Adaptive Exploration-Exploitation Active Learning of Gaussian Processes

George P. Kontoudis and Michael Otte

***Abstract*— Active Learning of Gaussian process (GP) surrogates is an efficient way to model unknown environments in various applications. In this paper, we propose an adaptive exploration-exploitation active learning method (ALX) that can be executed rapidly to facilitate real-time decision making. For the exploration phase, we formulate an acquisition function that maximizes the approximated, expected Fisher information. For the exploitation phase, we employ a closed-form acquisition function that maximizes the total expected variance reduction of the search space. The determination of each phase is established with an exploration condition that measures the predictive accuracy of GP surrogates. Extensive numerical experiments in multiple input spaces validate the efficiency of our method.**

I. Introduction

Autonomous agents have received considerable attention in recent years, due to novel theoretical results in machine learning and improved computational capabilities. Big data is employed from numerous autonomous systems, yet for robotics applications—that require real-time decision making—the need of scalable algorithms remains an open research problem. Gaussian processes (GPs) have shown great success in environmental monitoring [1], precision agriculture [2], controller learning [3], coverage planning [4], and communication quality [5]. The main disadvantage of GPs is the computational scalability with the size of datasets. Thus, active learning of GP surrogates seeks to formulate small and efficient datasets that produce similar prediction accuracy and uncertainty quantification to big datasets. The objective of this work is to synthesize an adaptive exploration-exploitation method that provides accurate predictions at a scalable time to enable real-time decision making.

Existing active learning methods fit a new GP surrogate model at every loop, after adding a new input/observation pair to the dataset [6]–[9]. This is a purely exploration-based strategy and exploits the GP surrogate only to determine the next input location, leading to unnecessary computations. The main idea of the proposed active learning is presented in Fig. 1. Suppose that the task is to explore an unknown spatial field and a robot so far has collected observations from a set of input locations (blue dots). Using a prediction quality condition, we determine the phase of active learning (exploration or exploitation). For the exploration phase: we fit a new GP surrogate model, employ an acquisition function to find the next sampling input location based on the new GP surrogate (dashed yellow circle), and predict the output at the next input using the new surrogate. In the exploitation phase: we employ the current GP surrogate to find the next input location (dashed green circle) and predict the output at the next input using the current surrogate. Next, the robot visits the assigned input, collects a sample, and updates the dataset. The observation and the predicted value are compared from the exploration condition to determine the accuracy of the GP surrogate and decide the next phase of active learning. For each phase, we use a different acquisition function to determine the next input, making our approach adaptive compared to current single-criterion methods [10].

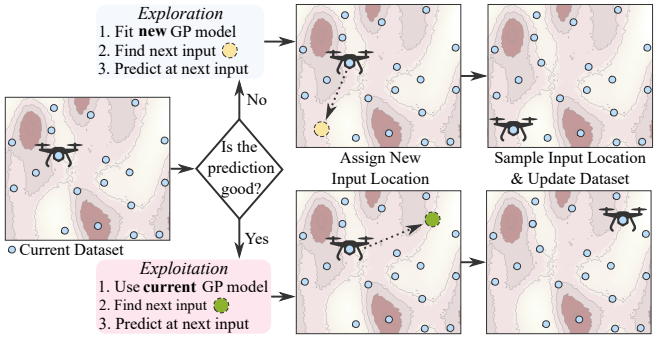

Fig. 1. Adaptive exploration-exploitation active learning of GP surrogates. The proposed method improves the computational scalability of active learning methods, while providing accurate predictions.

The complexity of a GP involves $\mathcal{O}(n^3)$ computations for training and $\mathcal{O}(n^2)$ for prediction, where n is the size of the dataset [11]. Recently, approximation methods have attracted attention as a way to reduce the computational complexity of GPs [12]–[14]. Another approach to avoid large datasets is to perform active learning and formulate a small yet efficient dataset [15, Ch. 6]. The first active learning of GP surrogates is proposed in [6] and included the active learning MacKay (ALM) and active learning Cohn (ALC) methods. Previously, we derived a closed-form gradient for the optimization of ALC, termed cALC [16]. The difference of each active learning method lies in the acquisition function that is optimized to obtain the next input location. In particular, ALM aims to find the next input by maximizing the Shannon information, while ALC seeks to find an input location that minimizes the total expected variance over the entire search space. In [10], an exploration-exploitation active learning with theoretical bounds is proposed using mutual information (ALMI). Another active learning method that intends to find input locations is by maximizing the expected Fisher information (ALFI) [17].

Active learning methods are used in various applications [18], including ALM [19], [20], ALC [7], ALMI [21],

G. P. Kontoudis and M. Otte are with the Dept. of Aerospace Engineering, University of Maryland, College Park, MD, USA. {kont,otte}@umd.edu.

This work was supported by the MRC Postdoctoral Fellowship Program and the Office of Naval Research via grant N00014-20-1-2712.

and ALFI [8], [22], [23]. Out of these methods, ALC is the most accurate non-myopic predictor, but it is computationally expensive due to the evaluation of a reference set. On the other hand, ALFI offers rapid updates, but it is designed for model estimation, ignoring the main objective of active learning, i.e., accurate predictions. Most of these methods are exploration-based in that a new GP is fitted after the acquisition of each new input/observation pair. This is computationally expensive due to GP training ($\mathcal{O}(n^3)$) and unnecessary as a new GP surrogate model is not always required. ALMI [10] employs the same acquisition function for both exploration and exploitation phases. Although ALMI is non-myopic for balancing exploration and exploitation, it is myopic for uncertainty reduction over the entire space. In this work, we introduce an adaptive exploration-exploitation active learning strategy that is scalable for onboard computations, while ensuring both accurate predictions and model estimation. The proposed active learning is non-myopic for uncertainty reduction and it is tailored with different criteria for each phase to facilitate quicker information gathering.

The contributions of this paper are: i) the synthesis of an adaptive exploration-exploitation active learning method (ALX); and ii) the formulation of an acquisition function with Fisher information (ALFIA) that approximates the likelihood function for the exploration phase. In the exploration phase we seek to acquire data that improve the GP surrogate model (ALFIA) while in exploitation phase we aim to collect data that improve the prediction accuracy (ALC). Numerical examples illustrate computational reduction as well as prediction accuracy improvement.

II. Active Learning of Gaussian Processes

In this section, we discuss Gaussian processes, review active learning, and state the problem.

A. Gaussian Processes

The sampled observations (blue dots in Fig. 1) follow,

$$y(\boldsymbol{x}) = f(\boldsymbol{x}) + \epsilon, \tag{1}$$

where $f(\boldsymbol{x}) \sim \mathcal{GP}(0, k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}'))$ is a GP with zero-mean and covariance function $k_{\boldsymbol{\theta}} : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$. The input is described as $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^D$ and the sensor noise as $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$, where σ_ϵ^2 is the variance. We select the separable squared exponential (SSE) (also known as the automatic relevance determination (ARD)) that follows,

$$k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp\left\{-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_d - x_d')^2}{l_d^2}\right\}, \tag{2}$$

where l_d is the length-scale hyperparameter associated with the d-th input dimension and σ_f^2 the signal variance. A length-scale hyperparameter describes the fluctuation of the unknown field. In particular, a high length-scale corresponds to smooth unknown field, while a low value describes highly uneven field. The SSE assigns a different hyperparameter value to each input dimension. We seek to predict the field f given a dataset $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$ with $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^n$ the inputs and $\boldsymbol{y} = \{y_i\}_{i=1}^n$ the outputs. The learning follows: i) train the hyperparameters $\boldsymbol{\theta}$; and ii) predict at unknown inputs $\boldsymbol{x}_*$.

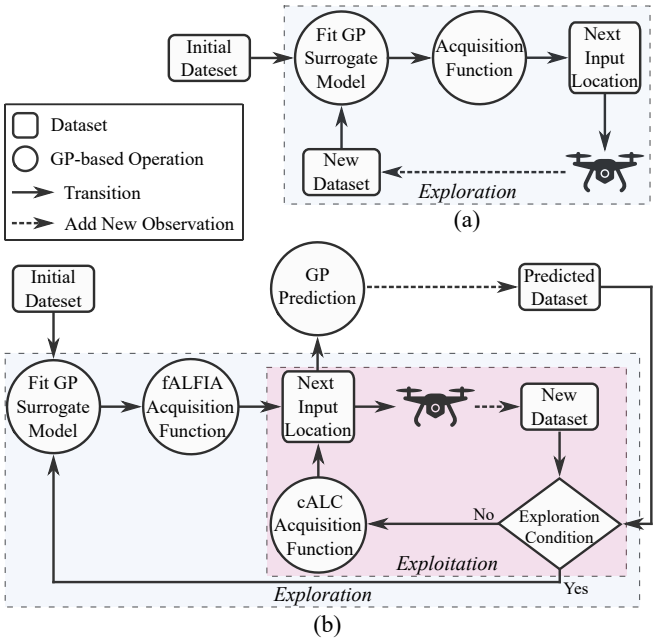

Fig. 2. (a) Current active learning of GP surrogates. (b) Proposed adaptive exploration-exploitation active learning (ALX) of GP surrogate models.

