# Global Peak Finding based on Distributed Gaussian Process and Active Sensing

Weijie Qi
*weijie.qi@studenti.unitn.it*

Yunru Qu
*yunru.qu@studenti.unitn.it*

***Abstract*—This paper presents a fully distributed multi-agent learning and cooperative control approach for addressing the task of multiple agents working together to explore and converge to the global maximum in an unknown environment. As illustrated in Fig. 1, each agent collects noisy altitude measurements obtained from laser-based sensors and leverage Gaussian Process Regression (GPR) to construct spatial information maps with confidence intervals. However, traditional Gaussian processing algorithms struggle with real-time processing of information from multiple distributed sensors due to their centralized framework. Considering limited computational and communication capabilities, the proposed algorithm uses the first E standard orthogonal eigenfunctions obtained from the Karhunen-Loève expansion of the selected kernel. Each agent maintains and updates its own environment modelling estimate and move toward the peak of the field while avoiding collisions. The effectiveness of the algorithm is validated through simulations.**

***Index Terms*—Gaussian process, distributed estimation, robotic sensor networks, average consensus**

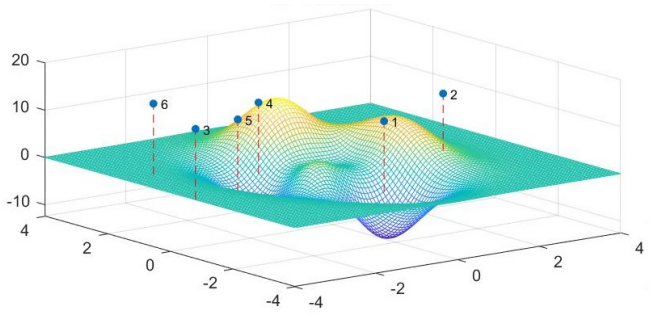



Fig. 1. Illustration of the environment and robot system for measurement. Each numbered point represents an agent, and the dashed lines represent the lasers emitted by the agents for distance measurement.

## I. Introduction

### A. Related Work

Exploration and modeling in an unknown environment is a fundamental task that is widely applied, such as collecting and modeling the topographic height, temperature field, pollutant concentration, and crop density distribution of the environment. Mobile robots are particularly well-suited for carrying sensors to relevant sampling locations, thereby reducing the time and manpower required for exploration. Multi-robot systems perform exploration tasks more efficiently than single robots. The collective technique of multiple robots moving and acquiring local information to construct a global environmental map is known as robotic sensor networks [1]. To effectively carry out exploration and modeling in such environments, it is crucial to collect data efficiently and construct accurate environment models.

Popular methods for simulating nonlinear spatial phenomena include Gaussian processes [2], [3], kriging models [4], and spatio-temporal Kalman filter models [5]. Among them, Gaussian processes have gained significant popularity in mobile sensor networks. Initially, Gaussian processes were primarily used for regression analysis and interpolation to estimate unknown data points or generate smooth curves. However, with the advancement of machine learning techniques and probabilistic modeling theory, Gaussian processes have been increasingly applied in environment modeling, leading to substantial progress. Researchers have specifically focused on addressing the requirements of robotic systems, such as real-time online learning and multi-agent cooperative control.

In the context of Gaussian processes used for environmental modeling, [6] proposed Gaussian processes for data-efficient learning in robotics and control. The goal was to learn a probabilistic transition model of the system. They highlighted the advantages of Gaussian processes in providing probabilistic predictions and handling uncertainty. In another study [7], the challenge of real-time online model learning for robots was addressed. The authors introduced an approach called local Gaussian process regression, which allows for online learning by offering an online version of Gaussian processes. This method is particularly useful in scenarios where waiting for offline processing of collected data is not feasible. Distributed multi-agent systems were the focus of research conducted by [8]. The authors presented distributed multi-agent Gaussian regression using finite-dimensional approximations as a solution to overcome limitations found in centralized Gaussian process models within multi-agent systems. Their proposal involved leveraging multiple Gaussian processes with Karhunen-Loéve kernel expansion to address communication and computational constraints. Informative planning and online learning with sparse Gaussian processes were explored by [9]. Their work extended previous research by considering mobile robots and addressing the need for informative planning. Through

All the code developed for implementing the algorithms presented in the paper is publicly available in the repository https://github.com/Winnie-Qi/Global-Peak-Finding-based-on-Distributed-Gaussian-Process-and-Active-Sensing.

utilizing sparse Gaussian processes, they successfully tackled computational challenges associated with mobile robots while enabling effective online learning. Taking it a step further, [10] considered collision avoidance within a distributed learning and cooperative control framework for multi-agent systems. Their research focused on addressing challenges encountered by real-world mobile robots while emphasizing collision avoidance as a key problem to be solved within this context.

### B. Our Work

We are interested in the problem of collecting data of the terrain height for the environment and developing a model to describe the terrain. To investigate this, we performed multi-robot exploration simulations in a virtual environment. Our experiments involved employing multiple unmanned aerial vehicles (UAVs) equipped with laser sensors to simulate topographic surveys and gather relevant data.

Our work mainly focuses on the following three parts.

