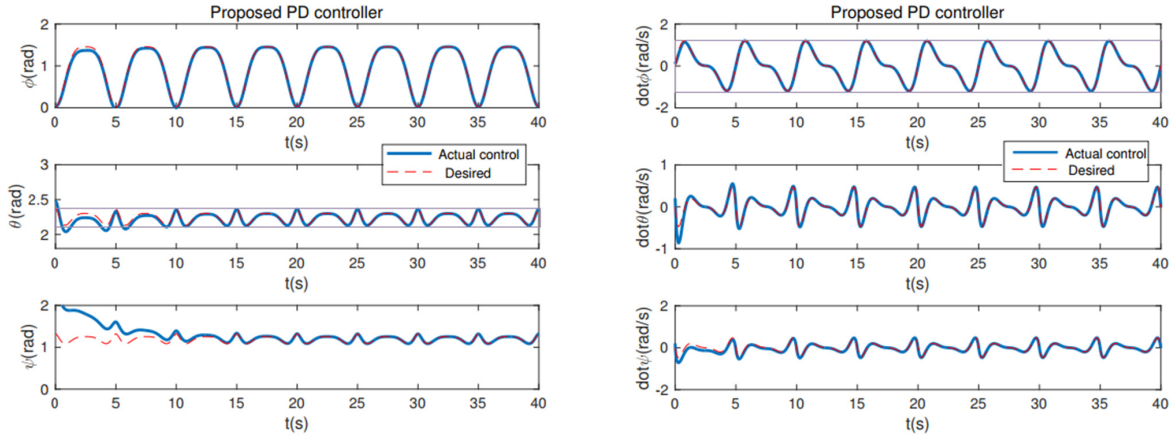


Figure 4. The proposed PD controller: (**a**) Position Plot; (**b**) Velocity Plot.

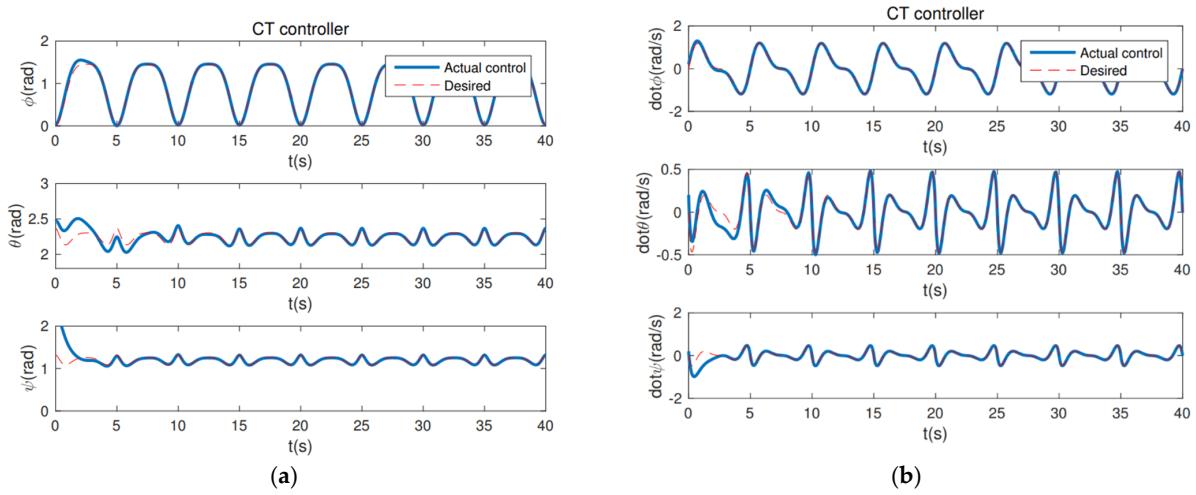


Figure 5. The CT controller: (**a**) Position Plot; (**b**) Velocity Plot.

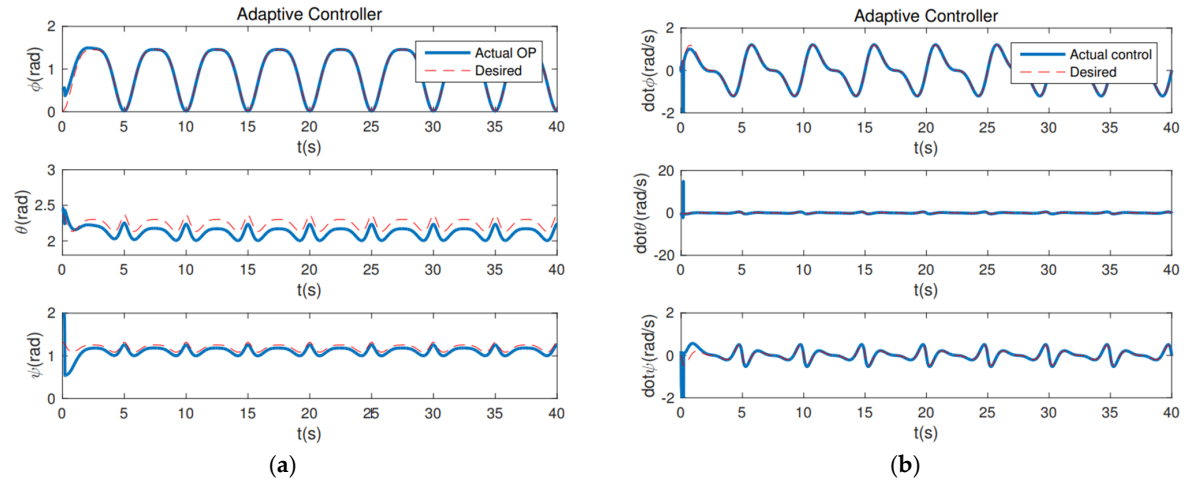


Figure 6. The Adaptive controller: (**a**) Position Plot; (**b**) Velocity Plot.

Table 1. Mean absolute error.

Controller	ϕ	θ	ψ	$\dot{\emptyset}$	$\dot{\theta}$	$\dot{\psi}$
CT	0.0077	0.0284	0.0488	0.0135	0.0239	0.0289
Adaptive	0.0216	0.1226	0.1009	0.0378	0.0451	0.1969
The Proposed PD	0.0209	0.0205	0.1392	0.0072	0.0137	0.0349

To further compare with the multiple agent processing methods, these approaches are tested: (1) Independent GP (IGP): model trained independently for each input [6]; (2) Combined GP (CGP): one agent to train GP by combining data across inputs [34]; (3) Proposed distributed GP with BIC (Bayesian Information Criterion) criterion. The training results (interpolation and extrapolation manners) of NMSE (normalized mean square error) with regard to the number of training data points are demonstrated in Figure 7. Note that IGP and CGP, i.e., existing GP, are centralized methods, which are different from the proposed GP, which is a distributed method for training. From Figure 7, the first line displays the training results of the interpolation manner for the three methods. As we can see from it, for joint 1, the proposed distributed GP achieves the best performance for any number of training points. For joint four and joint six, the performance of the proposed GP is close to IGP and also better than CGP when the training points increase. The second line displays the training results of the extrapolation manner for the three methods. As we can see from it, for joint 1, the proposed distributed GP achieves the best performance for small numbers of training points (<500). However, when training points are increased further, the CGP is better than the proposed distributed GP (note that they are very close). For joint four and joint six, the performance of the proposed GP is close to IGP and also better than CGP when the training points increase. To sum up, the proposed distributed GP model can reach the performance of the centralized method and is close to (even outperforms) the existing state-of-the-art centralized-based multiple combined and multi-task methods.