Training: The hyperparameter vector is trained by employing the maximum likelihood estimation (MLE) method with log-likelihood function,

$$\ln p(\boldsymbol{y} \mid \boldsymbol{X}) = -\frac{1}{2}\boldsymbol{y}^\intercal \boldsymbol{C}_n^{-1}\boldsymbol{y} - \frac{1}{2}\ln|\boldsymbol{C}_n| - \frac{1}{2}n\ln 2\pi,$$

where $\boldsymbol{C}_n = \boldsymbol{K} + \sigma_\epsilon^2 I_n$ is the covariance matrix with $\boldsymbol{K} = k_{\boldsymbol{\theta}}(\boldsymbol{X}, \boldsymbol{X}) \in \mathbb{R}^{n\times n}$ the correlation matrix and $\boldsymbol{\theta} = (l_1, \ldots, l_D, \sigma_f, \sigma_\epsilon)^\intercal \in \mathbb{R}^{D+2}$ the hyperparameters.

Prediction: Using the estimated hyperparameters $\hat{\boldsymbol{\theta}}$, the posterior distribution of an unsampled input $\boldsymbol{x}_* \in \mathbb{R}^D$ results in a multi-variate normal distribution $p(y_* \mid \mathcal{D}, \boldsymbol{x}_*) \sim \mathcal{N}(\mu(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*))$ with prediction mean and variance,

$$\mu_{\text{full}}(\boldsymbol{x}_*) = \boldsymbol{k}_*^\intercal \boldsymbol{C}_n^{-1}\boldsymbol{y},$$
$$\sigma_{\text{full}}^2(\boldsymbol{x}_*) = k_{**} - \boldsymbol{k}_*^\intercal \boldsymbol{C}_n^{-1}\boldsymbol{k}_*,$$

where $\boldsymbol{k}_* = k_{\boldsymbol{\theta}}(\boldsymbol{X}, \boldsymbol{x}_*) \in \mathbb{R}^n$ and $k_{**} = k_{\boldsymbol{\theta}}(\boldsymbol{x}_*, \boldsymbol{x}_*) \in \mathbb{R}$.

Complexity: The training entails cubic computations $\mathcal{O}(n^3)$ to calculate the inverse of the covariance $\boldsymbol{C}_n^{-1}$. At every iteration of the MLE a new inverse of the covariance is needed as the covariance matrix is a function of $\boldsymbol{\theta}$. The prediction is executed in quadratic time $\mathcal{O}(n^2)$.

B. Active Learning

Active learning is a sequential adaptive sampling method that aims to identify new sampling locations by minimizing an acquisition function based on a GP surrogate model. The main idea is to carefully select new informative sampling locations and form a dataset that achieves high prediction accuracy even with a small dataset size [24]. Subsequently, small dataset size leads to reduced computation demands (e.g., $\mathcal{O}(n^3)$ for GP training and $\mathcal{O}(n^2)$ for GP prediction).

A block diagram of the current structure for active learning of GP surrogate models is presented in Fig. 2-(a). The first step is to form an initial dataset $\mathcal{D}_{\text{init}} = (\boldsymbol{X}_{\text{init}}, \boldsymbol{y}_{\text{init}})$. The inputs of the initial dataset $\boldsymbol{X}_{\text{init}}$ are pre-selected as a batch, because small dataset sizes produce low fidelity GP models and we do not trust them to make decisions for new sampling locations. Using the initial dataset $\mathcal{D}_{\text{init}}$, we fit a GP surrogate model to estimate the hyperparameters $\hat{\boldsymbol{\theta}}_n$. The training of the GP hyperparameters is performed with the MLE method which is expressed as a constraint optimization problem with upper and lower bound constraints. Next, the fitted GP surrogate model feeds the active learning criterion or acquisition function to determine the next input location $\boldsymbol{x}_{n+1}$. There are numerous active learning criteria that can be used and we analyze some of the most successful in the next sections. The next sampling input $\boldsymbol{x}_{n+1}$ is assigned to an agent which visits that location to collect an observation y_{n+1}. After collecting the new observation, the agent updates the dataset by adding the new observation (dashed line in Fig. 2-(a)). The new dataset is the union of the previous and current collected data $\mathcal{D}_{n+1} = \mathcal{D}_n \cup (y_{n+1}, \boldsymbol{x}_{n+1})$. Finally, the new dataset $\mathcal{D}_{n+1}$ is used to fit a new GP surrogate model. This loop continues for a predetermined number of iterations.

To avoid unnecessary sampling, we inherit an empirical rule to determine the initial dataset size that is ten times the input dimension $n_{\text{init}} = 10D$ [18], [25]. The initial input locations $\boldsymbol{X}_{\text{init}} = \{\boldsymbol{x}_k\}_{k=1}^{10D} \in \mathbb{R}^{D\times 10D}$ can be obtained either by randomly sampling the area of interest (i.e., $[\boldsymbol{x}_k]_{ij} \sim \mathcal{U}_{[0,1]}$ for normalized inputs) or by using a model-free space filling method [15, Ch. 4]. In this work, we use a space filling sampling method; the Latin hypercube sampling (LHS) $\boldsymbol{X}_{\text{init}} = \boldsymbol{X}_{\text{LHS}}$ that has similar properties to the uniform distribution [26], [27]. Despite the progress in space filling methods, LHS remains competitive [15], [28]. The Latin hypercube matrix $\boldsymbol{L} \in \mathbb{R}^{D\times 10D}$ is composed by D levels of permutation and $10D$ columns, where each entry follows $[\boldsymbol{X}_{\text{LHS}}]_{ij} = ([\boldsymbol{L}]_{ij} + (D-1)/2 + [\boldsymbol{u}]_{ij})/D$ with $[\boldsymbol{u}]_{ij} \sim \mathcal{U}_{[0,1]}$ for normalized inputs. After obtaining the initial inputs $\boldsymbol{X}_{\text{init}}$, the agent samples the input locations $\boldsymbol{y}_{\text{init}}$ to form the initial dataset $\mathcal{D}_{\text{init}} = (\boldsymbol{X}_{\text{init}}, \boldsymbol{y}_{\text{init}})$.

Notably, every loop of active learning in Fig. 2-(a) is an exploration loop as the GP surrogate is used only to determine the next sampling location $\boldsymbol{x}_{n+1}$. This exploration-based active learning scheme is expensive for two reasons. First, the GP training is performed at every loop and entails $\mathcal{O}(n^3)$ computations. Second, some active learning criteria (e.g., active learning Cohn (ALC)) are also computational expensive as they require evaluation on several locations of the reference dataset. As a result, the current formulation of active learning—combines the fit of a GP surrogate model (i.e., GP training) and the evaluation of specific acquisition functions—prohibits real-time decision making.

Problem 1: Synthesize an active learning framework that achieves high prediction accuracy and precise model estimation with reduced computations to facilitate real-time implementation for robotics applications.