- We built a fully distributed exploration and learning architecture in which each agent independently learns the environment model using a distributed Gaussian process algorithm. Unlike traditional Gaussian process algorithms designed for centralized systems, which involve exchanging all M measurements among agents and inverting an M x M matrix, rendering them non-scalable, the computational complexity of our approach scales with an E, where E$\ll$M. This architecture ensures the computational efficiency and scalability of the system.
- We enabled collaborative work among the system of agents by employing average consensus protocols for data communication. We first define the communication network of the system using graph theory. Then, we introduce a dynamic mechanism that leverages average consensus to update the state, ensuring that agents collectively achieve a shared objective.
- We developed a distributed multi-agent motion control scheme that empowers each agent to independently explore the environment, taking into account the uncertainty of Gaussian Process (GP) estimation. For each individual agent, once it has accumulated enough information about its esitimate, the exploration phase concludes, and it proceeds to navigate towards the global highest point on the constructed map. This entire process is executed in parallel and autonomously by each agent, eliminating the need for coordination from a central processor.

The remaining sections of this report are structured as follows. We start by providing an overview of the problem setting in Section II. In Section III, we review the traditional Gaussian regression problem and transform it into a distributed approach that is well-suited for continuous data collection. Motion control mechanism with collision avoidance ability is proposed in Section IV. Simulation experiments are presented in Section V. Lastly, Section VI concludes the report.

## II. Problem Statement

### A. Setting of the Environment

There is a surface with multiple peaks in 3-D space, which can be modeled using the bivariate function $f(.)$. Given a pair of $(x, y)$ coordinates, $f(x, y)$ represents the height of the surface at this point. However, in our problem, we assume that the environment model $f(.)$ is unknown.

### B. Setting of the System

In our work, we assume the following:

- The agents in our scenario are UAVs equipped with laser sensors. These UAVs fly above the environment at a specific altitude. The motion control of the agent at altitude is not considered in our study.
- Each time the agent's sensor is sampled, it captures information representing the height of a specific point on the terrain. We omit the process of calculating the height from the sensor data. The agent collects this data at fixed intervals during its movement.
- The position of the agent itself is assumed to be accurate and free from noise.
- However, the information acquired by the agent, which is the terrain height, is considered to be noisy.
- The boundaries that define the environment are known in advance. The topographic profile of the environment is assumed to be time-invariant.
- Each agent can communicate directly with other agents within its communication range. The communication between agents is bidirectional.

### C. Setting of Communication

During the process of exploring the environment, the robot estimates the environmental model by utilizing both measured data and shared data acquired from neighboring robots within a specific distance. Based on graph theory, this communication can be represented by an adjacency matrix. It is an undirected graph as the communication between agents is bidirectional. In our system with N agents, the adjacency matrix $\mathcal{N}_i(k)$ is an N x N square matrix. Each entry indicates the presence or absence of an edge connecting two vertices, which represents whether the distances between two agents fall within a predefined communication range.

All agents adhere to the same average consensus protocol for information sharing and integration. Considering the varying uncertainty of the global estimate across different locations, instead of directly communicating the global estimate of the environment, we utilize a specific "feature" sampled by the distributed Gaussian process for communication. This "feature" matrix is represented as $\{X_i\}_{i=1}^N$. The details of $\{X_i\}_{i=1}^N$ will be further explained in III-D.

We define the following dynamics:

$$X_i((k+1)T) = X_i(kT) + U_i(kT). \tag{1}$$

To achieve the average consensus, the following equation is utilized:

$$U_i(k) = -\frac{1}{|\mathcal{N}_i(k)|} \sum_{j \in \mathcal{N}_i(k)} \gamma \left( X_i(k) - X_j(k) \right), \quad (2)$$

where $\gamma \in \mathbb{R}$ is the parameter for convergence rate.

When the average consensus is reached, the following condition is met:

$$\sum_{i \in \mathcal{N}} X_i(k) = \bar{X}\left(k_0\right), \quad \forall k \in \left(k_0, \infty\right). \quad (3)$$

## III. Distributed Gaussian Process

### A. Gaussian Process Regression

The Gaussian process serves as the fundamental data model we employ, utilizing kernel functions to establish spatial relationships among sampled data. This approach aids in reducing the amount of data sampling needed and facilitates Bayesian inference predictions for unknown locations. A typical form of a Gaussian process is represented as:

$$f(x) \sim \mathcal{GP}\left(m(x), k\left(x, x'\right)\right). \quad (4)$$

Here, $f(x)$ denotes the function to be modeled, which corresponds to the measurement model $f(.)$ in II-A. $m(x)$ represents the mean function, while $k(x, x')$ denotes the covariance function, also referred to as the kernel function, which specifies the correlation between input points.

The choice of covariance function depends on the characteristics of the data being modeled, as different covariance functions have distinct properties, such as smoothness, length scale, and periodicity. In our scenario, we opt for the squared exponential kernel, which takes the following form:

$$k(x, x') = \sigma^2 exp(-\frac{1}{2}(x - x')^T), \quad (5)$$

in which $\sigma^2$ is a hyperparameter that determines the variance and length scale of the kernel function.

We define the training input data, denoted as $\mathbf{X}$ and $\mathbf{y}$. $\mathbf{X}$ represents the locations of each agent, while $\mathbf{y}$ represents the corresponding measurement height values obtained at those locations. Additionally, we define the test location, $\mathbf{X}_*$, which is used to construct the complete topographic model. To make predictions about the output values at new input points, specifically $\mathbf{y}^*(\mathbf{X}^*)$, we employ the process of updating the posterior distribution. This involves utilizing Bayes' rule to compute the posterior distribution:

$$\hat{f}\left(\mathbf{X}_*\right) = K\left(\mathbf{X}_*, \mathbf{X}\right)\left(K(\mathbf{X}, \mathbf{X}) + \sigma_v^2 I\right)^{-1} \mathbf{y}, \quad (6)$$

in which $\sigma_v^2$ is the variance of the Gaussian noise, as will be shown in III-B later.