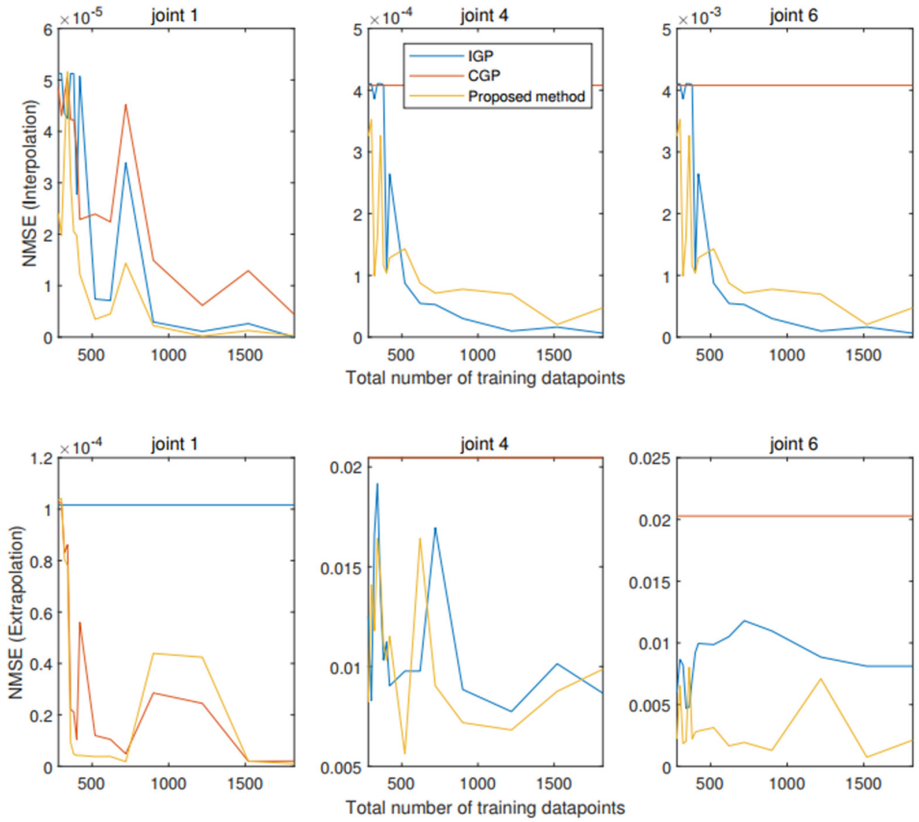


Figure 7. Training results using different methods.

5.2. Trajectory Tracking of an Unmanned Quadrotor

This section tests the proposed distributed GP-based model predictive control (GPMPC) of an unmanned quadrotor. The trajectory of an unmanned quadrotor is generated by a discrete-time Euler–Lagrange dynamical system [2]. The goal is to track its positions (X, Y, Z) and Euler angles (ϕ, θ, ψ). To compare with the state-of-the-art controllers, the efficient MPC (EMPC) [71] and the efficient nonlinear MPC (ENMPC) [72] are also tested in the simulations. The parameters are selected as $L = 5$ and $\mathbf{Q} = \mathbf{R} = diag(1, 1, 1)$.

In the first scenario, the unmanned quadrotor tracks a "Lorenz" trajectory with Gaussian white noise (zero mean and unit variance), which is shown in Figure 8. To train the system model, we use the efficient MPC design proposed in [71]. One hundred seventy measurements, states, and controls are used to train the GP. The datum from the rotational system is with the range [0, 1], the angle φ is with a range $[-1.6,\ 1.6]$, and the input is with the range $\left[-4 \times 10^{-8},\ 7 \times 10^{-8}\right]$. The training of GP takes 5 s, and we use 10 agents to train. The values of mean squared error (MSE) trained by GP are very small. The mean squared error (MSE) for the positions is 4.3618×10^4 close to the stable GPMPC (GPMPC1) [73] with 4.3498×10^4; MSE for the angels is 1.5743×10^8 also close to GPMPC1 with 4.3030×10^9. This indicates that the proposed distributed GP (GPMPC2) is efficient and well-trained, which is illustrated in Figure 9 (with different confidences). Note that the stable GPMPC (GPMPC1) is the most recent best method at present and is also a method for one agent. Therefore, the training results are very close, which indicates that the proposed distributed GP can achieve a good training performance.

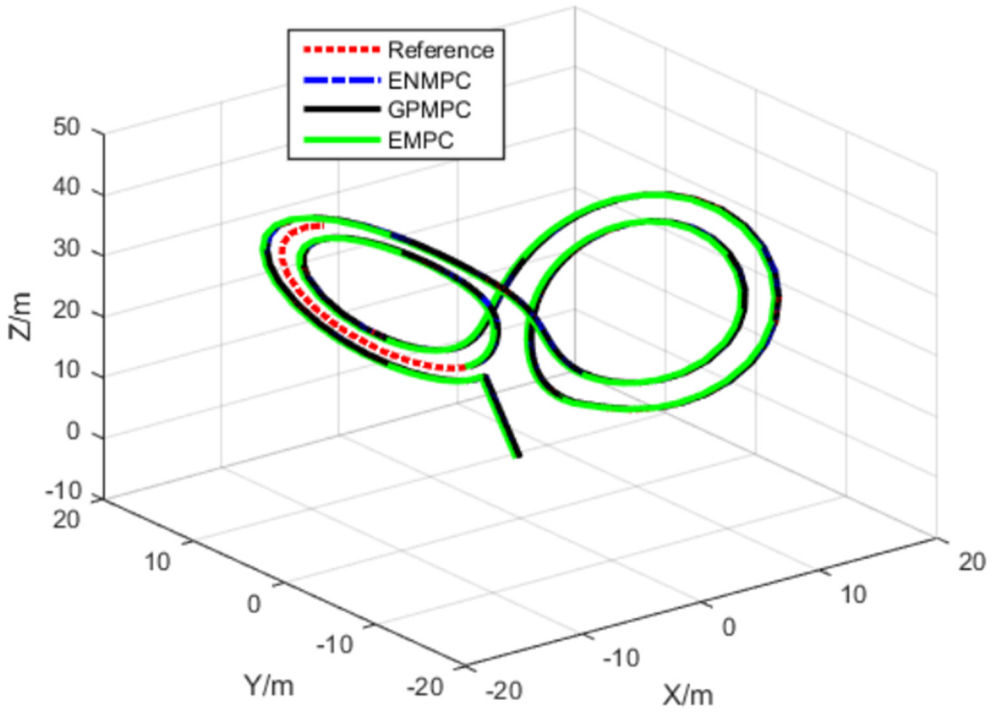


Figure 8. Lorenz trajectory tracking.

The positions and attitudes tracking results are demonstrated in Figure 10, and the tracking errors are displayed in Figure 11 and Table 2. As we can see from Figures 10 and 11, the proposed distributed GP can learn the system model well and track the trajectories with high precision, which is close to the state-of-the-art controllers. This also indicates that as long as the training sets are introduced, we can track the trajectory without model knowledge (model-free), i.e., the proposed GP can learn the system model well. In addition, as long as multiple agents are introduced, the model uncertainties and noise disturbances can be eliminated and suppressed.