III. Proposed Active Learning Method

A. Active Learning Cohn (ALC)

The active learning Cohn (ALC) employs the negative expected reduction in variance over the input space as an acquisition function. In other words, ALC seeks for an input location that if observed and added to the dataset, it reduces the variance of the search space more than any other pair of input/observation. Formally, the acquisition function of the ALC is defined as $J_{\text{ALC}}(\hat{\boldsymbol{\theta}}_n; \boldsymbol{x}) := -\int_{\mathcal{X}} \Delta\sigma^2(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}$, where $\Delta\sigma^2(\boldsymbol{x}) = \sigma_n^2(\boldsymbol{x}) - \widetilde{\sigma}_{n+1}^2(\boldsymbol{x})$ is the deduced variance, with $\sigma_n^2(\boldsymbol{x})$ the variance using the current dataset, and $\widetilde{\sigma}_{n+1}^2(\boldsymbol{x})$ the approximated variance using the estimated hyperparameters from the previous step $\hat{\boldsymbol{\theta}}_n$ and assuming that the candidate location $\boldsymbol{x}_{n+1}$ is added in the dataset. Since the integral of the acquisition function J_{ALC} cannot be computed analytically, we employ an approximation with a summation over a reference set $\boldsymbol{X}_{\text{ref}} \in \mathbb{R}^{N_{\text{ref}}\times D}$. The reference set is formulated as a LHS $\boldsymbol{X}_{\text{ref}} = \boldsymbol{X}_{\text{LHS}}$, similarly to the initial dataset. The ALC optimization problem yields,

$$\begin{aligned}\boldsymbol{x}_{n+1} &= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ -\int_{\mathcal{X}} \Delta\sigma^2(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \right\} \\ &\approx \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ -\frac{1}{N_{\text{ref}}} \sum_{t=1}^{N_{\text{ref}}} \Delta\sigma^2(\boldsymbol{x}, \boldsymbol{x}_t) \right\} \\ &= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ -\frac{1}{N_{\text{ref}}} \sum_{t=1}^{N_{\text{ref}}} \sigma_n^2(\boldsymbol{x}_t) - \widetilde{\sigma}_{n+1}^2(\boldsymbol{x}, \boldsymbol{x}_t) \right\} \quad (3)\end{aligned}$$

The variances $\sigma_n^2(\boldsymbol{x}_t)$ and $\widetilde{\sigma}_{n+1}^2(\boldsymbol{x}, \boldsymbol{x}_t)$ are provided by,

$$\begin{aligned}\sigma_n^2(\boldsymbol{x}_t) &= k_n - \boldsymbol{k}_{n,t}^{\intercal} \boldsymbol{C}_n^{-1} \boldsymbol{k}_{n,t}, \\ \widetilde{\sigma}_{n+1}^2(\boldsymbol{x}, \boldsymbol{x}_t) &= k_n - \boldsymbol{k}_{n+1,t}^{\intercal} \boldsymbol{C}_{n+1}^{-1} \boldsymbol{k}_{n+1,t},\end{aligned}$$

where $k_n = k_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{x}_t) \in \mathbb{R}$, $\boldsymbol{k}_{n,t} = k_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{X}_n) \in \mathbb{R}^n$, $\boldsymbol{k}_{n+1,t} = k_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{X}_{n+1}) \in \mathbb{R}^{n+1}$, $\boldsymbol{C}_n = k_{\boldsymbol{\theta}}(\boldsymbol{X}_n, \boldsymbol{X}_n) \in \mathbb{R}^{n\times n}$, $\boldsymbol{C}_{n+1} = k_{\boldsymbol{\theta}}(\boldsymbol{X}_{n+1}, \boldsymbol{X}_{n+1}) \in \mathbb{R}^{(n+1)\times(n+1)}$ with $\boldsymbol{X}_{n+1} = [\boldsymbol{X}_n^{\intercal} \quad \boldsymbol{x}]^{\intercal} \in \mathbb{R}^{n+1}$. The elements of the new covariance matrix $\boldsymbol{C}_{n+1}$ are,

$$\boldsymbol{C}_{n+1} = \begin{bmatrix} \boldsymbol{C}_n & \boldsymbol{k}_{n+1} \\ \boldsymbol{k}_{n+1}^{\intercal} & k_{n+1,n+1} \end{bmatrix},$$

where $\boldsymbol{k}_{n+1} = k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{X}_n) \in \mathbb{R}^n$ and $k_{n+1,n+1} = k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}) \in \mathbb{R}$.

Proposition 1: [7, Appendix A.1] The ALC acquisition function of the optimization (3) takes the form of,

$$J_{\text{ALC}}(\hat{\boldsymbol{\theta}}_n; \boldsymbol{x}, \boldsymbol{x}_t) = -\frac{1}{N_{\text{ref}}} \sum_{t=1}^{N_{\text{ref}}} \frac{\left(\boldsymbol{k}_{n,t}^{\intercal} \boldsymbol{C}_n^{-1} \boldsymbol{k}_{n+1} - k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}_t)\right)^2}{k_{n+1,n+1} - \boldsymbol{k}_{n+1}^{\intercal} \boldsymbol{C}_n^{-1} \boldsymbol{k}_{n+1}}. \quad (4)$$

□

Typically, we use the finite difference method (FDM) to approximate the gradient and solve the optimization problem,

$$\frac{\partial J(\hat{\boldsymbol{\theta}}_n; \boldsymbol{x}, \boldsymbol{x}_t)}{\partial [\boldsymbol{x}]_d} \approx \frac{J(\hat{\boldsymbol{\theta}}_n; \boldsymbol{x} + \boldsymbol{s}_d, \boldsymbol{x}_t) - J(\hat{\boldsymbol{\theta}}_n; \boldsymbol{x} - \boldsymbol{s}_d, \boldsymbol{x}_t)}{2s} \quad (5)$$

where $\boldsymbol{s}_d = [\boldsymbol{0}_{1:d-1}^{\intercal} \quad s \quad \boldsymbol{0}_{d+1:D}^{\intercal}]^{\intercal} \in \mathbb{R}^D$. The ALC with

approximated gradient using FDM is termed as fALC. The fALC is reliable in low dimensional input spaces and when the spacing parameter s is accurately selected [29, Ch. 7]. However, in high dimensional input spaces fALC is not only inaccurate, but also computationally expensive as it requires $2D+1$ evaluations of the acquisition function at a collocated grid. In [16], we showed that the gradient of the acquisition function for the ALC can be computed analytically (cALC). cALC produces more accurate results than fALC with similar computations for high dimensional input spaces.

Proposition 2: [16] The closed-form gradient of the acquisition function (4) for the ALC optimization (3) with the SSE covariance function (2) yields,

$$\frac{\partial J(\hat{\boldsymbol{\theta}}_n;\boldsymbol{x},\boldsymbol{x}_t)}{\partial \boldsymbol{x}} = \frac{\nabla_{\boldsymbol{x}} h(\hat{\boldsymbol{\theta}}_n;\boldsymbol{x}) \sum_{t=1}^{N_{\text{ref}}} g_t(\hat{\boldsymbol{\theta}}_n;\boldsymbol{x},\boldsymbol{x}_t)}{N_{\text{ref}} h(\hat{\boldsymbol{\theta}}_n;\boldsymbol{x})^2} - \frac{h(\hat{\boldsymbol{\theta}}_n;\boldsymbol{x}) \sum_{t=1}^{N_{\text{ref}}} \nabla_{\boldsymbol{x}} g_t(\hat{\boldsymbol{\theta}}_n;\boldsymbol{x},\boldsymbol{x}_t)}{N_{\text{ref}} h(\hat{\boldsymbol{\theta}}_n;\boldsymbol{x})^2}.$$

The elements h, g, $\nabla_{\boldsymbol{x}} h$, $\nabla_{\boldsymbol{x}} g_t$ are provided by,

$$\begin{aligned}
h(\hat{\boldsymbol{\theta}}_n;\boldsymbol{x}) &= k_{n+1,n+1} - \boldsymbol{k}_{n+1}^\intercal \boldsymbol{C}_n^{-1} \boldsymbol{k}_{n+1},\\
\frac{\partial h(\hat{\boldsymbol{\theta}}_n;\boldsymbol{x})}{\partial \boldsymbol{x}} &= 4\left((\boldsymbol{1}_D \boldsymbol{k}_{n+1}^\intercal) \odot (\Lambda^{-1}\Delta \boldsymbol{X}^\intercal)\right) \boldsymbol{C}_n^{-1} \boldsymbol{k}_{n+1},\\
g_t(\hat{\boldsymbol{\theta}}_n;\boldsymbol{x},\boldsymbol{x}_t) &= \left(\boldsymbol{k}_{n,t}^\intercal \boldsymbol{C}_n^{-1} \boldsymbol{k}_{n+1} - k_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{x}_t)\right)^2,\\
\frac{\partial g_t(\hat{\boldsymbol{\theta}}_n;\boldsymbol{x},\boldsymbol{x}_t)}{\partial \boldsymbol{x}} &= -4\left(\boldsymbol{k}_{n,t}^\intercal \boldsymbol{C}_n^{-1} \boldsymbol{k}_{n+1} - k_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{x}_t)\right) \Lambda^{-1} \times \\
&\quad \Big(\Delta \boldsymbol{X}^\intercal \left((\boldsymbol{k}_{n,t}^\intercal \boldsymbol{C}_n^{-1})^\intercal \odot \boldsymbol{k}_{n+1}\right) \\
&\quad - k_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{x}_t)(\boldsymbol{x}-\boldsymbol{x}_t)\Big),
\end{aligned}$$

where $\odot$ is the Hadamard product, $\Lambda = \text{diag}(l_1^2, l_2^2, \ldots, l_D^2)$, $\Delta \boldsymbol{X} = [\boldsymbol{x} - [\boldsymbol{X}_n]_1, \ldots, \boldsymbol{x} - [\boldsymbol{X}_n]_n]^\intercal \in \mathbb{R}^{n\times D}$, and $\boldsymbol{1}_D = [1,\ldots,1]^\intercal \in \mathbb{R}^D$. ▫