It also provides a measure of uncertainty for each prediction:

$$\begin{aligned} \Sigma\left(\mathbf{X}_*\right) =& k\left(\mathbf{X}_*, \mathbf{X}_*\right) \\ & - K\left(\mathbf{X}_*, \mathbf{X}\right)\left(K(\mathbf{X}, \mathbf{X}) + \sigma_v^2 I\right)^{-1} K\left(\mathbf{X}, \mathbf{X}_*\right). \end{aligned} \quad (7)$$

This measure of uncertainty can be valuable in decision-making processes, as it allows us to assess the reliability and confidence associated with the predicted output values.

### B. Measurement Model

We consider a scenario where N agents are deployed to explore the environment. Each agent is equipped with a 2-D position state (ignoring the height value since it is assumed to be constant), denoted as $x_i \in \mathbb{R}^2$, for $i = 1, \ldots, N$, and is responsible for measuring the environmental process $y_i \in \mathbb{R}$. These measurements are subject to corruption by white Gaussian noise $\nu_i \sim N(0, \sigma_v^2)$, where $\sigma_v$ represents the variance parameter of the noise. The relationship between the measurement $y_i$ and the true environmental value $f(x_i)$ can be expressed as $y_i = f(x_i) + \nu_i$. In this context, the algorithm aims to estimate the measurement model $f(.)$.

We consider an agent that has M measurements. Each of its measurements follows the relationship $y_m = f(x_m) + \nu_m$, $\{\nu_m\}_{m=1}^M$ are assumed to be mutually independent. We now express the model in a more compact vector form.

$$\begin{aligned} \boldsymbol{x} &:= \left[x_1, \ldots, x_M\right]^T \\ \boldsymbol{y} := \left[y_1, \ldots, y_M\right]^T & \quad \boldsymbol{v} := \left[v_1, \ldots, v_M\right]^T \end{aligned} \quad (8)$$

The posterior estimate of a test point $x_*$ is then given by

$$\widehat{f}(x_*) = \left[K\left(x, x_1\right) \ldots K\left(x, x_M\right)\right] H_{\mathrm{MAP}} \begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix}, \quad (9)$$

with

$$H := \left( \begin{bmatrix} K\left(x_1, x_1\right) & \cdots & K\left(x_1, x_M\right) \\ \vdots & & \vdots \\ K\left(x_M, x_1\right) & \cdots & K\left(x_M, x_M\right) \end{bmatrix} + \sigma_v^2 I \right)^{-1}. \quad (10)$$

### C. Finite dimension approximation- Karhunen-Loéve (KL) kernel expansion

Standard kernel-based GP assumes that all sample data is available to a central processor. Directly using the squared exponential kernel shown in (5) to compute the correlation between all the sample data is not applicable in distributed scenarios. Therefore, a new kernel transformation is required to enable kernel-based GP regression in a distributed architecture. Based on Mercer's Theorem, the non-negative definiteness of this covariance function enables its spectral decomposition using the Karhunen–Loève expansion. It is defined by:

$$\lambda_e \phi_e(\mathbf{X}) = \int_{\chi} k\left(\mathbf{X}, \mathbf{X}'\right) \phi_e\left(\mathbf{X}'\right) d\mu\left(\mathbf{X}'\right). \quad (11)$$

The kernel allows for the following uniformly convergent expansion

$$k(\mathbf{x}, \mathbf{X}') = \sum_{e=1}^{+\infty} \lambda_e \phi_e(\mathbf{x}) \phi_e(\mathbf{X}'), \lambda_1 \geq \lambda_2 \geq \ldots > 0, \quad (12)$$

where the $\{\lambda_e, e \in \mathbb{Z}^+\}$ and $\{\phi_e, e \in \mathbb{Z}^+\}$ are respectively the nonzero eigenvalues and eigenfunctions of the kernel. It is a continuous version of the standard finite-dimensional result for symmetric positive semi-definite matrices.

The stochastic process $f(x)$ itself can then be expressed as

$$f(\mathbf{x}) = \lim_{p \to +\infty} \sum_{e=1}^{p} a_e \phi_e(\mathbf{x}), a_e \sim N(0, \lambda_e), e = 1, 2, \dots \tag{13}$$

It decomposes a real function into a linear combination of orthogonal basis terms, where the $\phi_e$'s are fixed eigenfunctions of the kernel, and $a_e$'s are i.i.d. normal random variables with zero mean and eigenvalues as the covariances. When the order p increases to infinity, the mean squared error decreases to zero in the stochastic process space.

We assume that the eigenfunctions are ordered in terms of non-decreasing values of the associated eigenvalues. Then, we truncate the expansion and capture the first E eigenvalues and eigenfunctions:

$$\begin{aligned} f(\mathbf{x}) &= \sum_{e=1}^{E} a_e \phi_e(\mathbf{x}) + \sum_{e=E+1}^{+\infty} a_e \phi_e(\mathbf{x}) \\ &= f_E(\mathbf{x}) + \sum_{e=E+1}^{+\infty} a_e \phi_e(\mathbf{x}), \end{aligned} \tag{14}$$

where the selection of the truncation point E depends on the desired level of accuracy in reconstructing the covariance function.