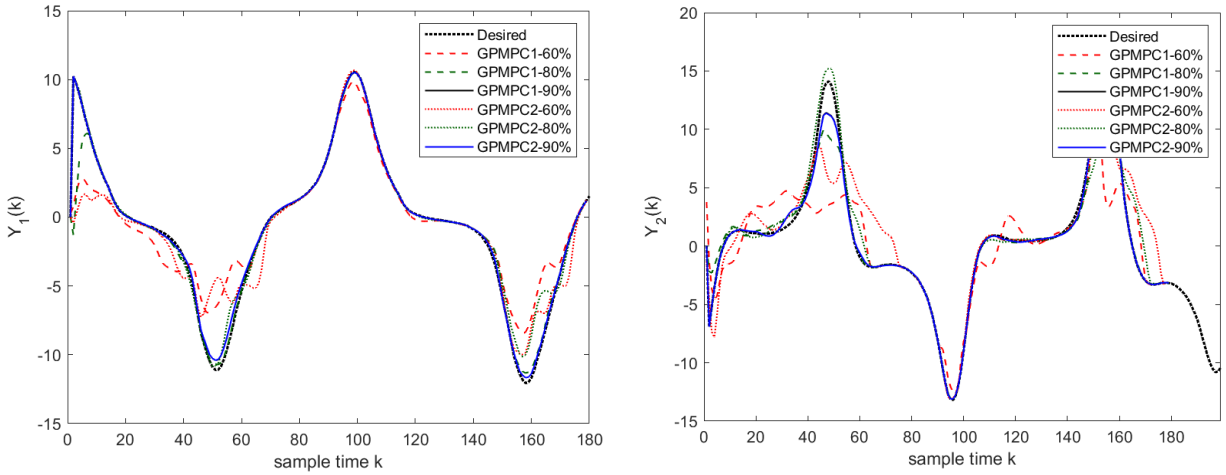


Figure 9. Training performance with 60%, 80%, and 90% confidence: (**a**) $Y_1(k)$; (**b**) $Y_2(k)$.

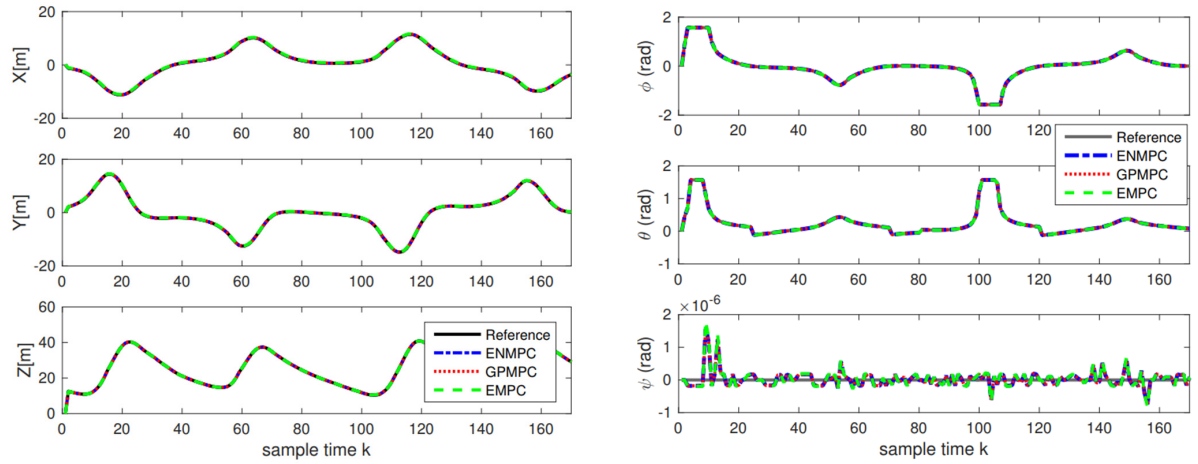


Figure 10. Tracking results: (**a**) Positions tracking; (**b**) Attitudes tracking.

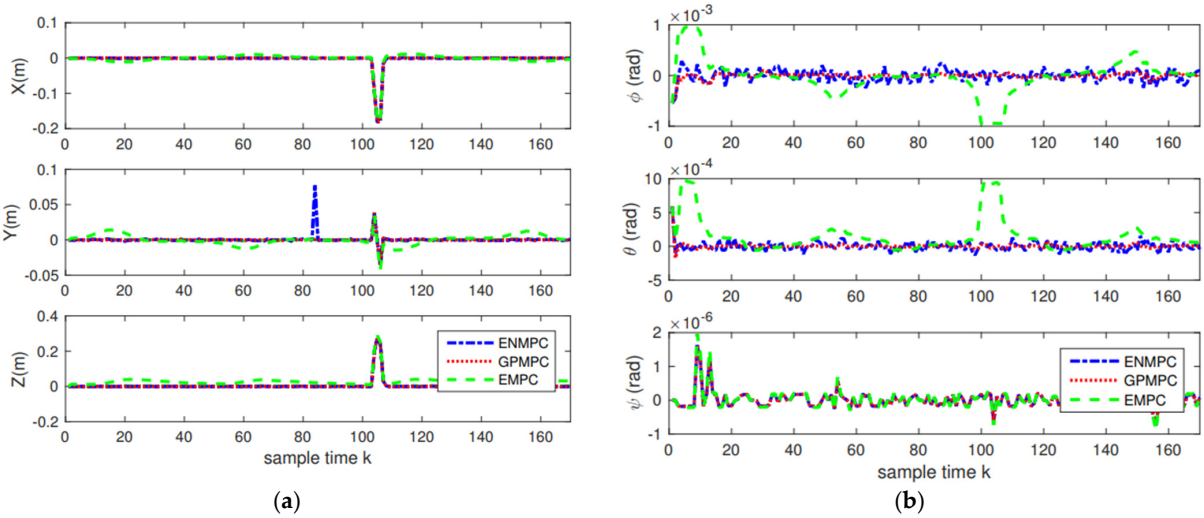
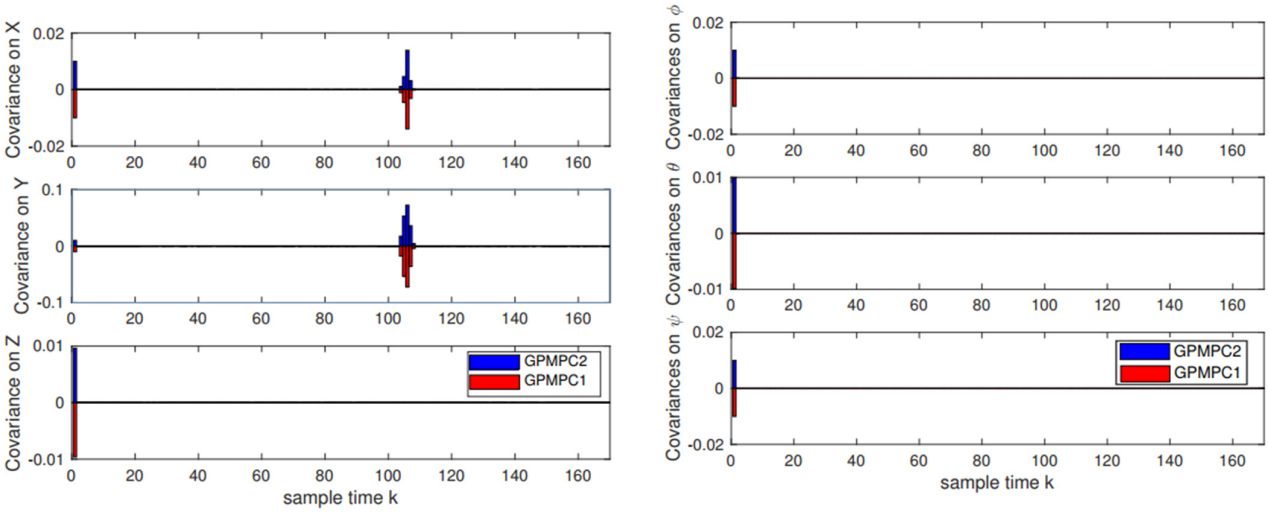


Figure 11. Tracking errors: (**a**) Positions errors; (**b**) Attitudes errors.