B. Active Learning Fisher Approximation (ALFIA)

The active learning Fisher approximation (ALFIA) uses the expected Fisher information matrix of the lengthscale hyperparameters $\boldsymbol{l}$ from (2) on an approximated log-likelihood function. Essentially, the Fisher information matrix represents the expected value of the observed information by computing the negative second derivative of the log-likelihood function with respect to $\boldsymbol{l}$. The acquisition function of active learning with Fisher information (ALFI) is defined as,

$$\begin{aligned}
J_{\text{ALFI}}(\hat{\boldsymbol{l}}_n;\boldsymbol{x}) &\coloneqq -\left|\mathcal{F}_{n+1}(\hat{\boldsymbol{l}}_n;\boldsymbol{x})^{-1}\right| \\
&= -\left|\mathrm{E}\left[-\frac{\partial^2 \mathcal{L}_{n+1}(\boldsymbol{y}_{n+1} \mid \hat{\boldsymbol{l}}_n;\boldsymbol{x})}{\partial \boldsymbol{l}^2}\right]^{-1}\right|, \quad (6)
\end{aligned}$$

where $\mathcal{F}_{n+1} \in \mathbb{R}^{D\times D}$ is the total expected Fisher information matrix and $\mathcal{L}_{n+1}$ is the total expected log-likelihood function if we hypothetically add a pair of input/observation $(\boldsymbol{x}_{n+1}, y_{n+1})$ in the dataset $\mathcal{D}_n$. The inverse of the Fisher information is the lower bound of the Cramér-Rao inequality [30]. This implies that the maximization of the Fisher information is optimal as it minimizes the variance of the lengthscale estimation $\text{VAR}[\hat{\boldsymbol{l}}_n]$. When we have multiple lengthscales the Fisher information is expressed as a matrix and the minimization of its determinant is called D-optimality. The total expected log-likelihood yields,

$$\begin{aligned}
\mathcal{L}_{n+1}(\boldsymbol{y}_{n+1} \mid \hat{\boldsymbol{l}}_n) &= \ln p(\boldsymbol{y}_{n+1} \mid \hat{\boldsymbol{l}}_n; \boldsymbol{X}_n, \boldsymbol{x}) \\
&= \ln p(\boldsymbol{y}_n \mid \hat{\boldsymbol{l}}_n; \boldsymbol{X}_n) p(y_{n+1} \mid \hat{\boldsymbol{l}}_n; \boldsymbol{x}) \\
&= \ln p(\boldsymbol{y}_n \mid \hat{\boldsymbol{l}}_n; \boldsymbol{X}_n) + \ln p(y_{n+1} \mid \hat{\boldsymbol{l}}_n; \boldsymbol{x}) \\
&= L_n(\hat{\boldsymbol{l}}_n; \boldsymbol{X}_n) + L_{n+1}(\hat{\boldsymbol{l}}_n; \boldsymbol{x}). \quad (7)
\end{aligned}$$

Thus, the acquisition function (6) by using (7) yields,

$$\begin{aligned}
J_{\text{ALFI}}(\hat{\boldsymbol{l}}_n; \boldsymbol{X}_n, \boldsymbol{x}) &= -\left|\mathrm{E}\left[-\left(\frac{\partial^2 L_n}{\partial \boldsymbol{l}^2} + \frac{\partial^2 L_{n+1}}{\partial \boldsymbol{l}^2}\right)\right]^{-1}\right| \\
&= -\left|\left(\mathrm{E}\left[-\frac{\partial^2 L_n}{\partial \boldsymbol{l}^2}\right] + \mathrm{E}\left[-\frac{\partial^2 L_{n+1}}{\partial \boldsymbol{l}^2}\right]\right)^{-1}\right| \\
&= -\left|\Big(F_n(\hat{\boldsymbol{l}}_n; \boldsymbol{X}_n) + F_{n+1}(\hat{\boldsymbol{l}}_n; \boldsymbol{x})\Big)\right|^{-1}. \quad (8)
\end{aligned}$$

The elements of the first term $F_n \in \mathbb{R}^{D\times D}$ of the total expected Fisher information matrix $\mathcal{F}_{n+1}$ are computed by,

$$\begin{aligned}
[F_n]_{ij} &= \left[\mathrm{E}\left[-\frac{\partial^2 L_n}{\partial l_i \partial l_j}\right]\right]_{ij} \\
&= \frac{1}{2}\text{tr}\left\{\boldsymbol{C}_n^{-1}\left(\frac{\partial^2 \boldsymbol{C}_n}{\partial l_i \partial l_j} - \frac{\partial \boldsymbol{C}_n}{\partial l_j}\boldsymbol{C}_n^{-1}\frac{\partial \boldsymbol{C}_n}{\partial l_i}\right)\right\} + \\
&\frac{0.5n}{\boldsymbol{y}_n^\intercal \boldsymbol{C}_n^{-1} \boldsymbol{y}_n} \boldsymbol{y}_n^\intercal \boldsymbol{C}_n^{-1}\left(2\frac{\partial \boldsymbol{C}_n}{\partial l_j}\boldsymbol{C}_n^{-1}\frac{\partial \boldsymbol{C}_n}{\partial l_i} - \frac{\partial^2 \boldsymbol{C}_n}{\partial l_i \partial l_j}\right)\boldsymbol{C}_n^{-1}\boldsymbol{y}_n - \\
&\frac{0.5n}{(\boldsymbol{y}_n^\intercal \boldsymbol{C}_n^{-1} \boldsymbol{y}_n)^2}\left(\boldsymbol{y}_n^\intercal \boldsymbol{C}_n^{-1}\frac{\partial \boldsymbol{C}_n}{\partial l_i}\boldsymbol{C}_n^{-1}\boldsymbol{y}_n\right)\left(\boldsymbol{y}_n^\intercal \boldsymbol{C}_n^{-1}\frac{\partial \boldsymbol{C}_n}{\partial l_j}\boldsymbol{C}_n^{-1}\boldsymbol{y}_n\right) \quad (9)
\end{aligned}$$

where $\partial \boldsymbol{C}_n/\partial l_i \in \mathbb{R}^{n\times n}$ and $\partial^2 \boldsymbol{C}_n/\partial l_i \partial l_j \in \mathbb{R}^{n\times n}$. There exist d matrices of $\partial \boldsymbol{C}_n/\partial l_i$, d matrices of $\partial \boldsymbol{C}_n/\partial l_j$, and $d\times d$ matrices of $\partial^2 \boldsymbol{C}_n/\partial l_i \partial l_j$ for the computation of (9). If we assume that the expected log-likelihood function L_{n+1} is a Gaussian distribution then the second term of the expected Fisher information matrix F_{n+1} in (8) can be computed as in [8], [23], [31], [32]. Unfortunately, the expected log-likelihood function L_{n+1} is not a Gaussian distribution, but a Student's-t distribution because the active learning is implemented with small dataset sizes. Since the log-likelihood of a Student's-t distribution does not admit a closed-form solution, we need to approximate the expected log-likelihood function L_{n+1}. We approximate L_{n+1} by using a Gaussian surrogate model with matched moments [33] that yields,

$$\begin{aligned}
L_{n+1}(\hat{\boldsymbol{l}}_n;\boldsymbol{x}) &= \ln p(y_{n+1} \mid \hat{\boldsymbol{l}}_n;\boldsymbol{x}) \\
&\approx -\frac{1}{2}\ln \boldsymbol{y}_n^\intercal \boldsymbol{C}_n^{-1}\boldsymbol{y}_n \left(k_{n+1,n+1} - \boldsymbol{k}_{n+1}^\intercal \boldsymbol{C}_n^{-1}\boldsymbol{k}_{n+1}\right) - \\
&\quad \frac{(n-2)(y_{n+1} - \boldsymbol{k}_{n+1}^\intercal \boldsymbol{C}_n^{-1}\boldsymbol{y}_n)^2}{2\boldsymbol{y}_n^\intercal \boldsymbol{C}_n^{-1}\boldsymbol{y}_n \left(k_{n+1,n+1} - \boldsymbol{k}_{n+1}^\intercal \boldsymbol{C}_n^{-1}\boldsymbol{k}_{n+1}\right)} + c =: \tilde{L}_{n+1},
\end{aligned}$$

where $\tilde{L}_{n+1}$ is the approximated, expected log-likelihood function and $c = -0.5(\ln 2\pi - \ln(n-2))$ represents all constant terms.