### *D. Distributed Gaussian Processes for Continuous Data Acquisition and Communication*

Our next step involves finding finite-dimensional estimators of $\widehat{f}$ that are suitable for distributed implementations. By considering the decomposition described in (14), we adopt the truncated E-dimensional approximation, which serves as the asymptotically optimal solution for estimation. Referring to [10], the matrix form of this approximation is as follows:

$$\widehat{f}_E(x_*) := [\phi_1(x_*), ..., \phi_E(x_*)] H_E \boldsymbol{y}, \tag{15}$$

where

$$H_E := (\frac{G^T G}{Nm} + \frac{\sigma_v^2}{Nm} \Lambda_E^{-1})^{-1} \frac{G^T}{Nm}, \tag{16}$$

$$G := \begin{bmatrix} G_{11} & \dots & G_{1E} \\ \vdots & & \vdots \\ G_{M1} & \dots & G_{ME} \end{bmatrix}, \tag{17}$$

$$G_{me} := \phi_e(x_m), \quad m = 1, \dots, M, e = 1, \dots, E, \tag{18}$$

and $\Lambda_E := \operatorname{diag}(\lambda_1, \dots, \lambda_E)$.

Decompose (16) into local quantities for each agent:

$$\begin{aligned} \frac{G^T G}{Nm} &= \frac{1}{N} \sum_{i=1}^{N} \Phi(\mathbf{x}_i) \Phi^T(\mathbf{x}_i) = \frac{1}{N} \sum_{i=1}^{N} \alpha_i(k_0) \\ \frac{G^T \mathbf{y}}{Nm} &= \frac{1}{N} \sum_{i=1}^{N} \Phi(\mathbf{x}_i) y_i = \frac{1}{N} \sum_{i=1}^{N} \beta_i(k_0) \end{aligned}. \tag{19}$$

Now (15) is reformulated into a fully distributed form and can be calculated in parallel by each agent:

$$\hat{f}_{E,i}(x_*) := \Phi^T(\mathbf{x}) \left( \alpha_i(k) + \frac{\sigma_\nu^2}{N} \Lambda_E^{-1} \right)^{-1} \beta_i(k). \tag{20}$$

$\{\alpha_i\}_{i=1}^N$ and $\{\beta_i\}_{i=1}^N$ here are the state matrices used in the average consensus algorithm mentioned in II-C. Precisely, the update dynamics based on II-C are:

$$\begin{aligned} \alpha_i(k+1) &= \alpha_i(k) + \Delta\alpha_i(k) \\ \beta_i(k+1) &= \beta_i(k) + \Delta\beta_i(k) \end{aligned}, \tag{21}$$

where

$$\begin{aligned} \Delta\alpha_i(k) &= -\sum_{j \in \mathcal{N}_i} \gamma(\alpha_i(k) - \alpha_j(k)) \\ \Delta\beta_i(k) &= -\sum_{j \in \mathcal{N}_i} \gamma(\beta_i(k) - \beta_j(k)) \end{aligned} \tag{22}$$

Analogously, the distributed form of updating (7) is:

$$\begin{aligned} \Sigma_{E,i}(\mathbf{x}) :=& k(\mathbf{x}, \mathbf{x}) \\ &- \Phi^T(\mathbf{x}) \left( \alpha_i + \frac{\sigma_\nu^2}{N} \Lambda_E^{-1} \right)^{-1} \alpha_i \Lambda_E \Phi(\mathbf{x}) \end{aligned}, \tag{23}$$

## IV. Active Sensing and Motion Control

### *A. Anti-flocking Dynamics*

The concept of the flocking algorithm draws inspiration from the natural behavior of animals, particularly birds, that exhibit flocking behavior [12]. While the tendency of individuals to come together and move as a cohesive group can be undesirable in other situations where dispersion and coverage are important. The dynamics of solitary animals have been introduced using the term "anti-flocking" in [11]. This behavior is characterized by three key aspects: 1) selfishness, as individuals move in a direction that maximizes their own gains; 2) decentralization, as their actions are distributed and autonomous; and 3) collision avoidance, as they maintain a safe distance from each other to prevent collisions. In our particular case, we have adopted the anti-flocking algorithm introduced in [11], which enables exploration and exploitation in obstacle-free environments. Consequently, the term "anti-flocking" in our system can be understood as follows: 1) the agents move towards positions that offer maximum uncertainty for data collection; 2) they operate in a decentralized manner; and 3) they actively avoid collisions while staying within the designated range.

### *B. Motion Control Mechanism*

While exploring the distributed information map, the agents are controlled using the anti-flocking algorithm to optimize cumulative area coverage while minimizing overlap in sensing coverage [11]. Each agent aims to identify the position, denoted as `p`, with the maximum variance, `m`, within a self-centralized square area of exploration. If the variance `m` at position `p` exceeds the threshold `M`, the agent moves towards that position to effectively collect information. Conversely, if the variance `m` at position `p` is below the threshold `M`, the

agent adjusts its exploration scale and tunes the threshold M to explore an area with higher uncertainty. At the end of the exploration process, agents converge toward the position with the highest evaluative value.

For each agent (n = 1, 2, 3,..., N), we denote its position as $X_k(n)$, its velocity as $V_k(n)$, and its acceleration as $U_k(n)$. Dynamics of each agent is given by

$$\begin{cases} \dot{X}_n(t) = V_n(t) \\ \dot{V}_n(t) = U_n(t), \quad n = 1, 2, \ldots, N. \end{cases} \tag{24}$$

Agents are expected to remain within the designated boundary. As an agent approaches the boundary, it experiences a constant repulsive force acting in the direction away from the boundary. This force slows down the agent's approach, preventing it from crossing the boundary. We define the threshold distance between the agent and the boundary as $d_B$. Only when an agent enters this threshold distance from the boundary will it be affected by the repulsive force. Positions outside this threshold distance will not exert any repulsive force. It is important to choose an appropriate value for $d_B$ to ensure that agents can still explore the area near the boundary. While agents may temporarily venture beyond the boundary, the continuous action of the repulsive force will automatically guide them back within the range of the map. To facilitate this, we conduct boundary checks and define the repulsive acceleration as follows:

$$U_b(n) = \begin{cases} U_B, & if|X_k(n) - Boudaries| < d_B \\ 0, & otherwise. \end{cases}. \tag{25}$$