Table 2. Training errors and tracking errors.

Training Methods\Training Errors	**Positions**	**Attitudes**
GPMPC1	4.3496×10^{-4}	4.3030×10^{-9}
GPMPC2	4.3618×10^{-4}	1.5746×10^{-8}
Controllers\Mean Absolute Errors	**Positions**	**Attitudes**
EMPC	0.0287	9.6239×10^{-11}
ENMPC	0.0050	7.6412×10^{-11}
GPMPC	0.0049	2.2407×10^{-11}

Furthermore, the covariance on positions and attitudes by different GP models (the stable GPMPC (GPMPC1) [69] and the proposed distributed GPMPC (GPMPC2)) is displayed in Figure 12. This indicates that the proposed distributed GP can also reach the performance of the state-of-the-art GP model.



(**a**) (**b**)

Figure 12. Covariance results: (**a**) Positions covariance; (**b**) Attitudes covariance.

In the second scenario, the unmanned quadrotor tracks an "Elliptical" trajectory with Gaussian white noise (zero mean and unit variance), which is shown in Figure 13. The tracking performance and the tracking errors are shown in Figures 14 and 15 and Table 3. From Figures 13–15, we can also see that the proposed distributed GP can learn the trajectory model effectively, which is very close to the desired reference trajectory and is close to the state-of-the-art controllers. The covariance results by GPMPC1 and GPMPC2 are shown in Figure 16, which further verifies the effectiveness of the proposed distributed GP.

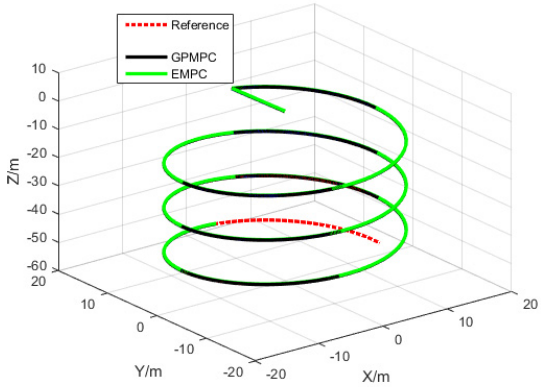
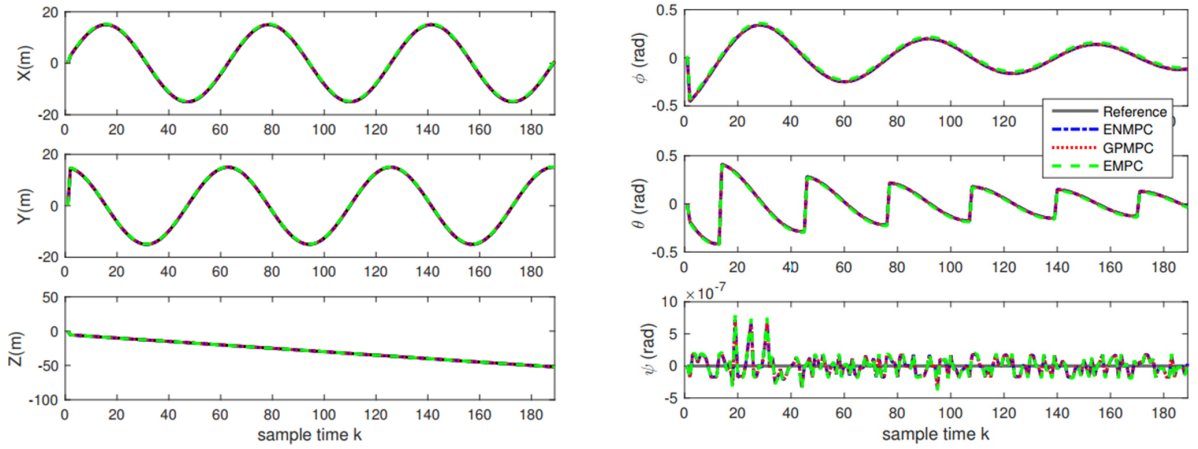
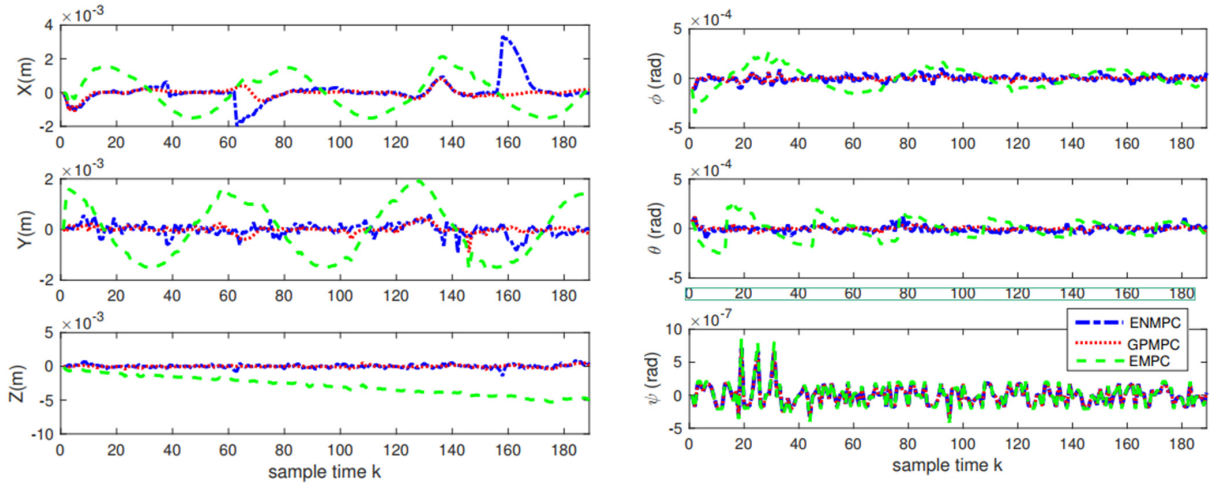


Figure 13. Elliptical trajectory tracking.



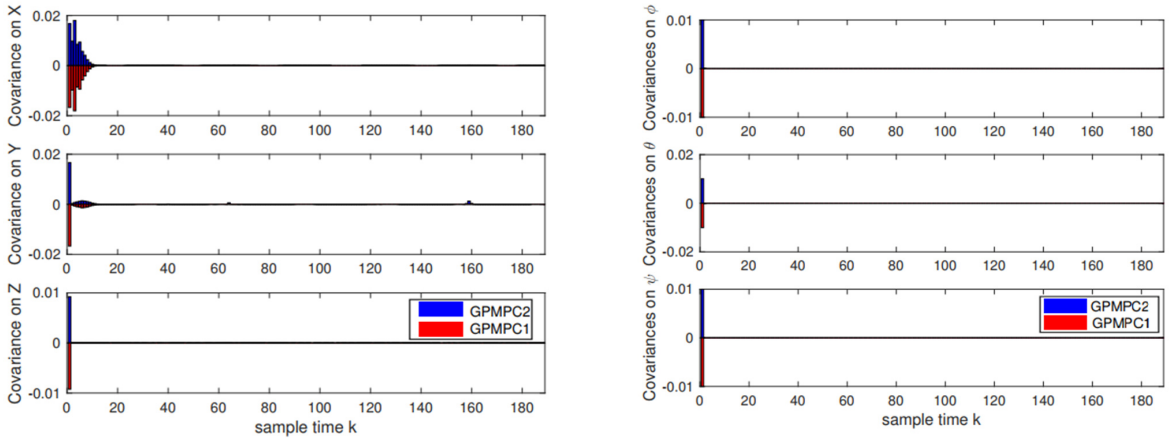
(**a**) (**b**)

Figure 14. Tracking results: (**a**) Positions tracking; (**b**) Attitudes tracking.