The ALFIA optimization problem takes the form of,

$$\begin{aligned}\boldsymbol{x}_{n+1} &= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ -\left| \left(F_n(\hat{\boldsymbol{l}}_n) + \mathrm{E}\left[-\frac{\partial^2 L_{n+1}}{\partial \boldsymbol{l}^2} \right] \right)^{-1} \right| \right\} \\ &\approx \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ -\left| \left(F_n(\hat{\boldsymbol{l}}_n) + \mathrm{E}\left[-\frac{\partial^2 \tilde{L}_{n+1}}{\partial \boldsymbol{l}^2} \right] \right)^{-1} \right| \right\} \\ &= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ -\left| \left(F_n(\hat{\boldsymbol{l}}_n) + \tilde{F}_{n+1}(\hat{\boldsymbol{l}}_n; \boldsymbol{x}) \right) \right|^{-1} \right\}, \quad (10)\end{aligned}$$

where $\tilde{F}_{n+1} \in \mathbb{R}^{D\times D}$ is the approximated expected Fisher information matrix.

Proposition 3: The ALFIA criterion yields,

$$J_{\text{ALFIA}}(\hat{\boldsymbol{l}}_n; \boldsymbol{x}) = -\left| \left(F_n(\hat{\boldsymbol{l}}_n) + \tilde{F}_{n+1}(\hat{\boldsymbol{l}}_n; \boldsymbol{x}) \right) \right|^{-1}. \quad (11)$$

The elements of the first term F_n are computed by (9) and the elements second term $\tilde{F}_{n+1}$ follow,

$$\begin{aligned}\left[\tilde{F}_{n+1}\right]_{ij} &= \left[\mathrm{E}\left[-\frac{\partial^2 \tilde{L}_{n+1}}{\partial l_i \partial l_j} \right] \right]_{ij} \\ &= \frac{W_{n,i} W_{n,j}}{2\left(\boldsymbol{y}_n^\intercal \boldsymbol{C}_n^{-1} \boldsymbol{y}_n \left(k_{n+1,n+1} - \boldsymbol{k}_{n+1}^\intercal \boldsymbol{C}_n^{-1} \boldsymbol{k}_{n+1} \right) \right)^2} \\ &\quad + \frac{V_{n,i} V_{n,j}}{\boldsymbol{y}_n^\intercal \boldsymbol{C}_n^{-1} \boldsymbol{y}_n \left(k_{n+1,n+1} - \boldsymbol{k}_{n+1}^\intercal \boldsymbol{C}_n^{-1} \boldsymbol{k}_{n+1} \right)}. \quad (12)\end{aligned}$$

The $W_{n,i} \in \mathbb{R}$, $V_{n,i} \in \mathbb{R}$ are provided by,

$$\begin{aligned}W_{n,i} = {} & \boldsymbol{y}_n^\intercal \boldsymbol{C}_n^{-1} \frac{\partial \boldsymbol{C}_n}{\partial l_i} \boldsymbol{C}_n^{-1} \boldsymbol{y}_n \left(k_{n+1,n+1} - \boldsymbol{k}_{n+1}^\intercal \boldsymbol{C}_n^{-1} \boldsymbol{k}_{n+1} \right) + \\ & \boldsymbol{y}_n^\intercal \boldsymbol{C}_n^{-1} \boldsymbol{y}_n \left(\frac{\partial \boldsymbol{k}_{n+1}^\intercal}{\partial l_i} \boldsymbol{C}_n^{-1} \boldsymbol{k}_{n+1} + \right. \\ & \left. \boldsymbol{k}_{n+1}^\intercal \boldsymbol{C}_n^{-1} \left(\frac{\partial \boldsymbol{k}_{n+1}}{\partial l_i} - \frac{\partial \boldsymbol{C}_n}{\partial l_i} \boldsymbol{C}_n^{-1} \boldsymbol{k}_{n+1} \right) \right), \quad (13)\end{aligned}$$

$$V_{n,i} = \left(\boldsymbol{C}_n^{-1} \left(\frac{\partial \boldsymbol{k}_{n+1}}{\partial l_i} - \frac{\partial \boldsymbol{C}_n}{\partial l_i} \boldsymbol{C}_n^{-1} \boldsymbol{k}_{n+1} \right) \right)^\intercal \boldsymbol{y}_n, \quad (14)$$

where $\partial \boldsymbol{k}_{n+1}/\partial l_i \in \mathbb{R}^n$ and $\partial \boldsymbol{C}_n/\partial l_i \in \mathbb{R}^{n\times n}$. Similarly, $W_{n,j} \in \mathbb{R}$, $V_{n,j} \in \mathbb{R}$ are computed by replacing l_i in (13), (14) with l_j. There exist d vectors of $\partial \boldsymbol{k}_{n+1}/\partial l_i$, d vectors of $\partial \boldsymbol{k}_{n+1}/\partial l_j$, d matrices of $\partial \boldsymbol{C}_n/\partial l_i$, and d matrices of $\partial \boldsymbol{C}_n/\partial l_j$ for the computation of (12). ▫

For the optimization of ALFIA (10) we employ the FDM method (5) similarly to the ALC problem in Section III-A. We term the ALFIA with approximated gradient as fALFIA.

C. Adaptive Exploration-Exploitation Active Learning (ALX)

In this section, we present the proposed adaptive exploration-exploitation active learning (ALX) method that employs both cALC and fALFIA. ALX is designed to achieve both accurate predictions and precise model estimation, while requiring scalable computations. A comparison of cALC, fALFIA, and ALX methods is shown in Table I.

Active learning methods are efficient in different fields depending on the acquisition function. The acquisition functions are designed to achieve specific goals, but they are not universal for all applications. To this end, they are usually selected or designed based on the desired goals. For example, the acquisition function of ALC is designed to improve the prediction accuracy. On the other hand, the ALFIA acquisition function is designed to estimate the hyperparameters of the GP surrogate model. Although ALC has significant capabilities on prediction accuracy, it does not ensure accurate model estimation. Similarly, ALFIA can accurately estimate the hyperparameters of the GP surrogate model, yet it usually provides inaccurate predictions.

TABLE I
QUALITATIVE COMPARISON OF ACTIVE LEARNING METHODS

	cALC	fALFIA	ALX
Prediction quality	Accurate	Moderate	Accurate
Model estimation	Moderate	Accurate	Accurate
Fit GP surrogate	Continuous	Continuous	Intermittent
Computations	High	Moderate	Low
Phase	Explor.	Explor.	Explor./Explot.
Acquisition fun.	ALC	ALFIA	ALC/ALFIA

In addition to design properties, computational scalability is of significant importance especially for robotic missions that require rapid decision making with onboard resources. Fitting a GP surrogate model at every loop entails substantial computations. Moreover, some acquisition functions have high computational demands for their evaluation. In particular, the ALC acquisition function is expensive due to the evaluation of the reference set. On the contrary, the acquisition function of ALFIA can be rapidly executed with a single evaluation, yet even ALFIA may become intractable with continuous fit of a new GP surrogate model at every loop. Both active learning methods (cALC and fALFIA) require continuous fit of a GP surrogate model at every loop as shown in Fig. 2-(a). This results in purely exploration-based active learning, because the surrogate is not exploited for other tasks. The only utility of the surrogate appears when the active learning criterion computes the next input location.

The block diagram of the proposed ALX method is presented in Fig. 2-(b). The exploration phase is emphasized with a light blue box and the exploitation phase with a light red box. The first two steps are similar to a typical active learning method. We form an initial dataset $\mathcal{D}_{\text{init}}$ with $n_{\text{init}} = 10D$ and then we fit a GP surrogate model. Next, the GP surrogate model is provided to the fALFIA acquisition function. The selection of the fALFIA acquisition function for the initial determination of the next input location is attributed to two reasons: 1) the fALFIA criterion is better at exploring the search space and obtaining an accurate GP surrogate model that can then be exploited to improve prediction accuracy; and 2) the fitting of a GP surrogate model is already computationally expensive, thus we employ the least expensive acquisition function fALFIA—compared to cALC—to alleviate the total computations per loop.