Similarly, when agents come into close proximity, a repulsive potential should be employed to prevent collisions. This repulsive force is activated when the distance between agents falls below a specific threshold. It is crucial that the distance threshold is smaller than the communication radius between agents, as exceeding it would hinder information exchange. Unlike boundary avoidance, it necessitates a stronger repulsive force as agents approach each other. Hence, we establish the repulsive force to be proportional to the inverse of the distance between agents. The repulsive pairwise acceleration is defined as follows:

$$U_c(n_i, n_j) = \begin{cases} \frac{h}{|X(n_i) - X(n_j)|}, & if|X(n_i) - X(n_j)| < d_c \\ 0, & otherwise. \end{cases} \tag{26}$$

The motion control mechanism adopted for the $i^{th}$ individual agent at the $k^{th}$ action frame is described in Algorithm 1. The boundary check (25) and collision check (26) operations are performed before executing each action, ensuring the agent's safety (lines 1-2 of Algorithm 1). At the initial position, each agent begins the exploration phase with the default value of $ExplorationFlag$ set to 1 (lines 3-12 of Algorithm 1). During this phase, the agent explores a rectangular area of size $L * L$ and moves towards the point $max_X(n)$ with the highest variance. If the maximum variance within the rectangular area is smaller than the threshold $M$, the size of the area, denoted by L, is expanded, and the search for the maximum value is repeated within the expanded range. This process continues until a variance greater than M is found within the $L * L$ window or until $L$ expands beyond a specified threshold. Once the exploration phase concludes, the agent proceeds to navigate towards the highest point in the terrain. The agent identifies the position within the entire map area where the GP estimate has the maximum value. The position with the maximum value obtained during both stages is output as an angle of deflection relative to the agent's current location. We denote the standard constant velocity as $V_0$ and the deflection angle of next movement as $ang$. The velocity for the next motion frame is calculated as follows:

$$\begin{aligned} V &= [V_0 * cos(ang), V_0 * sin(ang)] \\ V &= V + U_B + U_c. \end{aligned} \tag{27}$$

Then the next position of the agent is:

$$X_{i,k+1} = X_{i,k} + V. \tag{28}$$

**Algorithm 1:** Anti-Flocking Algorithm

**Input:** $X_{i,k}, ExplorationFlag, P_{i,E}, f_{i,E}$
**Result:** $X_{i,k+1}$

```
1  Boundary Check: output = U_b
2  Collision Check: output = U_c
3  if ExplorationFlag = 1 then
4      while true do
5          L ← L_initial
           [max_value, max_index] = max(P_{i,E}(X_{i,k}(1) −
           L : X_{i,k}(1) + L, X_{i,k}(2) − L : X_{i,k}(2) + L))
6          if max_value > M then
7              ang = Arctan(|max_index(2) − X_{i,k}(2)| / |max_index(1) − X_{i,k}(1)|);
8              break
9          else
10             L = L + 1;
11             if L > L_thresh then
12                 break
13             end
14         end
15     end
16 else
17     [∼, max_index] = max(F_{i,E});
18     ang = Arctan((max_i(2) − X_{i,k}(2)) / (max_i(1) − X_{i,k}(1)));
19 end
20 V = [V_0 * cos(ang), V_0 * sin(ang)];
21 V = V + U_b + U_c;
22 X_{k+1} = X_k + V
```

## V. Simulation Result

### A. Static Sensors

First, we implement stationary positions uniformly distributed in space. The goal of this simulation is to demonstrate the iterative and propagative behavior of information exchange between agents, ultimately leading to the reconstruction of an environmental model.

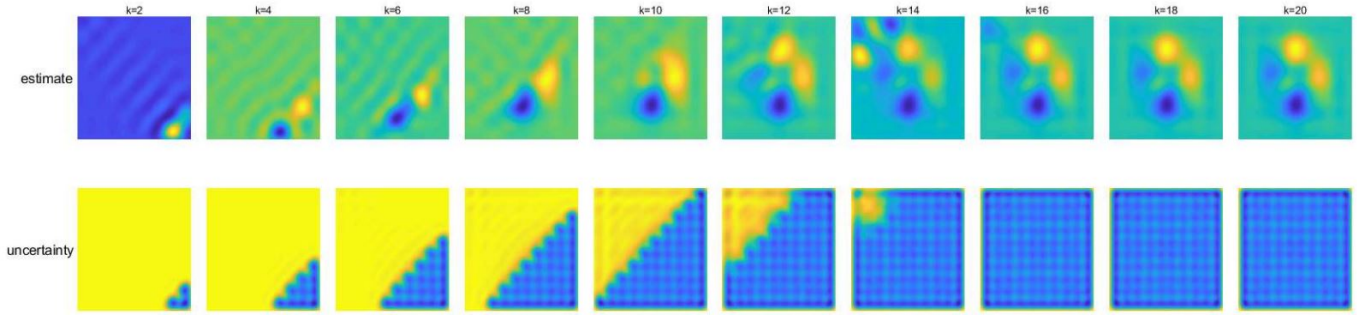



Fig. 2. The estimate and uncertainty distribution from the *No.* 10 agent.