(**a**) (**b**)

Figure 15. Tracking errors: (**a**) Positions errors; (**b**) Attitudes errors.



(**a**) (**b**)

Figure 16. Covariance results: (**a**) Positions covariance; (**b**) Attitudes covariance.

Table 3. Training errors and tracking errors.

Training Methods\Training Errors	**Positions**	**Attitudes**
GPMPC1	6.1494×10^{-8}	3.8282×10^{-10}
GPMPC2	5.1161×10^{-7}	1.1889×10^{-9}
Controllers\Mean Absolute Errors	**Positions**	**Attitudes**
EMPC	0.0287	0.0263
ENMPC	0.0050	1.8769×10^{-4}
GPMPC	0.0049	8.4033×10^{-5}

6. Conclusions

This paper used the Gaussian process to learn the trajectory model, and a distributed GP-based model learning strategy was proposed. For the continuous- and discrete-time system, we, respectively, designed a GP-based PD controller and a GP-based MPC controller to address the problem. To address the uncertainties of the model and the disturbances of the noises, a distributed multiple-agent system was used to train the model. In addition, since data-driven algorithms needed a large number of training sets, the distributed GP model could also be employed to address this problem by using a Kullback–Leibler average consensus fusion criterion.

The proposed GP can solve the actual model-free problem as long as the training data sets are given. Since the considered multi-agent is interconnected and it is only used to eliminate the uncertainties of the model and disturbances of the noises, future research mainly focuses on the efficiency of distributed Gaussian process and the robustness of the multi-agent network. In the future, we will focus on the application deployment of an unmanned aerial vehicle (UAV) and its usage in UAV detection and location. UAV racing is a challenging problem to overcome.

Author Contributions: Conceptualization, L.S. and D.X.; methodology, D.X.; software, D.X.; validation, D.X.; formal analysis, L.S. and D.X.; investigation, L.S. and D.X.; resources, D.X.; data curation, D.X.; writing—original draft preparation, D.X.; writing—review and editing, L.S. and D.X.; supervision, L.S.; funding acquisition, L.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Shaanxi Provincial Fund under Grant 2020JM-185 and the National Natural Science Foundation of China under grant 62171338.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A Proof of Theorem 1

Proof. First, we prove the Lipschitz constant of $\mu(\cdot)$ and the modulus of continuity of $\sum(\cdot)$. At two different states $\mathbf{x}$ and $\mathbf{x}^{'}$, the norm of the difference of $\mu(\cdot)$ is $\parallel \mu(\mathbf{x}) - \mu(\mathbf{x}') \parallel = \parallel (K_{\mathbf{x}\mathbf{X}} - K_{\mathbf{x}'\mathbf{X}})\alpha \parallel$ with $\alpha = \left(K_{\mathbf{X}\mathbf{X}} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{y}$. Then, based on the Lipschitz continuity of the kernel and the Cauchy–Schwarz inequality, we have $\parallel \mu(\mathbf{x}) - \mu(\mathbf{x}') \parallel \leq L_k\sqrt{N} \parallel \alpha \parallel\parallel \mathbf{x} - \mathbf{x}' \parallel$, which proves that μ is Lipschitz continuous.

Similarly, we have

$$\parallel \Sigma(\mathbf{x}) - \Sigma(\mathbf{x}') \parallel \leq 2L_k \parallel \mathbf{x} - \mathbf{x}' \parallel + \parallel K_{\mathbf{x}\mathbf{X}} - K_{\mathbf{x}'\mathbf{X}} \parallel\parallel \left(K_{\mathbf{X}\mathbf{X}} + \sigma^2\mathbf{I}\right)^{-1} \parallel\parallel K_{\mathbf{X}\mathbf{x}} + K_{\mathbf{X}\mathbf{x}'} \parallel .$$

Due to the Lipschitz continuity of kernel k (also K), we have

$$\| K_{\mathbf{x}\mathbf{X}} - K_{\mathbf{x}'\mathbf{X}} \| \leq L_k\sqrt{N} \| \mathbf{x} - \mathbf{x}' \|, \text{and } \| K_{\mathbf{X}\mathbf{x}} + K_{\mathbf{X}\mathbf{x}'} \| \leq 2\sqrt{N} maxk(\mathbf{x}, \mathbf{x}')$$

Therefore, the continuous modulus $\omega_{\sum}$ can be obtained by combing the above equations and taking the square root.

Finally, we prove the probabilistic uniform error bound. According to [74], for every grid X_τ with the $|X_\tau|$-th number of grid points and $\max\limits_{x \in \boldsymbol{X}} \min\limits_{x' \in \boldsymbol{X}_\tau} \| x - x' \| \leq \tau, \forall . \boldsymbol{x} \in \boldsymbol{X}_\tau$, it follows that

$$\|f(\mathbf{x}) - \mu(\mathbf{x}')\| \leq \sqrt{\beta(\tau)\sum(\mathbf{x})}, \tag{36}$$

with a probability of at least $1 - |X_\tau| e^{-\beta(\tau)/2}$. Then, set $\beta(\tau) = 2\log\left(\frac{|\mathbf{X}_\tau|}{\delta}\right)$, then (36) holds with a probability of at least $1 - \delta$. Furthermore, since f, μ, and $\sum$ are continuous, $\forall \mathbf{x} \in \mathbf{X}$, $\forall \mathbf{x}' \in \mathbf{X}_\tau$, it follows $\tau L_f \geq min \| f(\mathbf{x}) - f(\mathbf{x}') \|, \tau L_\mu \geq min \| \mu(\mathbf{x}) - \mu(\mathbf{x}') \|$ and $\omega_{\Sigma(\mathbf{x})} \geq min \| \sqrt{\Sigma(\mathbf{x})} - \sqrt{\Sigma(\mathbf{x}')} \|$. In addition, based on [74], the minimum number of grid points is denoted by the covering number $M(\tau, X)$. Therefore, $\forall \mathbf{x} \in \mathbf{X}$ it follows that $p\left(\| f(\mathbf{x} \|) - \mu(\mathbf{x}) \leq \gamma(\tau) + \sqrt{\beta(\tau)\sum(\mathbf{x})}\right) \geq 1 - \delta$ where $\gamma(\tau) = \left(L_\mu + L_f\right)\tau + \sqrt{\beta(\tau)}\omega_{\sum(\tau)}$ and $\beta(\tau) = 2\log\left(\frac{M(\tau, \mathbf{X})}{\delta}\right)$. This concludes the proof. □

Appendix B Proof of Theorem 2

Proof. According to Theorem 1, given $\beta_N(\tau) = 2log\left(\frac{M(\tau, \boldsymbol{x})\pi^2 N^2}{3\delta}\right)$ and $N > 0$, it follows that

$$\sup_{\mathbf{x} \in \mathbf{X}} |f(\mathbf{x}) - \mu_N(\mathbf{x})| \leq \gamma_N(\tau) + \sqrt{\beta_N(\tau)\sum\nolimits_N(\mathbf{x})}. \tag{37}$$

with a probability of at least $1 - \delta/2$. In addition, given the distance $r = \max\limits_{\mathbf{x}, \mathbf{x}' \in \mathbf{X}} \| \mathbf{x} - \mathbf{x}' \|$, we can obtain a trivial bound $M(\tau, \boldsymbol{X}) \leq \left(1 + \frac{r}{\tau}\right)^n$. Then, we can obtain that $\beta_N(\tau) \leq 2n\log\left(1 + \frac{r}{\tau}\right) + 4\log \pi N - 2\log 3\delta$.