After determining the next input location, the ALX not only assigns $\boldsymbol{x}_{n+1}$ to an agent for the observation collection y_{n+1}, but also takes an additional concurrent step by predicting the observation $\hat{y}_{n+1}$ at the assigned location $\boldsymbol{x}_{n+1}$

with the current GP surrogate model hyperparameters $\hat{\boldsymbol{\theta}}_n$. Since the inverse of the covariance matrix $\boldsymbol{C}_n^{-1}$ is already computed from fitting the GP surrogate model, the GP prediction entails relatively low computations of $\mathcal{O}(n^2)$. We now have two datasets of the same size, the new dataset $\mathcal{D}_{n+1} = \mathcal{D}_n \cup (\boldsymbol{x}_{n+1}, y_{n+1}) \in \mathbb{R}^{(n+1)\times D}$ and the predicted dataset $\hat{\mathcal{D}}_{n+1} = \mathcal{D}_n \cup (\boldsymbol{x}_{n+1}, \hat{y}_{n+1}) \in \mathbb{R}^{(n+1)\times D}$. These two datasets are compared by an exploration condition to measure the discrepancy of the predicted observation $\hat{y}_{n+1}$ and the sampled observation y_{n+1}. In particular, the exploration condition evaluates the inequality,

$$|\hat{y}_{n+1} - y_{n+1}| > \beta \left(\max(\boldsymbol{y}_{n+1}) - \min(\boldsymbol{y}_{n+1})\right), \qquad (15)$$

where $\max(\boldsymbol{y}_{n+1})$ is the maximum value, $\min(\boldsymbol{y}_{n+1})$ the minimum value from the set of observations, and $\beta \in [0, 1]$ the user-defined exploration cost value. High exploration cost values β promote exploitation, while low β favor exploration. Thus, if the exploration condition (15) is satisfied, then the exploration continues by fitting a new GP surrogate model with the new dataset $\mathcal{D}_{n+1}$, otherwise the same GP surrogate model with hyperparameters $\hat{\boldsymbol{\theta}}_n$ is exploited to find the next input location. In other words, the exploration condition evaluates the absolute error between the prediction using the current model and the collected observation. If the prediction error is higher than a threshold, then we deduct that the model is inaccurate and fit a new GP surrogate (exploration), otherwise we use the current surrogate (exploitation).

For the exploitation phase the new dataset is provided to the cALC acquisition function that exploits the current GP surrogate model with $\hat{\boldsymbol{\theta}}_n$. The selection of the cALC acquisition function for the exploitation phase has a twofold reasoning: 1) the cALC criterion is better at improving the prediction accuracy as long as it has an accurate GP surrogate model; and 2) the evaluation of cALC is isolated from the GP model fit to reduce the computations. Essentially, when a new GP surrogate model is required we use the fALFIA acquisition function that is better at upgrading the model estimation accuracy (exploration) and when an accurate model exists we exploit the surrogate and use cALC which is better at improving the prediction accuracy (exploitation).

Implementation details of the proposed ALX method are provided in Alg. 1. The first step is to compute a multi-start location dataset $\boldsymbol{X}_{\text{m-s}}$ by using the `conditionLHS` routine that works similarly to LHS (Section II-B), conditioned on the current input locations $\boldsymbol{X}_n$. Since both acquisitions functions of ALC (4) and ALFIA (11) are non-convex on the optimizing parameter $\boldsymbol{x} \in \mathcal{X}$, the multi-start location dataset $\boldsymbol{X}_{\text{m-s}}$ ensures that local minima are avoided. Next, the exploration condition (15) is utilized to decide the phase of active learning (exploration or exploitation). For the exploration phase (lines 4-7), a new GP surrogate model is fitted to obtain the hyperparameters $\hat{\boldsymbol{\theta}}_n$ and the inverse covariance $\boldsymbol{C}_n^{-1}$. Next, fALFIA is used as described in Proposition 3 to output the next sample location candidates $\{\boldsymbol{x}_{n+1,i}\}_{i=1}^{N_{\text{m-s}}}$ and the corresponding acquisition function values $\{J_{n+1,i}\}_{i=1}^{N_{\text{m-s}}}$ for all multi-start locations using the new hyperparameters.

Algorithm 1 Exploration-Exploitation AL (ALX)

Input: $\mathcal{D}_n = (\boldsymbol{X}_n, \boldsymbol{y}_n)$, D, $N_{\text{m-s}}$, N_{ref}, $n_{\max}$, β, γ, η

Output: $\hat{\boldsymbol{\theta}}_{n_{\max}}$, $\mathcal{D}_{n_{\max}}$

1: **repeat**
2: $\quad\boldsymbol{X}_{\text{m-s}} \leftarrow$ `conditionLHS`$(N_{\text{m-s}}, D, \boldsymbol{X}_n)$; $\{\boldsymbol{x}_i^{(1)}\}_{i=1}^{N_{\text{m-s}}} = \boldsymbol{X}_{\text{m-s}}$
3: $\quad$**if** $|\hat{y}_n - y_n| > \beta\,(\max(\boldsymbol{y}_n) - \min(\boldsymbol{y}_n))$ **then** ▷ Exploration
4: $\quad\quad\hat{\boldsymbol{\theta}}_n, \boldsymbol{C}_n^{-1} \leftarrow$ `GPtraining`$(\mathcal{D}_n)$
5: $\quad\quad$**for** $i = 1$ to $N_{\text{m-s}}$ **do**
6: $\quad\quad\quad\boldsymbol{x}_{n+1,i}, J_{n+1,i} \leftarrow$ `ALFIA`$(\boldsymbol{C}_n^{-1}, \hat{\boldsymbol{\theta}}_n, \boldsymbol{x}_i^{(1)}, \mathcal{D}_n, \gamma, \eta)$
7: $\quad\quad$**end for**
8: $\quad$**else** ▷ Exploitation
9: $\quad\quad\hat{\boldsymbol{\theta}}_n = \hat{\boldsymbol{\theta}}_{n-1}$; $\boldsymbol{C}_n^{-1} \leftarrow$ `fastUpdate`$(\boldsymbol{C}_{n-1}^{-1}, \hat{\boldsymbol{\theta}}_n, \boldsymbol{x}_n)$
10: $\quad\quad\boldsymbol{X}_{\text{ref}} \leftarrow$ `conditionLHS`$(N_{\text{ref}}, D, \boldsymbol{X}_n, \boldsymbol{X}_{\text{m-s}})$
11: $\quad\quad$**for** $i = 1$ to $N_{\text{m-s}}$ **do**
12: $\quad\quad\quad\boldsymbol{x}_{n+1,i}, J_{n+1,i} \leftarrow$ `ALC`$(\boldsymbol{C}_n^{-1}, \hat{\boldsymbol{\theta}}_n, \boldsymbol{x}_i, \boldsymbol{X}_n, \boldsymbol{X}_{\text{ref}}, \gamma, \eta)$
13: $\quad\quad$**end for**
14: $\quad$**end if**
15: $\quad\boldsymbol{x}_{n+1} \leftarrow$ `minJ`$(\{\boldsymbol{x}_{n+1,i}\}_{i=1}^{N_{\text{m-s}}}, \{J_{n+1,i}\}_{i=1}^{N_{\text{m-s}}})$ ▷ Next Input
16: $\quad y_{n+1} \leftarrow$ `sample`$(\boldsymbol{x}_{n+1})$
17: $\quad\hat{y}_{n+1} \leftarrow$ `GPprediction`$(\boldsymbol{C}_n^{-1}, \hat{\boldsymbol{\theta}}_n, \boldsymbol{x}_{n+1}, \mathcal{D}_n)$
18: $\quad\boldsymbol{y}_{n+1} = \boldsymbol{y}_n \cup y_{n+1}$; $\boldsymbol{X}_{n+1} = \boldsymbol{X}_n \cup \boldsymbol{x}_{n+1}$ ▷ New Dataset
19: **until** $n_{\max}$
20: **return** $\hat{\boldsymbol{\theta}}_{n_{\max}}, \mathcal{D}_{n_{\max}}$