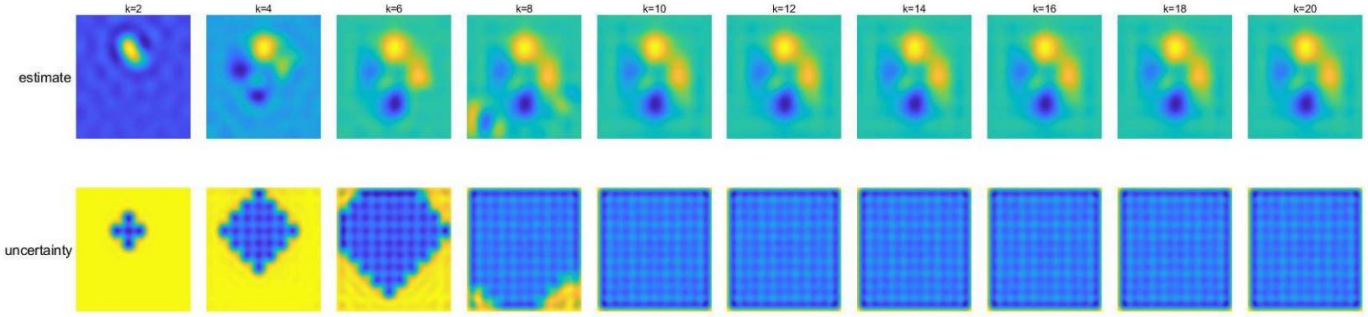



Fig. 3. The estimate and uncertainty distribution from the *No.* 65 agent.

We convert the 3D height map shown in Fig. 1 to a 2D top-down view, as depicted in Fig. 4. This representation uses a colormap to depict the terrain's height using different colors.

(18) requires computing the eigenfunctions and eigenvalues of dimension E for each test point. Specifically, we precompute the eigenfunctions and eigenvalues with a resolution of 100×100 within the range $(-4, 4)$, and store them as look-up tables (LUT). During the execution phase, the coordinates of the test point are rounded to locate the corresponding point in the LUT. Consequently, we can retrieve the precomputed eigenfunctions and eigenvalues that are associated with the given test point.

We evenly distribute 10x10 sensors on the map, and their labels and positions are shown in Fig. 4. Each sensor can only obtain a noisy measurement of the terrain height at its own coordinate point. In the scenario with 10x10 sensors, the distance between left-right and up-down sensors is 0.778, and the distance between diagonally adjacent sensors is 1.1. By setting the communication radius between 0.778 and 1.1, each sensor can directly communicate with its immediate up, down, left, and right neighbors, but cannot communicate directly with sensors that are diagonally adjacent. The connectivity of the graph based on the communication relationships among the sensors is illustrated in Fig. 4.

We set $\sigma^2 = 0.01$ for the Gaussian kernel (5), $E = 100$ for the E-dimensional estimator (15) and (23), and $\gamma = 0.7$ for the convergence rate in (21).

Fig. 2 and Fig. 3 show the iteration progress from $k = 1$ to $k = 20$. Although sensors can only directly communicate with their neighbors within their communication range, as the number of iterations increases, more information is exchanged between neighboring sensors. Therefore, after a sufficient number of iterations, each sensor obtains the global information.

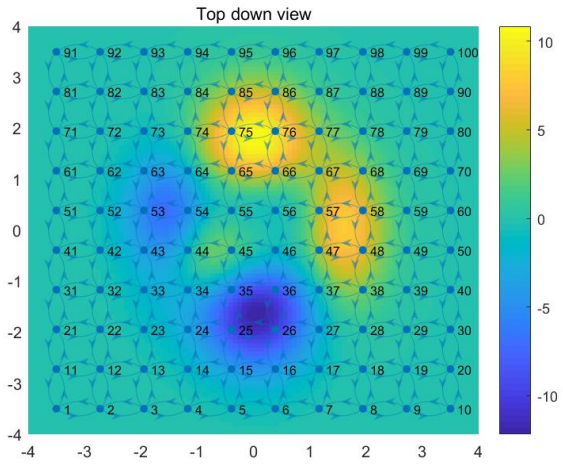



Fig. 4. Placement of stationary sensors in the environment.

The iteration process of *No.* 10 sensor, located in the bottom right corner of the map, is depicted in Fig. 2. It only has left and upper neighbors. On the other hand, *No.* 65 sensor, shown

in Fig. 3, is located in the middle part of the map and has four neighbors in all four directions. As a result, *No.* 65 sensor can acquire the global information faster than *No.* 10. Nonetheless, with the effect of averaging consensus, their global estimates will eventually converge to the same value.

### *B. Mobile Agents*

Then, we implemented active sensing for a network of six mobile agents. The agents are tasked with exploration while considering collision avoidance and coordination. It can be initiated with arbitrary initial locations and spread throughout the environment to gather information. The goal is to estimate the overall environment map and finally gather at the highest peak.

First, each agent performs exploration using the following set of rules. The agent prioritizes exploring areas near its current location over distant areas, and gradually expands its area of interest. During the exploration process, each agent searches for the maximum variance within a square window centered around itself with a side length of 10 (the unit length is the minimum length based on resolution). If the maximum variance value exceeds 11 (this number is related to the hyper-parameters used in calculating LUT and kernel functions), the agent moves towards the direction of the maximum value at a speed of 0.2. Otherwise, the agent increases the side length of the window by 1 and decreases M by 0.05. The agent then repeats the process of searching for the maximum variance within the updated window. This process continues until the side length of the window exceeds 20. Once the side length of the window exceeds the threshold, the agent transitions to moving towards the maximum value found in the reconstructed estimation of the entire environment.