On the one hand, Lipschitz constant L_μ is bounded by

$$L_\mu \leq L_k\sqrt{N}\left(K_{X_N X_N} + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y}_N.$$

On the other hand, since f is bounded by $\| f^{max} \|$ and $K_{\mathbf{X}_N \mathbf{X}_N}$ is positive semi-definite, $\| \left(K_{X_N X_N} + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y}_N \|$ is bounded by

$$\| \left(K_{X_N X_N} + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y}_N \| \leq \frac{\| \boldsymbol{y}_N \|}{\rho\left(K_{X_N X_N} + \sigma^2 \boldsymbol{I}\right)_{min} \frac{\sqrt{N} f^{max} + \| \varphi_N \|}{\sigma^2}}$$

where vector φ_N composed of N variables is a Gaussian disturbance with zero mean and covariance σ^2. This indicates that $\frac{\| \varphi_N \|^2}{\sigma^2}$ obeys a chi-square distribution, i.e., $\frac{\| \varphi_N \|^2}{\sigma^2} \sim \chi^2_N$. Note that with a probability of at least $1 - \eta_N$ we have

$$\| \varphi_N \|^2 \leq \left(2\sqrt{N\eta_N} + 2\eta_N + N\right)\sigma^2.$$

Then, by using the union bounds over all $N > 0$ and setting $\eta_N = \log\left(\frac{\pi^2 N^2}{3\delta}\right)$, we can obtain that

$$\| \left(K_{X_N X_N} + \sigma^2 \boldsymbol{I}_N\right)^{-1} \boldsymbol{y}_N \| \leq \frac{\sqrt{N} \| f^{max} \| + \sqrt{2\sqrt{N\eta_N} + 2\eta_N + N}\sigma}{\sigma^2}$$

with a probability of at least $1-\delta/2$. Therefore, the Lipschitz constant L_μ of the posterior mean function $u_N(\cdot)$ has

$$L_\mu \leq \frac{\sqrt{N} \parallel f^{max} \parallel + \sqrt{N\left(2\sqrt{N\eta_N} + 2\eta_N + N\right)}\sigma}{\sigma^2}$$

In addition, since ηN is logarithmically increased with the number of training samples N, it follows that $L_\mu \in \mathcal{O}(\mathrm{N})$ with a probability of at least $1-\delta/2$.

Furthermore, based on (17) and $\parallel \left(K_{X_N X_N} + \sigma^2 \mathbf{I}_N\right)^{-1} \parallel$, we can bound the modulus of continuity $\omega_{\sum_N}(\cdot)$ as

$$\omega_{\sum(\tau)} \leq \sqrt{2\tau L_k \left(1 + \frac{N \underset{\boldsymbol{x},\boldsymbol{x}' \in \boldsymbol{X}}{max}\, k(\boldsymbol{x},\boldsymbol{x}')}{\sigma^2}\right)}$$

According to (37), the uniform bound holds with a probability of at least $1-\delta$ with

$$\gamma_N(\tau) \leq \sqrt{2\tau L_k \beta_N(\tau) \left(1 + \frac{N \underset{\boldsymbol{x},\boldsymbol{x}' \in \boldsymbol{X}}{max}\, k(\boldsymbol{x},\boldsymbol{x}')}{\sigma^2}\right)} + L_f \tau + L_k \frac{\sqrt{N} \parallel f^{max} \parallel + \sqrt{N\left(2\sqrt{N\eta_N} + 2\eta_N + N\right)}\sigma}{\sigma^2}$$

Note that if the error is designed to vanish, the above equation should be guaranteed convergence to 0 as $N \to \infty$. This indicates that $\tau(N)$ decreases faster than $\mathcal{O}\left((\mathrm{Nlog}(N))^{-1}\right)$. Therefore, we can set $\tau(N) \in \mathcal{O}\left(N^{-2}\right)$, then we can obtain that $\underset{N\to\infty}{lim}\, \gamma_N(\tau_N) = 0$. Furthermore, this indicates that $\beta_N(\tau(N)) \in \mathcal{O}(\log(N))$. Therefore, there exists an $\epsilon > 0$ such that $\sum_N(x) \in \mathcal{O}\left(\log(N)^{-\frac{1}{2}-\epsilon}\right)$, it follows that $\sqrt{\beta_N(\tau(N))\sum_N(x)} \in \mathcal{O}(\log(N)^{-\epsilon})$, which concludes the proof. □

Appendix C Proof of Theorem 3

Proof. According to Lemma 2 and 3, and recalling that the noise w is the stationary Gaussian process, we use the following Lyapunov candidate $\dot{\mathbf{V}}(x) = 1/2r^2$. $\forall |r| > \frac{f(\mathbf{x})-\mu_N(\mathbf{x})}{k_c}$, it follows that $\dot{\mathbf{V}}(\mathbf{x}) = \frac{\partial V}{\partial r}\dot{r} = r\left(f(\mathbf{x}) - \hat{f}(\mathbf{x}) - k_c r\right) \leq |r||f(\mathbf{x}) - \mu_N(\mathbf{x})| - k_c|r|^2 \leq 0$. According to Theorem 1, the model error is bounded with probability $p\left(\dot{\mathbf{V}}(\mathbf{x}) < 0\right) \geq 1-\delta$. According to Lemma 3, we can obtain the global ultimate boundedness. □

References

1. Beckers, T.; Umlauft, J.; Kulic, D.; Hirche, S. Stable Gaussian process based tracking control of Lagrangian systems. In Proceedings of the 2017 IEEE 56th Annual Conference on Decision and Control (CDC), Melbourne, Australia, 12–15 December 2017; pp. 5180–5185.
2. Corke, P.I.; Khatib, O. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*; Springer: Berlin/Heidelberg, Germany, 2011; Volume 73.
3. Wang, D.; Mu, C. Adaptive-critic-based robust trajectory tracking of uncertain dynamics and its application to a spring-mass-damper system. *IEEE Trans. Ind. Electron.* **2018**, *65*, 654–663. [CrossRef]
4. Choi, J.; Jung, J.; Park, I. Area-efficient approach for generating quantized gaussian noise. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2016**, *63*, 1005–1013. [CrossRef]
5. Khansari-Zadeh, S.M.; Billard, A. Learning stable nonlinear dynamical systems with Gaussian mixture models. *IEEE Trans. Robot.* **2011**, *27*, 943–957. [CrossRef]
6. Choi, J. Data-aided sensing for Gaussian process regression in iot systems. *IEEE Internet Things* **2021**, *8*, 7717–7726. [CrossRef]
7. Diaz-Rozo, J.; Bielza, C.; Larranaga, P. Clustering of data streams with dynamic Gaussian Mixture Models: An IoT application in industrial processes. *IEEE Internet Things J.* **2018**, *5*, 3533–3547. [CrossRef]
8. Sheng, H.; Xiao, J.; Cheng, Y.; Ni, Q.; Wang, S. Short-term solar power forecasting based on weighted Gaussian process regression. *IEEE Trans. Ind. Electron.* **2018**, *65*, 300–308. [CrossRef]