For the exploitation phase (lines 9-13), we set the previous hyperparameters as the current $\hat{\boldsymbol{\theta}}_n = \hat{\boldsymbol{\theta}}_{n-1}$ and we compute the new inverse covariance matrix $\boldsymbol{C}_n^{-1}$ by using the partitioned inverse matrix [15, Chapter 6.3]. The reference set $\boldsymbol{X}_{\text{ref}}$ of ALC is designed using LHS, but conditioned on the current input locations $\boldsymbol{X}_n$ and the multi-start location dataset $\boldsymbol{X}_{\text{m-s}}$. Next, cALC is executed (Proposition 1, 2) that yields the next sample location candidates $\{\boldsymbol{x}_{n+1,i}\}_{i=1}^{N_{\text{m-s}}}$ and the corresponding acquisition function values $\{J_{n+1,i}\}_{i=1}^{N_{\text{m-s}}}$ for all multi-start locations. After obtaining the next sampling locations and the corresponding acquisition function values, we select the next input location $\boldsymbol{x}_{n+1}$ from the lowest acquisition function value. Then, we assign the next input location $\boldsymbol{x}_{n+1}$ and collect y_{n+1}, while concurrently we predict $\hat{y}_{n+1}$ with the GP hyperparameters $\hat{\boldsymbol{\theta}}_n$. The ALX method iterates for a predetermined number of $n_{\max}$ loops to output the hyperparameters of the GP surrogate model $\hat{\boldsymbol{\theta}}_{n_{\max}}$ and the final dataset $\mathcal{D}_{n_{\max}}$.

IV. Numerical Experiments

In this section, we provide numerical experiments to validate the efficacy of the proposed ALX methodology. We use numerous test functions with different input space dimensions: a) generative GP function using hyperparameter vector $\boldsymbol{\theta} = (l_1 = 1.9, l_2 = 0.6, \sigma_f = 2.1, \sigma_\epsilon = 0.1)^\intercal$ with input space dimension $D = 2$; b) Gramacy-Lee function with $D = 2$; c) Hartman function with $D = 3$; d) Rosenbrock function with $D = 4$; e) Hartman function with $D = 6$; and f) Sphere function with $D = 6$. Input locations are normalized to unit dimensions $\boldsymbol{x} \in [0, 1]^D$. Test functions (a), (c), and (e) are multimodal, while (b), (d), and (f) are unimodal. This set of test functions (a)–(f) is a common benchmark for the evaluation of active learning methods and GPs [7], [34]. The initial dataset has size $n_{\text{init}} = 10D$ and we set the size of multi-start locations to $N_{\text{m-s}} = 2D$ for all test functions. We compare six methods: i) batch GP with

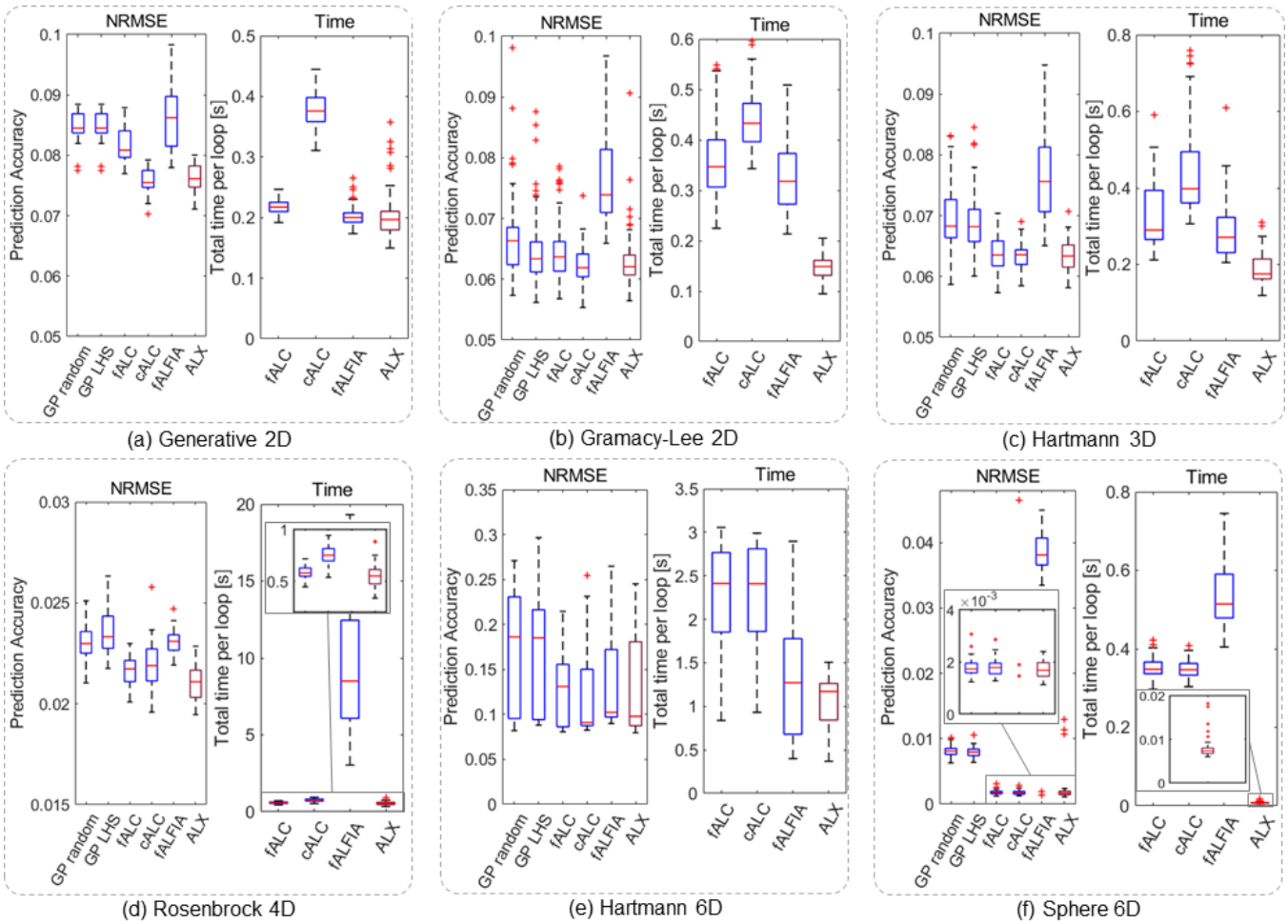

Fig. 3. Prediction accuracy in terms of NRMSE and total time per active learning loop for all test functions.

random selection of input locations $\boldsymbol{X}_n \sim \mathcal{U}_n[0,1]^D$ that is termed as GP-random; ii) batch GP with LHS selection of input locations $\boldsymbol{X}_n = \boldsymbol{X}_{\text{LHS},n}$, termed as GP-LHS; iii) approximated gradient ALC with finite differences, termed as fALC; iv) closed-form gradient ALC, termed as cALC; v) approximated gradient ALFIA with finite differences, termed as fALFIA; and vi) exploration-exploitation active learning, termed as ALX. We select $n_{\max} = 100$ loops of active learning for methods (iii)–(vi). The reference set for methods (iii), (iv) has size $N_{\text{ref}} = 15D$. All observations follow (1) with imposed iid noise $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$, where $\sigma_\epsilon = 0.1\,\text{std}(f(\boldsymbol{X}))$. The gradient descent step is set to $\gamma = 5$ and the convergence tolerance of the gradient descent to $\eta = 10^{-4}$ for methods (iii)–(vi). The exploration cost value of the ALX method (vi) is set to $\beta = 0.1$.

To remove the random assignment of data that may favor a specific method, we conduct 100 Monte Carlo replications for each test function (a)–(f). The numerical comparison involves the prediction NRMSE for methods (i)–(vi) and the required time for one loop for all active learning methods (iii)–(vi). The prediction NRMSE follows $\text{NRMSE} = [\sum_{s=1}^{n_S}(\mu(\boldsymbol{x}_{s,*}) - y(\boldsymbol{x}_{s,*}))^2]^{1/2}/(\max\{\boldsymbol{y}\} - \min\{\boldsymbol{y}\})$, with $n_S = 40,000$ the number of test points.

Best prediction accuracy & best scalability (Both 2D): The evaluation of all methods on the generative GP 2D and the Gramacy-Lee 2D is presented in Fig. 3-(a) and Fig. 3-(b) respectively. For the generative GP 2D, ALX and cALC outperform on median all other methods in prediction accuracy. However, ALX requires 47.6% lower time than cALC on median for every loop. For the Gramacy-Lee 2D, ALX, fALC, cALC, and GP-LHS produce the most accurate predictions. Yet, ALX entails 65.7% lower time than cALC and 57.1% lower time than fALC on median for every loop.