Throughout the movement of the agents, mechanisms are in place to prevent them from exceeding the boundaries of the map and to avoid collisions between agents. To prevent agents from exceeding the map boundaries, a repulsive force of 0.01 is applied when the agent is within a distance of 0.5 from the boundary. This force pushes the agent back towards the map. To enable agents to explore areas near the boundaries without straying too far, the boundary is not a hard constraint. Agents may temporarily move away from the boundary, but they will automatically return within the map boundaries under the continuous influence of the repulsive force. To avoid collisions between agents, a repulsive force is applied when the distance between them is less than 0.6. The repulsive force disappears when the distance between agents exceeds 0.6. The threshold distance should be smaller than the communication radius of the agents to ensure effective information exchange among them.

Fig. 5 illustrates the process from $k = 0$ to $k = 420$. k represents the number of iterations, and m represents the number of measurements obtained by each agent during the exploration process. The six agents collectively searched the map, converging at the peak location without colliding with one another. Each agent aimed to maintain a safe distance from its neighbors to improve exploration efficiency. Additionally, they avoided approaching areas where the estimation was already reliable.

The results of online GP variance estimation are shown at the bottom of each subfigure in Fig. 5. Over time, the uncertainty in all regions decreased. Similarly, the second-to-last row in each subfigure displays the GP mean estimation, gradually converging to a similar result as shown in 5. Through average consensus, the distributed GP estimations of each agent converged.

## VI. Conclusions

In this paper, we present a study on the application of Distributed Gaussian Process for active sensing and motion control. Our research aimed to address the problem of efficient data collection and motion planning in scalable sensor networks. We proposed a novel approach based on Distributed Gaussian Process. By modeling the spatial dependencies among sensor measurements, it enables accurate predictions and reduces the need for redundant data acquisition. To validate the effectiveness of our approach, we conducted simulations using a large-scale sensor network. The simulation results demonstrate the effectiveness of the distributed architecture in exploring the environment and gathering at the highest peak tasks. Currently the accuracy of GP estimate is determined by the pre-computed LUT, and increasing the resolution of LUT will increase the demand for memory exponentially. In future work, we plan to optimize the management of GP estimate accuracy and analyze the tradeoff strategy between estimation accuracy and memory requirements.

## References


[1] Julian, B.J., Angermann, M., Schwager, M. and Rus, D., 2012. Distributed robotic sensor networks: An information-theoretic approach. The International Journal of Robotics Research, 31(10), pp.1134-1154.

[2] MacKay, D.J., 1998. Introduction to Gaussian processes. NATO ASI series F computer and systems sciences, 168, pp.133-166.

[3] Williams, C.K. and Rasmussen, C.E., 2006. Gaussian processes for machine learning (Vol. 2, No. 3, p. 4). Cambridge, MA: MIT press.

[4] Cressie, N., 2015. Statistics for spatial data. John Wiley & Sons.

[5] Cressie, N. and Wikle, C.K., 2002. Space-time Kalman filter. Encyclopedia of environmetrics, 4, pp.2045-2049.

[6] Deisenroth, M.P., Fox, D. and Rasmussen, C.E., 2013. Gaussian processes for data-efficient learning in robotics and control. IEEE transactions on pattern analysis and machine intelligence, 37(2), pp.408-423.

[7] Nguyen-Tuong, D., Peters, J. and Seeger, M., 2008. Local Gaussian process regression for real time online model learning. Advances in neural information processing systems, 21.

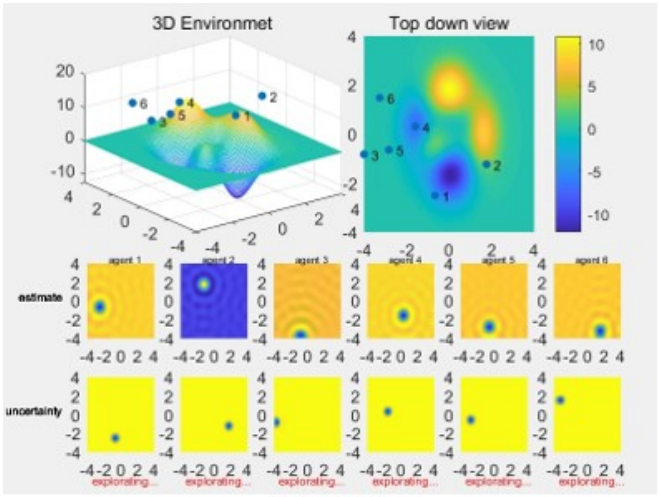
[8] Pillonetto, G., Schenato, L. and Varagnolo, D., 2018. Distributed multi-agent Gaussian regression via finite-dimensional approximations. IEEE Transactions on Pattern Analysis and Machine Intelligence, 41(9), pp.2098-2111.

[9] Ma, K.C., Liu, L. and Sukhatme, G.S., 2017, May. Informative planning and online learning with sparse gaussian processes. In 2017 IEEE International Conference on Robotics and Automation (ICRA) (pp. 4292-4298). IEEE.

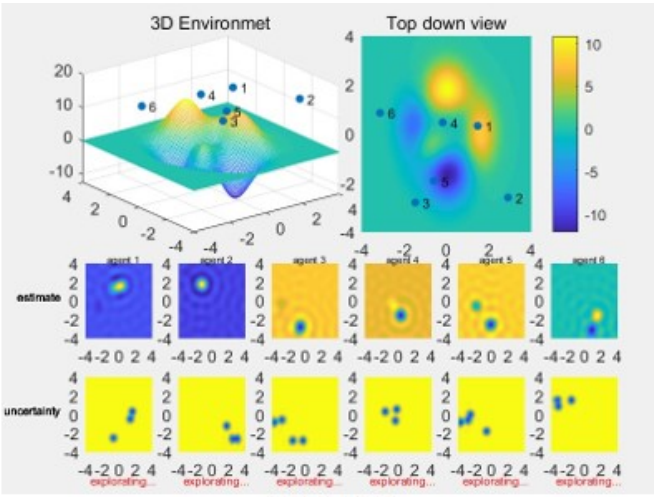
[10] Choi, J., Oh, S. and Horowitz, R., 2009. Distributed learning and cooperative control for multi-agent systems. Automatica, 45(12), pp.2802-2814.