9. Wen, Y.; Li, G.; Wang, Q.; Guo, X.; Cao, W. Modeling and analysis of permanent magnet spherical motors by a multi-task Gaussian process method and finite element method for output torque. *IEEE Trans. Ind. Electron.* **2021**, *68*, 8540–8549. [CrossRef]
10. Jin, X. Fault tolerant nonrepetitive trajectory tracking for mimo output constrained nonlinear systems using iterative learning control. *IEEE Trans. Cybern.* **2019**, *49*, 3180–3190. [CrossRef] [PubMed]
11. Fedele, G.; D'Alfonso, L. A kinematic model for swarm finite-time trajectory tracking. *IEEE Trans. Cybern.* **2019**, *49*, 3806–3815. [CrossRef]
12. Wilson, A.G.; Knowles, D.A.; Ghahramani, Z. Gaussian process regression networks. In Proceedings of the 29th International Conference on Machine Learning, Edinburgh, UK, 26 June–1 July 2012.
13. Pillonetto, G.; Schenato, L.; Varagnolo, D. Distributed multi-agent gaussian regression via finite-dimensional approximations. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *41*, 2098–2111. [CrossRef]
14. Varagnolo, D.; Pillonetto, G.; Schenato, L. Distributed parametric and nonparametric regression with on-line performance bounds computation. *Automatica* **2012**, *48*, 2468–2481. [CrossRef]
15. Krivec, T.; Papa, G.; Kocijan, J. Simulation of variational Gaussian process NARX models with GPGPU. *ISA Trans.* **2021**, *109*, 141–151. [CrossRef]
16. Aman, K.; Kocijan, J. Application of Gaussian processes for black-box modelling of biosystems. *ISA Trans.* **2007**, *46*, 443–457. [CrossRef] [PubMed]
17. Hensman, J.; Durrande, N.; Solin, A. Variational fourier features for Gaussian processes. *J. Mach. Learn. Res.* **2017**, *18*, 5537–5588.
18. Damianou, A.C.; Titsias, M.K.; Lawrence, N.D. Variational inference for latent variables and uncertain inputs in Gaussian processes. *J. Mach. Learn. Res.* **2016**, *17*, 1425–1486.
19. Meng, Z.; Lin, Z.; Ren, W. Robust cooperative tracking for multiple non-identical second-order nonlinear systems. *Automatica* **2013**, *49*, 2363–2372. [CrossRef]
20. Pu, S.; Yu, X.; Li, J. Distributed Kalman filter for linear system with complex multichannel stochastic uncertain parameter and decoupled local filters. *Int. J. Adapt. Control. Signal Process.* **2021**, *35*, 1498–1512. [CrossRef]
21. Yu, X.; Li, J. Adaptive Kalman filtering for recursive both additive noise and multiplicative noise. *IEEE Trans. Aerosp. Electron. Syst.* **2022**, *58*, 1634–1649. [CrossRef]
22. Huang, Y.; Meng, Z. Bearing-based distributed formation control of multiple vertical take-off and landing UAVs. *IEEE Trans. Control. Netw. Syst.* **2021**, *8*, 1281–1292. [CrossRef]
23. Yang, T.; Yi, X.; Wu, J.; Yuan, Y.; Wu, D.; Meng, Z.; Hong, Y.; Wang, H.; Lin, Z.; Johansson, K.H. A survey of distributed optimization. *Annu. Rev. Control.* **2019**, *47*, 278–305. [CrossRef]
24. Li, X.; Caimou, H.; Haoji, H. Distributed filter with consensus strategies for sensor networks. *J. Appl. Math.* **2013**, *2013*, 683249. [CrossRef]
25. Zhou, T. Coordinated one-step optimal distributed state prediction for a networked dynamical system. *IEEE Trans. Autom. Control.* **2013**, *58*, 2756–2771. [CrossRef]
26. Battistelli, G.; Chisci, L. Kullback-Leibler average, consensus on probability densities, and distributed state estimation with guaranteed stability. *Automatica* **2014**, *50*, 707–718. [CrossRef]
27. Umlauft, J.; Lederer, A.; Hirche, S. Learning stable Gaussian process state space models. In Proceedings of the 2017 American Control Conference (ACC), Seattle, DC, USA, 24–26 May 2017; pp. 1499–1504.
28. Jagtap, P.; Pappas, G.J.; Zamani, M. Control barrier functions for unknown nonlinear systems using Gaussian processes. In Proceedings of the 2020 59th IEEE Conference on Decision and Control (CDC), Jeju Island, Korea, 14–18 December 2020; pp. 3699–3704.
29. Pöhler, L.D.; Umlauft, J.; Hirche, S. Uncertainty-based Human Motion Tracking with Stable Gaussian Process State Space Models. *IFAC-Pap.* **2019**, *51*, 8–14. [CrossRef]
30. Umlauft, J.; Pöhler, L.D.; Hirche, S. An uncertainty-based control Lyapunov approach for control-affine systems modeled by Gaussian process. *IEEE Control. Syst. Lett.* **2018**, *2*, 483–488. [CrossRef]
31. Lederer, A.; Umlauft, J.; Hirche, S. Uniform error bounds for Gaussian process regression with application to safe control. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 659–669.
32. Deisenroth, M.; Ng, J.W. Distributed Gaussian processes. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 7–9 July 2015; pp. 1481–1490.
33. Xie, A.; Yin, F.; Xu, Y.; Ai, B.; Chen, T.; Cui, S. Distributed Gaussian processes hyperparameter optimization for big data using proximal ADMM. *IEEE Signal Processing Lett.* **2019**, *26*, 1197–1201. [CrossRef]
34. Bonilla, E.V.; Chai, K.M.; Williams, C. Multi-task Gaussian process prediction. In Proceedings of the Advances in Neural Information Processing Systems 20 (NIPS 2007), Vancouver, BC, Canada, 3–5 December 2008; pp. 153–160.
35. Alvarez, M.; Lawrence, N.D. Sparse convolved Gaussian processes for multi-output regression. In Proceedings of the Advances in Neural Information Processing Systems 21 (NIPS 2008), Vancouver, BC, Canada, 8 December 2009.
36. Gal, Y.; van der Wilk, M.; Rasmussen, C.E. Distributed variational inference in sparse Gaussian process regression and latent variable models. In Proceedings of the Advances in Neural Information Processing Systems 27 (NIPS 2014), Montreal, Canada, 8–13 December 2014; pp. 3257–3265.