Best prediction accuracy & best scalability (3D): We compare all methods on the Hartmann 3D test function as shown in Fig. 3-(c). ALX, cLAC, and fALC are the most accurate method on median in prediction, but ALX requires 55.9% lower time than cALC and 39.5% lower time than fALC on median for every loop.

Best prediction accuracy & best scalability (4D): The efficiency of all methods on the Rosenbrock 4D test function is depicted in Fig. 3-(d). The proposed ALX outperforms all methods in prediction accuracy on median with cALC and fALC producing competitive predictions. ALX entails 5.0% less time than fALC and 26.5% less time than cALC on median for every loop.

Best prediction accuracy & best scalability (Both 6D): In Fig. 3-(e), -(f), we evaluate all methods on the Hartmann 6D and Sphere 6D test function respectively. ALX and cALC outperform on median all other methods in prediction accuracy, while fALFIA produces competitive predictions for the Hartmann 6D test function. The proposed ALX requires 51.4% less time than cALC and 8.0% lower time than fALFIA on median for every loop. For the Sphere 6D function, ALX, fALX, and cALC produce the most accurate predictions, while ALX requires two orders of magnitude lower time than fALC and cALC on median for every loop.

Overall, the results reveal that the proposed ALX method is consistently the most accurate predictor as well as consistently the most computationally efficient method. From the other methods, cALC is also consistently the best method for prediction accuracy along with ALX, but not computationally scalable. In addition, fALFIA is inconsistent both for prediction accuracy and computations. Moreover, fALC is competitive for prediction accuracy, but computationally expensive for higher input search spaces.

V. Conclusion

We synthesize an adaptive exploration-exploitation active learning method (ALX) and we formulate a new acquisition function that maximizes the approximated, expected Fisher information (ALFIA). The proposed ALX method is composed of two phases: i) exploration with the new ALFIA acquisition function; and ii) exploitation with the ALC acquisition function. ALX produces consistently the most accurate predictions in terms of NRMSE and requires the least time per active learning loop for its execution. Ongoing work is focusing on the decentralization of the proposed method with multi-agent systems.

References

[1] J. Das, F. Py, J. B. Harvey, J. P. Ryan, A. Gellene, R. Graham, D. A. Caron, K. Rajan, and G. S. Sukhatme, "Data-driven robotic sampling for marine ecosystem monitoring," *The International Journal of Robotics Research*, vol. 34, no. 12, pp. 1435–1452, 2015.

[2] V. Suryan and P. Tokekar, "Learning a spatial field in minimum time with a team of robots," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1562–1576, 2020.

[3] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Trans. on Pattern Analysis and Machine Intel.*, vol. 37, no. 2, pp. 408–423, 2013.

[4] W. Luo and K. Sycara, "Adaptive sampling and online learning in multi-robot sensor coverage with mixture of Gaussian processes," in *Intern. Conference on Robotics and Automation*, 2018, pp. 6359–6364.

[5] G. P. Kontoudis, S. Krauss, and D. J. Stilwell, "Model-based learning of underwater acoustic communication performance for marine robots," *Robotics and Autonomous Systems*, vol. 142, p. 103811, 2021.

[6] S. Seo, M. Wallat, T. Graepel, and K. Obermayer, "Gaussian process regression: Active data selection and test point rejection," in *IEEE-INNS-ENNS International Joint Conference on Neural Networks*, vol. 3, 2000, pp. 241–246.

[7] R. B. Gramacy and H. K. Lee, "Adaptive design and analysis of supercomputer experiments," *Technometrics*, vol. 51, no. 2, pp. 130–145, 2009.

[8] X. Yue, Y. Wen, J. H. Hunt, and J. Shi, "Active learning for Gaussian process considering uncertainties with application to shape control of composite fuselage," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 1, pp. 36–46, 2020.

[9] C. N. Mavridis, G. P. Kontoudis, and J. S. Baras, "Sparse Gaussian process regression using progressively growing learning representations," in *IEEE Conference on Decision and Control*, 2022, pp. 1454–1459.

[10] A. Krause and C. Guestrin, "Nonmyopic active learning of Gaussian processes: An exploration-exploitation approach," in *International Conference on Machine Learning*, 2007, pp. 449–456.

[11] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*, 2nd ed. Cambridge, MA, USA: MIT Press, 2006.

[12] G. P. Kontoudis and D. J. Stilwell, "Decentralized nested Gaussian processes for multi-robot systems," in *IEEE International Conference on Robotics and Automation*, 2021, pp. 8881–8887.

[13] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When Gaussian process meets big data: A review of scalable GPs," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4405–4423, 2020.

[14] G. P. Kontoudis and D. J. Stilwell, "Fully decentralized, scalable Gaussian processes for multi-agent federated learning," *arXiv preprint arXiv: 2203.02865*, 2022.

[15] R. B. Gramacy, *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Chapman Hall/CRC, 2020.

[16] G. P. Kontoudis and M. Otte, "Closed-form active learning using expected variance reduction of Gaussian process surrogates," in *American Control Conference*, 2023, pp. 4626–4632.

[17] Z. Zhu and M. L. Stein, "Spatial sampling design for parameter estimation of the covariance function," *Journal of Statistical Planning and Inference*, vol. 134, no. 2, pp. 583–603, 2005.

[18] H. Liu, Y.-S. Ong, and J. Cai, "A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design," *Structural and Multidisciplinary Optimization*, vol. 57, no. 1, pp. 393–416, 2018.

[19] J. Schreiter, D. Nguyen-Tuong, M. Eberts, B. Bischoff, H. Markert, and M. Toussaint, "Safe exploration for active learning with Gaussian processes," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2015, pp. 133–149.

[20] T. Samman, A. Dutta, O. P. Kreidl, S. Roy, and L. Bölöni, "Secure multi-robot information sampling with periodic and opportunistic connectivity," in *IEEE International Conference on Robotics and Automation*, 2022, pp. 4951–4957.

[21] M. Corah and N. Michael, "Efficient online multi-robot exploration via distributed sequential greedy assignment." in *Robotics: Science and Systems*, 2017.

[22] F. Ghassemi and V. Krishnamurthy, "Separable approximation for solving the sensor subset selection problem," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 1, pp. 557–568, 2011.

[23] Y. Xu and J. Choi, "Adaptive sampling for learning Gaussian processes using mobile sensor networks," *Sensors*, vol. 11, no. 3, pp. 3051–3066, 2011.

[24] B. Settles, "Active learning literature survey," University of Wisconsin-Madison, Tech. Rep. Computer Science technical report 1648, 2009.

[25] J. L. Loeppky, J. Sacks, and W. J. Welch, "Choosing the sample size of a computer experiment: A practical guide," *Technometrics*, vol. 51, no. 4, pp. 366–376, 2009.

[26] M. McKay, R. Beckman, and W. Conover, "Comparison the three methods for selecting values of input variable in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, 1979.

[27] M. Stein, "Large sample properties of simulations using latin hypercube sampling," *Technometrics*, vol. 29, no. 2, pp. 143–151, 1987.

[28] F. A. Viana, "Things you wanted to know about the latin hypercube design and were afraid to ask," in *World Congress on Structural and Multidisciplinary Optimization*, vol. 19, no. 24.05, 2013.

[29] A. Sobester, A. Forrester, and A. Keane, *Engineering design via surrogate modelling: A practical guide*. John Wiley & Sons, 2008.

[30] M. L. Stein, *Interpolation of spatial data: Some theory for kriging*, 1st ed. New York, NY, USA: Springer-Verlag, 1999.

[31] K. V. Mardia and R. J. Marshall, "Maximum likelihood estimation of models for residual covariance in spatial regression," *Biometrika*, vol. 71, no. 1, pp. 135–146, 1984.

[32] T. Wilson and S. B. Williams, "Active sample selection in scalar fields exhibiting non-stationary noise with parametric heteroscedastic Gaussian process regression," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 6455–6462.

[33] R. B. Gramacy and D. W. Apley, "Local Gaussian process approximation for large computer experiments," *Journal of Computational and Graphical Statistics*, vol. 24, no. 2, pp. 561–578, 2015.

[34] V. Picheny, T. Wagner, and D. Ginsbourger, "A benchmark of Kriging-based infill criteria for noisy optimization," *Structural and Multidisciplinary Optimization*, vol. 48, no. 3, pp. 607–626, 2013.