[11] Ganganath, N., Cheng, C.T. and Chi, K.T., 2016. Distributed antiflocking algorithms for dynamic coverage of mobile sensor networks. IEEE transactions on industrial informatics, 12(5), pp.1795-1805.

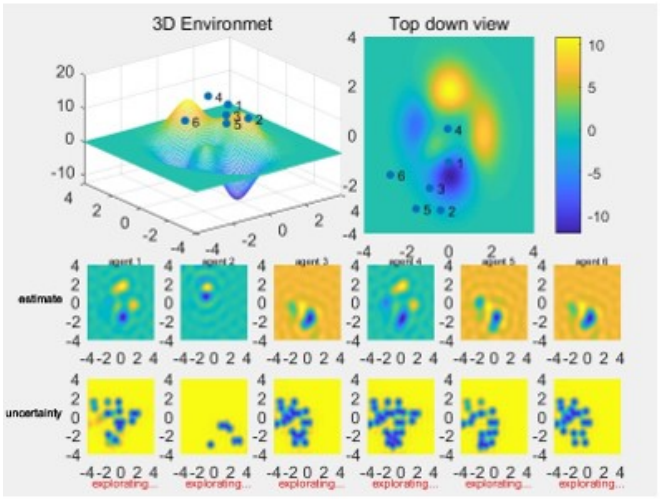
[12] Olfati-Saber, R., 2006. Flocking for multi-agent dynamic systems: Algorithms and theory. IEEE Transactions on automatic control, 51(3), pp.401-420.



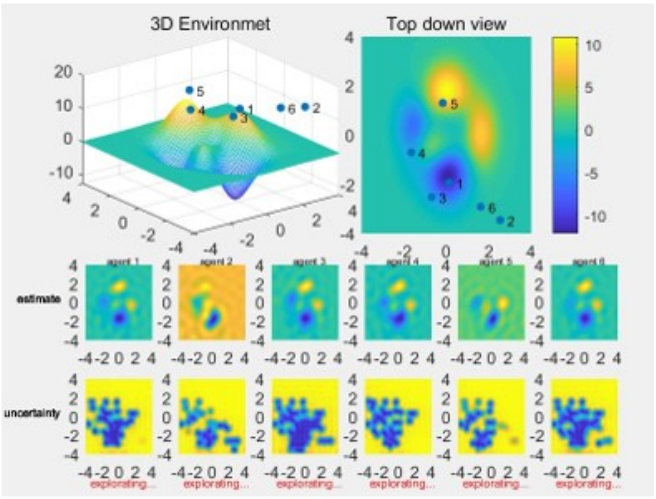

k=1, m=1



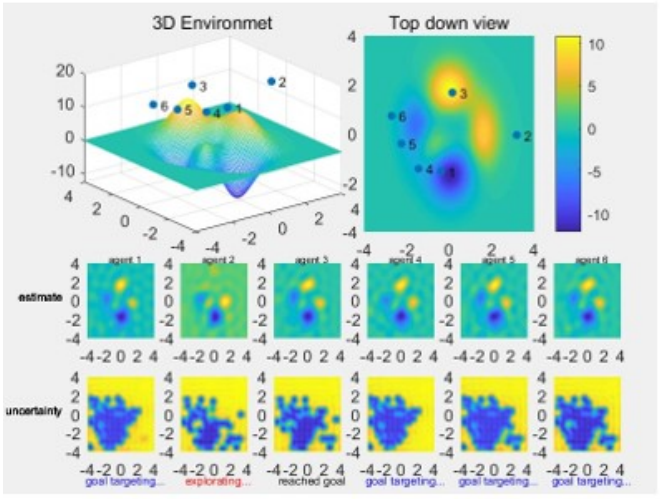

k=60, m=3



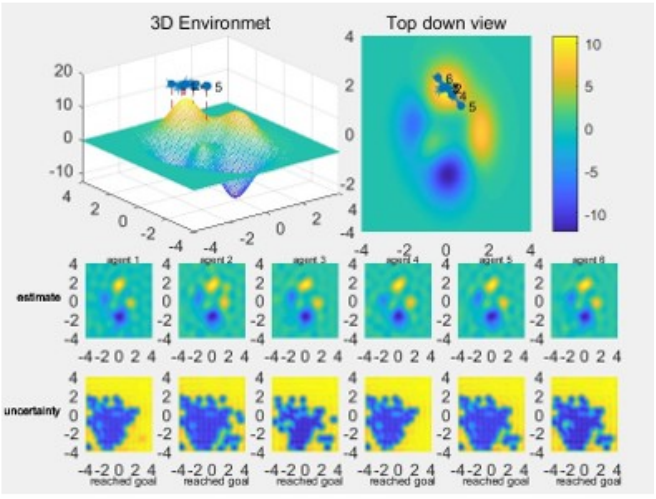

k=150, m=6




k=240, m=9




k=330, m=12




k=420, m=15

Fig. 5. 6 agents exploring the environment and finally gathering at the global peak.