37. Nerurkar, E.D.; Roumeliotis, S.I.; Martinelli, A. Distributed maximum a posteriori estimation for multi-robot cooperative localization. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 6 July 2009; pp. 1402–1409.
38. Franceschelli, M.; Gasparri, A. On agreement problems with gossip algorithms in absence of common reference frames. In Proceedings of the 2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, 15 July 2010; pp. 4481–4486.
39. Cunningham, A.; Indelman, V.; Dellaert, F. DDF-SAM 2.0: Consistent distributed smoothing and mapping. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 5220–5227.
40. Anderson, B.D.; Shames, I.; Mao, G.; Fidan, B. Formal theory of noisy sensor network localization. *SIAM J. Discret. Math.* **2010**, *24*, 684–698. [CrossRef]
41. Carron, A.; Todescato, M.; Carli, R.; Schenato, L. An asynchronous consensus-based algorithm for estimation from noisy relative measurements. *IEEE Trans. Control. Netw. Syst.* **2014**, *1*, 283–295. [CrossRef]
42. Thunberg, J.; Montijano, E.; Hu, X. Distributed attitude synchronization control. In Proceedings of the 2011 50th IEEE Conference on Decision and Control and European Control Conference, Orlando, FL, USA, 12–15 December 2011; pp. 1962–1967.
43. Piovan, G.; Shames, I.; Fidan, B.; Bullo, F.; Anderson, B.D. On frame and orientation localization for relative sensing networks. *Automatica* **2013**, *49*, 206–213. [CrossRef]
44. Sarlette, A.; Sepulchre, R. Consensus optimization on manifolds. *SIAM J. Control. Optim.* **2009**, *48*, 56–76. [CrossRef]
45. Choudhary, S.; Carlone, L.; Nieto, C.; Rogers, J.; Christensen, H.I.; Dellaert, F. Distributed trajectory estimation with privacy and communication constraints: A two-stage distributed Gauss-seidel approach. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 5261–5268.
46. Deisenroth, M.P.; Fox, D.; Rasmussen, C.E. Gaussian processes for data-efficient learning in robotics and control. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 408–423. [CrossRef] [PubMed]
47. Robinson, D.R.; Mar, R.T.; Estabridis, K.; Hewer, G. An efficient algorithm for optimal trajectory generation for heterogeneous multi-agent systems in non-convex environments. *IEEE Robot. Autom. Lett.* **2018**, *3*, 1215–1222. [CrossRef]
48. Wang, J.M.; Fleet, D.J.; Hertzmann, A. Gaussian process dynamical models for human motion. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *30*, 283–298. [CrossRef]
49. Negenborn, R.R.; Maestre, J.M. Distributed model predictive control: An overview and roadmap of future research opportunities. *IEEE Control. Syst. Mag.* **2014**, *34*, 87–97.
50. Stewart, B.T.; Wright, S.J.; Rawlings, J.B. Cooperative distributed model predictive control for nonlinear systems. *J. Process Control.* **2011**, *21*, 698–704. [CrossRef]
51. Ferramosca, A.; Limón, D.; Alvarado, I.; Camacho, E.F. Cooperative distributed MPC for tracking. *Automatica* **2013**, *49*, 906–914. [CrossRef]
52. Conte, C.; Jones, C.N.; Morari, M.; Zeilinger, M.N. Distributed synthesis and stability of cooperative distributed model predictive control for linear systems. *Automatica* **2016**, *69*, 117–125. [CrossRef]
53. Groß, D.; Stursberg, O. A cooperative distributed MPC algorithm with event-based communication and parallel optimization. *IEEE Trans. Control. Netw. Syst.* **2015**, *3*, 275–285. [CrossRef]
54. Alrifaee, B.; Heßeler, F.J.; Abel, D. Coordinated non-cooperative distributed model predictive control for decoupled systems using graphs. *IFAC-Pap.* **2016**, *49*, 216–221.
55. Alonso, C.A.; Matni, N. Distributed and localized closed loop model predictive control via system level synthesis. In Proceedings of the 2020 59th IEEE Conference on Decision and Control (CDC), Jeju, Korea, 14–18 December 2020; pp. 5598–5605.
56. Alonso, C.A.; Matni, N.; Anderson, J. Explicit distributed and localized model predictive control via system level synthesis. In Proceedings of the 2020 59th IEEE Conference on Decision and Control (CDC), Jeju, Korea, 14–18 December 2020; pp. 5606–5613.
57. Luis, C.E.; Schoellig, A.P. Trajectory generation for multiagent point-to-point transitions via distributed model predictive control. *IEEE Robot. Autom. Lett.* **2019**, *4*, 375–382. [CrossRef]
58. Torrente, G.; Kaufmann, E.; Föhn, P.; Scaramuzza, D. Data-driven MPC for quadrotors. *IEEE Robot. Autom. Lett.* **2021**, *6*, 3769–3776. [CrossRef]
59. Liu, D.; Tang, M.; Fu, J. Robust adaptive trajectory tracking for wheeled mobile robots based on Gaussian process regression. *Syst. Control. Lett.* **2022**, *163*, 105210. [CrossRef]
60. Akbari, B.; Zhu, H. Tracking Dependent Extended Targets Using Multi-Output Spatiotemporal Gaussian Processes. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 18301–18314. [CrossRef]
61. Hidalgo-Carrió, J.; Hennes, D.; Schwendner, J.; Kirchner, F. Gaussian process estimation of odometry errors for localization and mapping. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 5696–5701.
62. Brossard, M.; Bonnabel, S. Learning wheel odometry and IMU errors for localization. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 291–297.
63. Nguyen, T.V.; Bonilla, E.V. Collaborative multi-output Gaussian processes. In Proceedings of the UAI'14: Thirtieth Conference on Uncertainty in Artificial Intelligence, Citeseer, Quebec City, QC, Canada, 23–27 July 2014; pp. 643–652.

64. Carron, A.; Todescato, M.; Carli, R.; Schenato, L.; Pillonetto, G. Multi-agents adaptive estimation and coverage control using Gaussian regression. In Proceedings of the 2015 European Control Conference (ECC), Linz, Austria, 15–17 July 2015; pp. 2490–2495.
65. Mallasto, A.; Feragen, A. Learning from uncertain curves: The 2-wasserstein metric for gaussian processes. In Proceedings of the 31st Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
66. Rasmussen, C.E.; Williams, C.K. Gaussian Processes for Machine Learning. In *Adaptive Computation and Machine Learning*; MIT Press: Cambridge, MA, USA, 2006.
67. Khalil, H.K. Nonlinear Systems: International Edition. *Bull. Am. Acad. Arts Sci.* **2002**, *53*, 20–24.
68. Umlauft, J.; Hirche, S. Feedback linearization based on Gaussian processes with event triggered online learning. *IEEE Trans. Autom. Control.* **2020**, *65*, 4154–4169. [CrossRef]
69. Zhou, B.; Gao, L.; Dai, Y.H. Gradient methods with adaptive step-sizes. *Comput. Optim. Appl.* **2006**, *35*, 69–86. [CrossRef]
70. Ivanov, S.E.; Zudilova, T.; Voitiuk, T.; Ivanova, L.N. Mathematical modeling of the dynamics of 3-DOF robot-manipulator with software control. *Procedia Comput. Sci.* **2020**, *178*, 311–319. [CrossRef]
71. Abdolhosseini, M. Model Predictive Control of an Unmanned Quadrotor Helicopter: Theory and Flight Tests. Ph.D. Thesis, Concordia University, Montreal, QC, Canada, 2012.
72. Cannon, M. Efficient nonlinear model predictive control algorithms. *Annu. Rev. Control.* **2004**, *28*, 229–237. [CrossRef]
73. Beckers, T.; Kulic, D.; Hirche, S. Stable Gaussian process based tracking control of Euler-Lagrange systems. *Automatica* **2019**, *103*, 390–397. [CrossRef]
74. Srinivas, N.; Krause, A.; Kakade, S.M.; Seeger, M.W. Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE Trans. Inf. Theory* **2012**, *58*, 3250–3265. [CrossRef]